Penetration Test on AzireVPN for Malwarebytes Inc.

Final Report and Management Summary

2026-03-31

PUBLIC

X41 D-Sec GmbH
Soerser Weg 20
D-52070 Aachen
Amtsgericht Aachen: HRB19989

https://x41-dsec.de/
info@x41-dsec.de

Revision	*Date*	*Change*	*Author(s)*
1	2026-03-13	Errata	Djamal Touazi, JM, Markus Vervier and Robert Femmer
2	2026-02-20	Final Report and Management Summary	Djamal Touazi, JM, Markus Vervier and Robert Femmer
3	2026-02-09	Final Draft Report	Djamal Touazi, JM, Markus Vervier and Robert Femmer

Contents

Dashboard

Target	
Customer	Malwarebytes Inc.
Name	AzireVPN, Malwarebytes Privacy VPN
Type	Virtual Private Network Infrastructure
Version	As deployed between 2025-12-01 and 2026-01-09
Engagement	
Type	White-Box Penetration Test
Consultants	4: Djamal Touazi, JM, Markus Vervier and Robert Femmer
Engagement Effort	23 person-days, 2025-12-01 to 2026-01-09
Total issues found	14

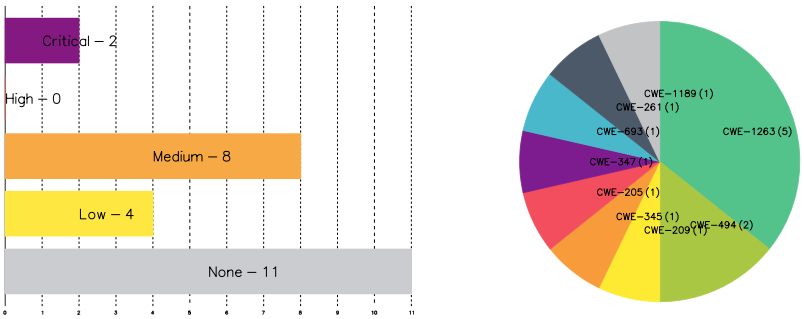

Figure 1: Issue Overview (l: Severity, r: CWE Distribution)

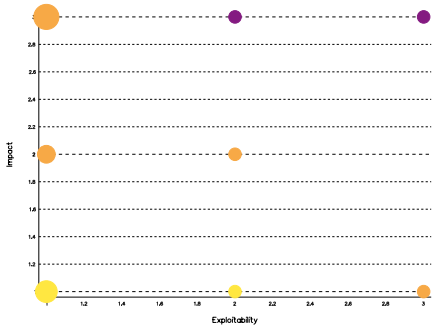

Figure 2: CVSS Impact and Exploitability Distribution

1 Executive Summary

Between December 2025 and January 2026, X41 D-Sec GmbH performed a source code audit and hardware penetration test against the AzireVPN services and relay server to identify vulnerabilities and weaknesses in the applications and hardware setup.

A total of 14 vulnerabilities were discovered during the test by X41. Two were rated as having a critical severity, none as high, eight as medium, and four as low. Additionally, eleven issues without a direct security impact were identified.

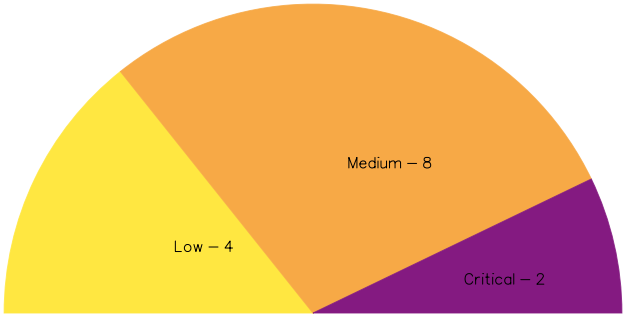

Figure 1.1: Issues and Severity

AzireVPN is a set of services that facilitates authentication, device setup, and payment processing for the network tunneling service. The reviewed infrastructure supports both AzireVPN and Malwarebytes Privacy VPN, services operated by Malwarebytes Inc. Vulnerabilities in these applications could potentially allow an attacker to access sensitive customer data, such as Internet connection metadata and payment information.

In a source code audit, all information about the system is made available. In a hardware penetration test the testers attempt to manipulate the physical environment of a system in a way to learn secrets or exert control on the software system running on the hardware. The test was performed by four experienced security experts between 2025-12-01 and 2026-01-09.

The most severe issue discovered allows an attacker to potentially craft arbitrary control messages to be executed by the VPN relay, if they manage to be able to communicate with the control plane directly. The vulnerability stems from a problematic usage of a cryptographic method which was intended to provide authenticity, but can only guarantee confidentiality. Despite being aware of issues with this approach, the code was allowed to remain unfixed in the code base, slated for refactoring at some future point in time. While mounting a successful attack against this particular vulnerability may be complex, one has to question the process by which it was decided to not fix the vulnerability as soon as possible.

On a positive note, the tooling reviewed in this audit produced Linux-based VPN server images that discarded log messages and had virtual terminals, serial consoles, and SSH access disabled when building production releases.

Additionally, the payment notification endpoint on the production system was found to lack the access controls that were present in the source code under review, allowing an attacker to forge payment notifications. This discrepancy between reviewed code and deployed system highlights the importance of verifying that security controls are consistently applied across environments.

The hardware penetration test revealed that while Malwarebytes Inc. has implemented physical hardening measures, these were found to be incomplete or bypassable. Several internal components were left unprotected, and X41 was able to recover the BIOS password and dump system memory using physical access alone. An attacker with brief physical access to a server in the data center could fully compromise it.

The Malwarebytes Inc. team was very responsive and proactive in addressing the identified issues.

Overall, the systems appear to be on a good security level compared to systems of similar size and complexity, though the identified issues, particularly in the supply chain, cryptographic implementation, and physical hardening, should be addressed promptly. The image produced by the build process of the provided source code is not deliberately configured to provide remote or local access.

2 Introduction

X41 reviewed AzireVPN, a virtual private network service allowing clients to use their infrastructure as a proxy for their Internet connections. Malwarebytes Inc. purchased AzireVPN and leverages its infrastructure for the Malwarebytes Privacy VPN service as well.

The service provides anonymity to clients and therefore the operation of services is considered sensitive, because a vulnerability or data leak may lead to exposing customers' PII[1] and put them in harm's way.

For example, attackers could try to attack the relay server and record traffic including connection metadata.

X41 does not give legal or business advice but recommends verifying findings where personal data disclosure is concerned against the regulations of the GDPR[2] and in particular the regulations against unauthorized disclosure of user data.

2.1 Methodology

The review was conducted as a source code review and white-box penetration test against the hardware as deployed in various data centers.

In a white-box penetration test, testers are given all information required to inspect the system on all levels necessary. This can include source code or administrative privileges to identify deeply-hidden vulnerabilities in a more efficient way.

A manual approach for penetration tests and for code reviews is used by X41. This process is supported by tools such as static code analyzers and industry standard web application security tools[3].

[1] Personally Identifiable Information
[2] `http://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A32016R0679`
[3] `https://portswigger.net/burp`

X41 adheres to established standards for source code reviewing and penetration testing. These are in particular the *CERT Secure Coding*[4] standards and the *Study - A Penetration Testing Model*[5] of the German Federal Office for Information Security.

The workflow of source code reviews is shown in figure 2.1. In an initial, informal workshop regarding the design and architecture of the application a basic threat model is created. This is used to explore the source code for interesting attack surface and code paths. These are then audited manually and with the help of tools such as static analyzers and fuzzers. The identified issues are documented and can be used in a GAP analysis to highlight changes to previous audits.

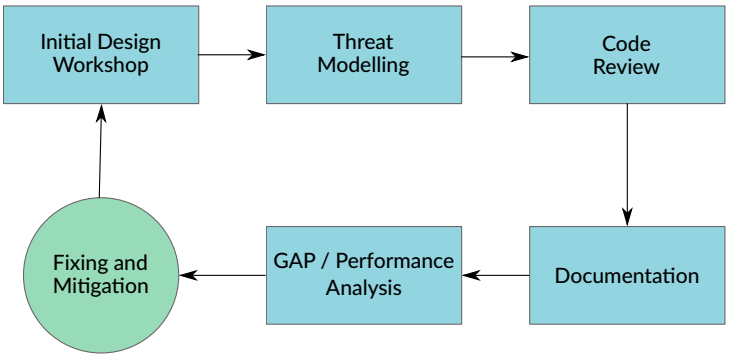

Figure 2.1: Code Review Methodology

[4] https://wiki.sei.cmu.edu/confluence/display/seccode/SEI+CERT+Coding+Standards
[5] https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/Studies/Penetration/penetration_pdf.pdf?__blob=publicationFile&v=1

2.2 Findings Overview

DESCRIPTION	SEVERITY	ID	REF
AES Middleware Implements a Padding Oracle	MEDIUM	AZR-PT-25-01	4.1.1
AES Encrypted Payloads Can be Replayed	LOW	AZR-PT-25-02	4.1.2
Possible Correlation of Forwarded Port and Client IP Address	LOW	AZR-PT-25-03	4.1.3
Unverified Debian Image	CRITICAL	AZR-PT-25-04	4.1.4
Unverified Boot Chain	CRITICAL	AZR-PT-25-05	4.1.5
Unverified Plain-Text DNS	MEDIUM	AZR-PT-25-06	4.1.6
Ineffective Physical Hardening of VGA Interface	LOW	AZR-PT-25-07	4.2.1
Weakness of Chassis Intrusion Detection Mechanism	LOW	AZR-PT-25-08	4.2.2
Incomplete Physical Hardening - USB 3.0 Header	MEDIUM	AZR-PT-25-09	4.2.3
Incomplete Physical Hardening - Clear CMOS Pads	MEDIUM	AZR-PT-25-10	4.2.4
Incomplete Physical Hardening - PLD1	MEDIUM	AZR-PT-25-11	4.2.5
Incomplete Physical Hardening - BIOS and BMC Flash Chips	MEDIUM	AZR-PT-25-12	4.2.6
Exploitation of Exposed BIOS Flash Chip - BIOS Password Recovery	MEDIUM	AZR-PT-25-13	4.2.7
Platform Vulnerable to Pre-Boot DMA Attacks	MEDIUM	AZR-PT-25-14	4.2.8
Return Value Unchecked	NONE	AZR-PT-25-100	4.3.1
Command Outside Chroot Environment	NONE	AZR-PT-25-101	4.3.2
Changing Email to Unverified Address Has No Consequences	NONE	AZR-PT-25-102	4.3.3
Use of Debian 12	NONE	AZR-PT-25-103	4.3.4
Use of Shell Script to Build Images	NONE	AZR-PT-25-104	4.3.5
Debian Package Signature Verification Not Enforced	NONE	AZR-PT-25-105	4.3.6
Immutable Log Unit Could Fail	NONE	AZR-PT-25-106	4.3.7
Build Automation	NONE	AZR-PT-25-107	4.3.8
Signed Commits	NONE	AZR-PT-25-108	4.3.9
Reducing Dependencies	NONE	AZR-PT-25-109	4.3.10
Unprotected Instant Payment Notification Endpoint	NONE	AZR-PT-25-110	4.3.11

Table 2.1: Security-Relevant Findings

2.3 Scope

The scope included the Virtual Private Network Infrastructure of AzireVPN. This included the client applications that customers install and run on their devices, the website and API[6] server, which are used to configure the VPN[7] access and handle payments as well as the physical hardware running the VPN gateway in data centers at various locations.

The source code for the client and server applications was provided as well as the configuration for the reverse proxy for the website and API. A server provisioned for production was shipped

[6] Application Programming Interface
[7] Virtual Private Network

to the office of X41 for a physical penetration test. The following assets were provided, listed with their respective SHA-1 sums.

audit.nov_14_snap.tar.zstd	`930d42bcba0b01f2c5803ec65f641bca202b7c44`
web4.test.azire.net-nginx-20251204-00.tar.bz2	`2ea1ee9a934410c78bb58fba0cf6e91a68993b2c`
test-env.zip	`67d9a89e8606a1c122ecddfbbf5756590d644e6b`

A Slack channel was set up for direct communication between developers and the testers.

2.4 Coverage

A security assessment attempts to find the most important or sometimes as many of the existing problems as possible, though it is practically never possible to rule out the possibility of additional weaknesses being identified in the future.

The time allocated to X41 for this assessment was sufficient to yield a basic but very broad coverage of the given scope.

The VPN server software was reviewed for security vulnerabilities and logic bugs. The website and API were reviewed for security issues commonly found in web applications. Specifically, the code base was reviewed for open redirects, SQL[8] injection, SSRF[9] and XSS[10] vulnerabilities. The review focused on instances of bugs which may allow an attacker to learn PII of users of the service. The Wireguard kernel module patches were reviewed for memory corruption issues and misuse of the Linux kernel API as well as logical bugs. Another focus was laid on possible attacks on the supply chain, specifically tracing the image for the VPN server from build to boot. The VPN client applications were audited for common security issues and correct use of the API provided by the respective operating system. This extends to the firewall configuration on Windows and MacOS.

The server was subjected to a physical penetration test that aimed at learning secret information like Wireguard encryption keys or the Wireguard private key by physically manipulating the server's hardware. These would allow an attacker to either decrypt user's traffic captured somewhere else or impersonate the server and receive the user's traffic directly.

[8] Structured Query Language
[9] Server-Side Request Forgery
[10] Cross-site Scripting

2.5 Recommended Further Tests

After resolving the findings in this report, it is recommended to retest them because the changes are likely to be fundamental to the application's security model.

Given the large scope of this audit, X41 recommends performing further tests with a narrower focus on certain parts of the scope to allow a more in-depth audit of each component.

Additionally, X41 recommends a test on the production infrastructure which hosts the applications to verify no vulnerabilities or weaknesses exist in the deployment and configuration.

3 Rating Methodology

Security vulnerabilities are given a purely technical rating by the testers when they are discovered during a test. Business factors and financial risks for Malwarebytes Inc. are beyond the scope of a penetration test, which focuses entirely on technical factors. However, technical results from a penetration test may be an integral part of a general risk assessment. A penetration test is based on a limited time frame and only covers vulnerabilities and security issues which have been found in the given time, there is no claim for full coverage.

The CVSS[1] is used to score all findings relevant to security. The resulting CVSS score is mapped to qualitative ratings as shown below.

3.1 CVSS

Testers rate all security-relevant findings using the CVSS industry standard version 4.

Vulnerabilities scored with CVSS get a numeric value based on several metrics ranging from 0.0 (least worst) to 10.0 (worst).

The score captures different factors that express the impact and the ease of exploitation of a vulnerability among other factors. For a detailed description of how the scores are calculated, please see the CVSS version 4.0 specification.[2]

The metrics used to calculate the final score are grouped into three different categories.

[1] Common Vulnerability Scoring System
[2] `https://www.first.org/cvss/v4.0/specification-document`

The *Base Metric Group* represents the intrinsic and fundamental characteristics of a vulnerability that are constant over time and user environments. It captures the following metrics:

- Attack Vector (AV)
- Attack Complexity (AC)
- Attack Requirements (AT)
- Privileges Required (PR)
- User Interaction (UI)
- Vulnerable/Subsequent System Confidentiality (VC/SC)
- Vulnerable/Subsequent System Integrity (VI/SI)
- Vulnerable/Subsequent System Availability (VA/SA)

The *Threat Metric Group* represents the current state of exploit techniques and the availability of proof of concepts. It captures the following metric:

- Exploit Maturity (E)

The *Environmental Metric Group* represents the characteristics of a vulnerability that are relevant and unique to a particular user's environment. It includes the following metrics:

- Confidentiality Requirement (CR)
- Integrity Requirement (IR)
- Availability Requirement (AR)
- Modified Attack Vector (MAV)
- Modified Attack Complexity (MAC)
- Modified Attack Requirements (MAC)
- Modified Privileges Required (MPR)
- Modified User Interaction (MUI)
- Modified Vulnerable System Confidentiality (MVC)
- Modified Vulnerable System Integrity (MVI)
- Modified Vulnerable System Availability (MVA)
- Modified Subsequent System Confidentiality (MSC)
- Modified Subsequent System Integrity (MSI)
- Modified Subsequent System Availability (MSA)

A CVSS vector defines a specific set of metrics and their values, and it can be used to reproduce and assess a given score. It is rendered as a string that exactly reproduces a score.

For example, the vector `CVSS:4.0/AV:N/AC:H/AT:N/PR:L/UI:A/VC:H/VI:L/VA:N/SC:N/SI:N/SA:N` defines a base score metric with the following parameters:

- Attack Vector: Network
- Attack Complexity: High
- Attack Requirements: None
- Privileges Required: Low
- User Interaction: Active
- Vulnerable System Confidentiality: High
- Vulnerable System Integrity: Low
- Vulnerable System Availability: None
- Subsequent System Confidentiality: None
- Subsequent System Integrity: None
- Subsequent System Availability: None

In this example, a network-based attacker performs a complex attack after gaining access to some privileges, by tricking a user into performing some actions. This allows the attacker to read confidential data and change some parts of that data.

The detailed scores are the following:

Metric	Score
Exploitability	Medium
Complexity	Medium
Vulnerable system	Medium
Subsequent system	Low
Exploitation	High
CVSS Score	5.8 (Medium)

CVSS vectors can be automatically parsed to recreate the score, for example, with the CVSS calculator provided by FIRST, the organization behind CVSS: `https://www.first.org/cvss/calculator/4.0#CVSS:4.0/AV:N/AC:H/AT:N/PR:L/UI:A/VC:H/VI:L/VA:N/SC:N/SI:N/SA:N`.

3.2 Severity Mapping

To help in understanding the results of a test, numeric CVSS scores are mapped to qualitative ratings as follows:

Severity Rating	CVSS Score
NONE	0.0
LOW	0.1–3.9
MEDIUM	4.0–6.9
HIGH	7.0–8.9
CRITICAL	9.0–10.0

3.3 Common Weakness Enumeration

The CWE[3] is a set of software weaknesses that allows vulnerabilities and weaknesses in software to be categorized. If applicable, X41 gives a CWE ID for each vulnerability that is discovered during a test.

CWE is a very powerful method for categorizing a vulnerability. It gives general descriptions and solution advice on recurring vulnerability types. CWE is developed by *MITRE*.[4] More information can be found on the CWE site at `https://cwe.mitre.org/`.

[3] Common Weakness Enumeration
[4] `https://www.mitre.org`

4 Results

This chapter describes the results of this test. The security-relevant findings are documented in Section 4.1. Additionally, findings without a direct security impact are documented in Section 4.3.

4.1 Findings

The following subsections describe findings with a direct security impact that were discovered during the test.

4.1.1 AZR-PT-25-01: AES Middleware Implements a Padding Oracle

Severity:	MEDIUM / 6.0
CVSS Vector:	CVSS:4.0/AV:A/AC:H/AT:N/PR:N/UI:N/VC:L/VI:H/VA:N/SC:N/SI:N/SA:N
CWE:	209 – Generation of Error Message Containing Sensitive Information
Component:	azire-wg-server/pkg/aes/aes.go:PKCS5UnPadding()

4.1.1.1 Description

The relay server and the Azire manager web server communicate via HTTPS[1]. To authenticate that requests to the relay server only originate from the Azire manager service, the requests are AES-CBC[2] encrypted using a pre-shared key. Since AES-CBC operates on blocks of 16 bytes, the last block must be padded to a 16 byte boundary before encryption. The padding is removed during decryption. The function used to remove the padding from the decrypted message is shown in listing 4.1.

[1] HyperText Transfer Protocol Secure
[2] Advanced Encryption Standard - Cipher Block Chaining

```
1 func PKCS5UnPadding(src []byte) []byte {
2     length := len(src)
3     unpadding := int(src[length-1])
4     return src[:(length - unpadding)]
5 }
```

Listing 4.1: PKCS5UnPadding() Is a Padding Oracle

Note that the last byte of the plaintext is used to strip the padding from the decrypted message given in *src*. If the last byte happens to be greater than the total length of the message, the routine panics, which can be observed by the client due to the connection dropping. While this is an imperfect padding oracle, since it does not fail if the padding byte happens to be smaller or equal to the length of (at least) one block (16 bytes), it can't be ruled out that with enough persistence a successful attack can be mounted. This may lead to an attacker crafting arbitrary payloads[3] which the relay server will successfully decrypt and service.

The attacker requires access to the separated management network to launch an attack.

4.1.1.2 Solution Advice

X41 recommends replacing AES-CBC with an authenticated encryption algorithm, such as AES-GCM[4]. In the long term, mTLS[5] may be considered to authenticate communication between *azire-manager* and *azire-wg-server*.

[3] https://static.usenix.org/events/woot10/tech/full_papers/Rizzo.pdf
[4] Advanced Encryption Standard - Galois Counter Mode
[5] mutual Transport Layer Security

4.1.2 AZR-PT-25-02: AES Encrypted Payloads Can be Replayed

Severity:	LOW / 1.0
CVSS Vector:	CVSS:4.0/AV:A/AC:L/AT:P/PR:H/UI:P/VC:N/VI:L/VA:N/SC:N/SI:N/SA:N
CWE:	345 – Insufficient Verification of Data Authenticity
Component:	azire-wg-server/internal/api/middleware/decrypt.go:DecryptMiddleware()

4.1.2.1 Description

The relay server and the Azire manager web server communicate via HTTPS. To authenticate that requests to the relay server only originate from the Azire manager service, the requests are AES-CBC encrypted using a pre-shared key. This scheme is not resistant against replay attacks. An attacker who has learned the ciphertext of one message to the relay server is able to replay this message as there are no guarantees of authentication, only confidentiality. This issue was known to the authors of the code prior to this audit as is evident from the commentary of the code.

The attacker requires access to the separated management network to launch an attack.

4.1.2.2 Solution Advice

X41 recommends replacing AES-CBC with an authenticated encryption algorithm, such as AES-GCM, which would also address the padding oracle described in the previous finding. In the long term, mTLS may be considered to authenticate communication between `azire-manager` and `azire-wg-server`.

4.1.3 AZR-PT-25-03: Possible Correlation of Forwarded Port and Client IP Address

Severity:	LOW / 2.1
CVSS Vector:	CVSS:4.0/AV:N/AC:L/AT:P/PR:N/UI:A/VC:L/VI:N/VA:N/SC:N/SI:N/SA:N
CWE:	205 – Observable Behavioral Discrepancy
Component:	azire-wg-server

4.1.3.1 Description

The port forwarding feature allows a user to forward a port on the relay server to the same port number bound to their wireguard interface. If the port on their wireguard interface is also exposed on the regular network interface and therefore the port is exposed to the Internet, an attacker may learn the real client IP[6] address by correlating TCP[7] sequence number differentials of the port opened on the relay and the set of opened ports with the same number on the Internet which can be enumerated in the case of IPv4[8].

4.1.3.2 Solution Advice

X41 recommends mentioning this issue in the threat model available to users. Further, X41 recommends offering the user to specify a destination port for the port forwarding rule, so that they may choose a port that is different from the port on the relay server, effectively decreasing the likelihood that an attacker may successfully correlate the forwarded port and the user's IP address.

[6] Internet Protocol
[7] Transmission Control Protocol
[8] Internet Protocol Version 4

4.1.4 AZR-PT-25-04: Unverified Debian Image

Severity:	CRITICAL / 9.4
CVSS Vector:	CVSS:4.0/AV:N/AC:H/AT:P/PR:N/UI:N/VC:H/VI:H/VA:N/SC:H/SI:N/SA:N
CWE:	494 – Download of Code Without Integrity Check
Component:	audit/sources/azire-pxe/wgimagetk.sh

4.1.4.1 Description

The Debian image (*debian-live-12.12.0-amd64-standard.iso*) downloaded in the build process for VPN server images is obtained via HTTPS from `https://cdimage.debian.org/mirror/cdimage/archive/12.12.0-live/amd64/iso-hybrid/` and its checksum is verified, however the signature of the checksum file is not validated against the Debian CD signing key[9].

An attacker with control over the remote host could modify the image, resulting in a complete compromise of the produced server image.

X41 would like to point out that while CVSS gives this a *CRITICAL* rating due to the severe consequences, the issue is hard to exploit in a real-world scenario. However, Linux distribution servers were hijacked in the past, such as the Linux Mint distribution in 2016[10]. For example, the host could be taken over in BGP[11] hijacks, IP address takeovers by attackers adjacent to the CDN[12] host or its network path, or attacks against DNS[13]. The IPv4 address of `cdimage.debian.org` (*CNAME* `mirror.accum.se`) does not seem[14] to be secured using RPKI[15], and the hostname's CAA[16] record only limits the CA[17] to Let's Encrypt[18] without limiting it to a certain account, providing little protection.

4.1.4.2 Solution Advice

X41 recommends following the instructions found at `https://www.debian.org/CD/verify` in order to verify the checksum signature file.

[9] `https://www.debian.org/CD/verify`
[10] `https://blog.linuxmint.com/?p=2994`
[11] Border Gateway Protocol (routing protocol)
[12] Content Delivery Network
[13] Domain Name System
[14] `https://rpki-validator.ripe.net/ui/?prefix=194.71.11.0%2F24&asns=AS1653`
[15] Resource Public Key Infrastructure
[16] Certification Authority Authorization
[17] Certificate Authority
[18] `https://letsencrypt.org`

4.1.5 AZR-PT-25-05: Unverified Boot Chain

Severity:	CRITICAL / 9.3
CVSS Vector:	CVSS:4.0/AV:A/AC:L/AT:N/PR:N/UI:N/VC:H/VI:H/VA:N/SC:H/SI:N/SA:N
CWE:	494 – Download of Code Without Integrity Check
Component:	audit/sources/azire-pxe/README.md

4.1.5.1 Description

According to the documentation provided by Malwarebytes Inc., the VPN servers perform a network boot using PXELINUX[19], which involves DHCP[20], HTTP[21], and/or TFTP[22].

The PXELINUX RFC[23] states:

> PXE and PXELINUX allow any entity acting as a DHCP server to execute arbitrary code upon a system. At present, no PXE implementation is known to implement authentication mechanisms [[24]] so that PXE clients can be sure they receive configuration information from the correct, authoritative DHCP server.
>
> The use of TFTP by PXE and PXELINUX also lacks any form of cryptographic signature – so a 'Man in the Middle' attack may lead to an attacker's code being executed on the client system. Since this is not an encrypted channel, any of the TFTP loaded data may also be exposed (such as in loading a "RAMDISK" image, which contains /etc/passwd or similar information).
>
> RFC 5071, Security Considerations[25]

The attack requires an attacker to physically access the network cable between VPN server and Mgmt Router (located in the same server rack), or gain control over the Mgmt Router itself, PXE[26] boot server, or the Wireguard tunnel between Mgmt Router and PXE boot server.

Furthermore, with physical access to the server, the current protections do not prevent an attacker from altering the boot flow or temporarily booting from another PXE server. The boot

[19] `https://wiki.syslinux.org/wiki/index.php?title=PXELINUX`
[20] Dynamic Host Configuration Protocol
[21] HyperText Transfer Protocol
[22] Trivial File Transfer Protocol
[23] Request for Comments
[24] `https://datatracker.ietf.org/doc/html/rfc3118`
[25] `https://datatracker.ietf.org/doc/html/rfc5071#section-8`
[26] Preboot Execution Environment

order and BIOS[27] settings can be modified either directly through the BIOS interface (after recovering the password as described in AZR-PT-25-113) or by manipulating NVRAM[28] entries in memory via a DMA[29] attack as described in AZR-PT-25-114.

4.1.5.2 Solution Advice

X41 recommends using a setup that verifies data obtained via network against code signing certificates. Additionally, file system data should be encrypted to prevent a network adjacent attacker from obtaining sensitive information. Using iPXE[30] with the *imgdecrypt* and/or *imgverify* may be an option.

[27] Basic Input/Output System
[28] Non-volatile RAM
[29] Direct Memory Access
[30] `https://ipxe.org/crypto`

4.1.6 AZR-PT-25-06: Unverified Plain-Text DNS

Severity:	MEDIUM / 6.3
CVSS Vector:	CVSS:4.0/AV:N/AC:L/AT:P/PR:N/UI:N/VC:N/VI:L/VA:N/SC:N/SI:N/SA:N
CWE:	347 – Improper Verification of Cryptographic Signature
Component:	audit/sources/azire-pxe/wgimagetk.sh

4.1.6.1 Description

The VPN server images are configured to trust remote DNS resolvers in the */etc/resolv.conf* file without using a secure channel.

The traffic may be observed and possibly manipulated by a network attacker.

4.1.6.2 Solution Advice

X41 recommends using DoH[31] or DoT[32] with a trustworthy validating DNS resolver and instructing the resolver to validate DNSSEC[33], or validating DNSSEC locally.

systemd-resolved could be used with the *DNSOverTLS*[34] and *DNSSEC*[35] options set to *true*. See also the *trust-ad*[36] option in */etc/resolv.conf*, notably "the local system has to trust both the DNSSEC-validating resolver and the network path to it, which is why an explicit opt-in is required."

[31] DNS over HTTPS
[32] DNS over TLS
[33] Domain Name System Security Extensions
[34] https://www.freedesktop.org/software/systemd/man/latest/resolved.conf.html#DNSOverTLS=
[35] https://www.freedesktop.org/software/systemd/man/latest/resolved.conf.html#DNSSEC=
[36] https://manpages.debian.org/trixie/manpages/resolv.conf.5.en.html#trust

4.2 Hardware Security Findings

X41 was tasked with reviewing the physical security of the servers and the hardening mechanisms implemented by Malwarebytes Inc.. All servers are owned and provisioned by Malwarebytes Inc., who does not use third-party or shared hosting infrastructure.

The assessed platform is based on a Supermicro X10DRU-i+ motherboard (Revision 1.02b). BIOS version: 3.5. Intel ME[37] firmware version: 3.1.3.131.

X41 identified the following physical hardening measures:

- Epoxy encapsulation applied to the rear I/O ports of the server chassis, covering the VGA[38], serial, and USB[39] interfaces.
- Epoxy potting of the internal USB port to prevent unauthorized peripheral connection.
- Epoxy fixation of the CR2032 CMOS[40] battery to mitigate removal or manipulation.
- Integration of chassis intrusion detection switch.

X41 identified that a pre-boot authentication password was enforced. Furthermore, security-relevant BIOS settings were hardened, including the disabling of all Super I/O interfaces (e.g., serial ports), the limitation of boot sources to PXE exclusively.

4.2.1 AZR-PT-25-07: Ineffective Physical Hardening of VGA Interface

Severity:	LOW / 2.4
CVSS Vector:	CVSS:4.0/AV:P/AC:L/AT:N/PR:N/UI:N/VC:L/VI:N/VA:N/SC:N/SI:N/SA:N
CWE:	1263 – Improper Physical Access Control
Component:	Supermicro X10DRU-i+ rear I/O VGA port

4.2.1.1 Description

X41 was able to restore access to the VGA port by mechanically removing the epoxy using a rotary tool (Dremel) equipped with a milling bit and a small polishing drill bit.

The operation was completed without significant difficulty and lasted approximately 10-15 minutes.

[37] Management Engine
[38] Video Graphics Array
[39] Universal Serial Bus
[40] Complementary metal–oxide–semiconductor

4.2.1.2 Solution Advice

Due to the mechanical design of the VGA connector, X41 assesses that fully disabling the port through epoxy encapsulation alone is ineffective, as the epoxy can be removed with minimal effort using basic tooling.

X41 identified two remaining mitigation options:

- The preferred option is to mechanically disable the interface by drilling out pins 13 and 14 (H-SYNC and V-SYNC). This approach is deemed the safest, as these signals are buffered and ESD[41]-protected by a dedicated protection device (CM2006-02QR) on the mother-board. Consequently, any unintended short circuit resulting from imprecise drilling would be expected to have minimal impact on overall board integrity or on the VGA signal routed to the IPMI[42] (as identified in Figure 4.1). The port should then be encapsulated with epoxy as before.
- The second option is to cut the PCB[43] trace leading to the VGA connector at the specific location indicated in Figure 4.2 (red marking). Any modification performed downstream of the decoupling capacitor may negatively affect the integrity of the video signal captured by the IPMI subsystem.

Figure 4.1: Mechanical Disablement Points on VGA Connector (Pins 13-14)

[41] Electrostatic Discharge
[42] Intelligent Platform Management Interface
[43] Printed Circuit Board

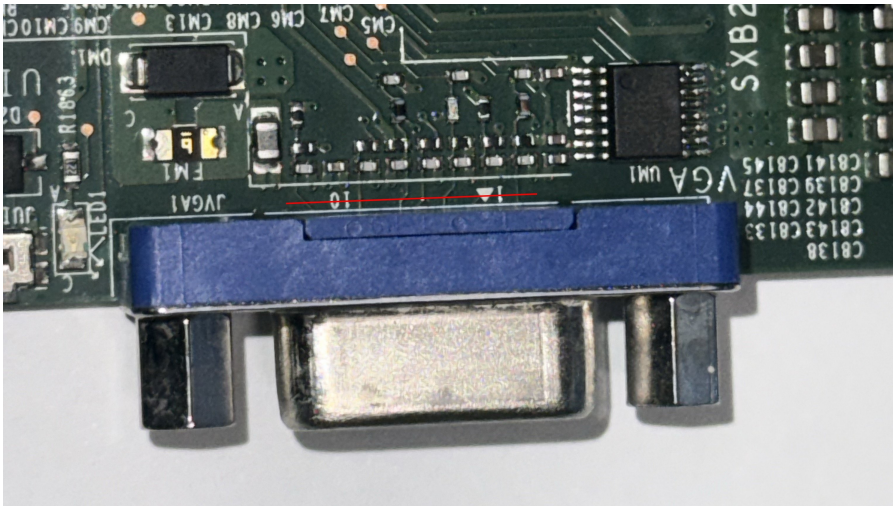

Figure 4.2: VGA Signal Trace - Recommended Cut Location

4.2.2 AZR-PT-25-08: Weakness of Chassis Intrusion Detection Mechanism

Severity:	LOW / 2.4
CVSS Vector:	CVSS:4.0/AV:P/AC:L/AT:N/PR:N/UI:N/VC:N/VI:L/VA:N/SC:N/SI:N/SA:N
CWE:	693 – Protection Mechanism Failure
Component:	Supermicro X10DRU-i+ chassis intrusion header

4.2.2.1 Description

Chassis intrusion detection is intended to identify physical tampering of the system enclosure. This mechanism is used by Malwarebytes Inc. as a condition to determine whether its software is permitted to execute.

X41 identified several weaknesses in the current implementation. The intrusion detection mechanism relies on detecting a short-circuit condition across the motherboard header pins. Under normal operating conditions, the circuit remains open. When the chassis is opened and the mechanical switch is released, the circuit closes, electrically shorting the two header pins.

This design introduces two major issues:

- If an attacker cuts only one of the wires or disconnects the intrusion header, the event will not be detected, as the sensing logic relies solely on a short-circuit condition.
- The effectiveness of the mechanism is further reduced by the physical placement of both the motherboard intrusion header and the chassis intrusion sensor on the left edge of the

1U enclosure (as shown in Figure 4.3).

The presence of chassis rail slots/openings allows the wire to be fished from outside the enclosure.

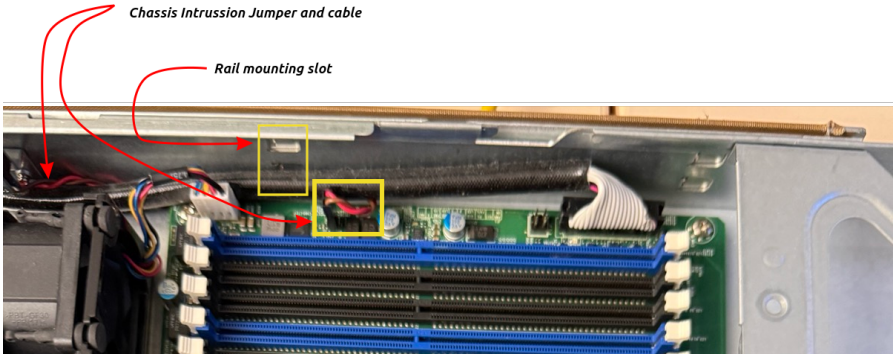

Figure 4.3: Chassis Intrusion Header and Wiring Location

Additionally, the sensor can be mechanically forced into the closed position by inserting a thin plastic card or similar object between the lid and the chassis, thereby preventing the switch from triggering.

4.2.2.2 Solution Advice

X41 recommends running the wire through a semi-rigid plastic straw to prevent it from being easily fished out and cut. We also recommend securing the header in place using hot glue or epoxy. Additionally, applying hot glue around part of the chassis before closing the lid would create a seal, making it more difficult to insert an object and tamper with the mechanism.

X41 could not recommend making an optical chassis intrusion detection device, as Google holds a patent on this type of device.

4.2.3 AZR-PT-25-09: Incomplete Physical Hardening - USB 3.0 Header

Severity:	MEDIUM / 5.3
CVSS Vector:	CVSS:4.0/AV:P/AC:L/AT:P/PR:N/UI:N/VC:H/VI:H/VA:N/SC:N/SI:N/SA:N
CWE:	1263 – Improper Physical Access Control
Component:	Supermicro X10DRU-i+ USB 3.0 header

4.2.3.1 Description

During the review of the motherboard hardening, X41 observed that a USB 3.0 header was left without epoxy encapsulation. This was unexpected, as the adjacent USB 3.0 port was properly encapsulated as shown in Figure 4.4.

As this was the only USB header present on the board and the sole USB interface left unprotected, X41 assumes this was an oversight. It should be noted that USB interfaces are disabled at the BIOS level. An attacker would need to first modify the BIOS configuration in order to make use of the exposed header.

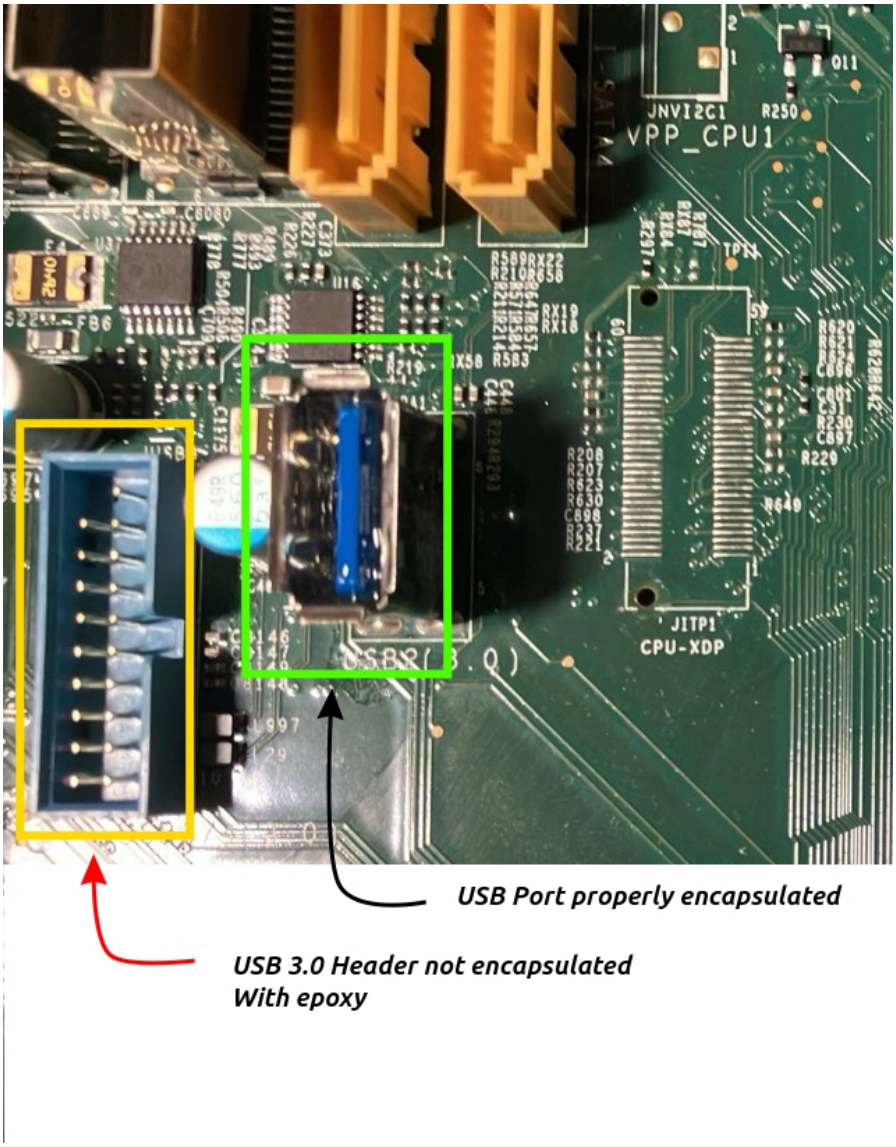

Figure 4.4: Exposed USB 3.0 Header

4.2.3.2 Solution Advice

X41 recommends encapsulating this header in the same manner as the other USB port and consulting the motherboard manual[44].

[44] https://www.supermicro.com/manuals/motherboard/C612/MNL-1597.pdf

4.2.4 AZR-PT-25-10: Incomplete Physical Hardening - Clear CMOS Pads

Severity:	MEDIUM / 5.1
CVSS Vector:	CVSS:4.0/AV:P/AC:L/AT:N/PR:N/UI:N/VC:N/VI:H/VA:N/SC:N/SI:N/SA:N
CWE:	1263 – Improper Physical Access Control
Component:	Supermicro X10DRU-i+ Clear CMOS pads (JBT1)

4.2.4.1 Description

During the review of the motherboard hardening, X41 observed that the Clear CMOS pads were left without epoxy encapsulation. This was unexpected, as the adjacent CMOS battery was properly potted, as shown in Figure 4.5.

This indicates that the motherboard documentation was not thoroughly reviewed with respect to the Clear CMOS functionality.

Leaving the Clear CMOS pads accessible allows an attacker with physical access to reset the BIOS configuration, including removal of the BIOS password. While this action is not stealthy, when combined with the previously identified issues, it could enable an attacker to recover a fully operational system.

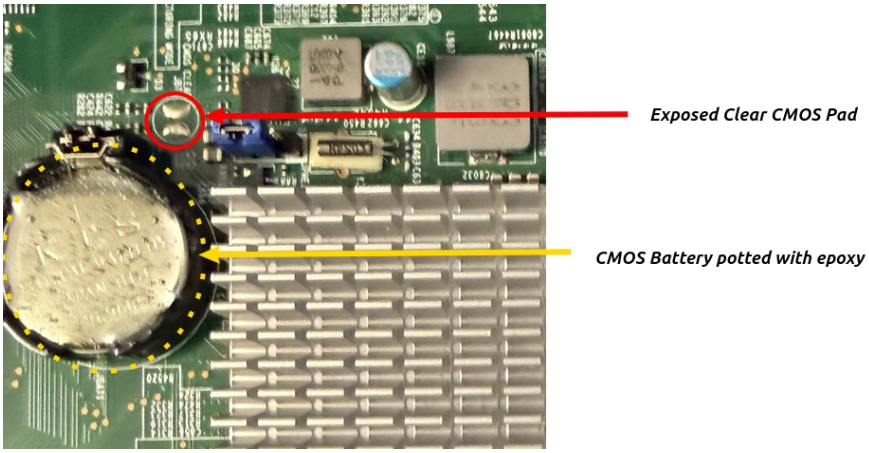

Figure 4.5: Exposed Clear CMOS Pads

4.2.4.2 Solution Advice

X41 recommends encapsulating the Clear CMOS pads with epoxy to prevent unauthorized access. Additionally, the motherboard manual[45] should be carefully reviewed to ensure that all configuration-reset mechanisms are properly identified and secured as part of the physical hardening process.

[45] `https://www.supermicro.com/manuals/motherboard/C612/MNL-1597.pdf`

4.2.5 AZR-PT-25-11: Incomplete Physical Hardening - PLD1

Severity:	MEDIUM / 5.3
CVSS Vector:	CVSS:4.0/AV:P/AC:L/AT:P/PR:N/UI:N/VC:H/VI:H/VA:N/SC:N/SI:N/SA:N
CWE:	1263 – Improper Physical Access Control
Component:	Supermicro X10DRU-i+ PLD1 header

4.2.5.1 Description

During the review of the motherboard hardening, X41 observed that the PLD1 header was left unprotected, as shown in Figure 4.6.

The CPLD[46] interface exposed through the PLD1 header allows direct interaction with the flash devices, both the BIOS SPI[47] flash chip and the BMC[48] SPI flash chip, provided you have the required engineering hardware from Supermicro.

[46] Complex Programmable Logic Device
[47] Serial Peripheral Interface
[48] Baseboard Management Controller

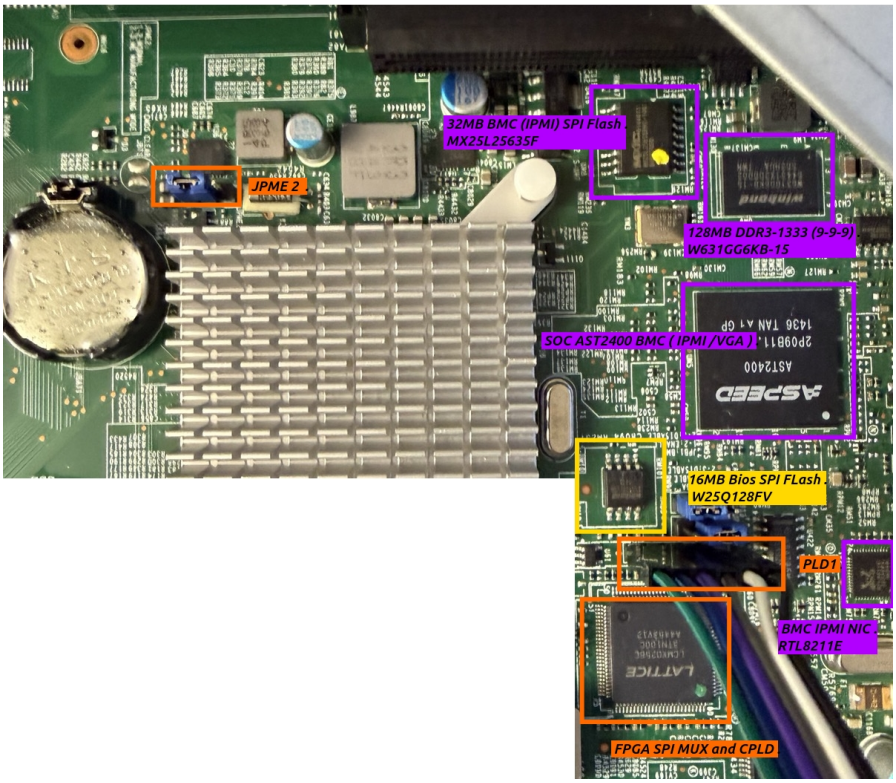

Figure 4.6: South Bridge Area Breakout Diagram

4.2.5.2 Solution Advice

X41 recommends encapsulating this header with epoxy to restrict unauthorized access. Additionally, the motherboard manual[49] should be carefully reviewed to ensure that all engineering mode mechanisms are properly identified and secured as part of the physical hardening process.

[49] `https://www.supermicro.com/manuals/motherboard/C612/MNL-1597.pdf`

4.2.6 AZR-PT-25-12: Incomplete Physical Hardening - BIOS and BMC Flash Chips

Severity:	MEDIUM / 6.8
CVSS Vector:	CVSS:4.0/AV:P/AC:L/AT:N/PR:N/UI:N/VC:H/VI:H/VA:N/SC:N/SI:N/SA:N
CWE:	1263 – Improper Physical Access Control
Component:	Supermicro X10DRU-i+ W25Q128FV / MX25L25635F

4.2.6.1 Description

During the review of the motherboard hardening, X41 observed that both the BIOS flash chip (W25Q128FV) and the BMC flash chip (MX25L25635F) were left unprotected, as shown in Figure 4.7.

These chips respectively store the UEFI[50]/BIOS firmware and the BMC firmware, which manages multiple platform features, including IPMI functionality.

Leaving these components physically accessible allows an attacker with appropriate equipment to directly read, modify, or replace the firmware, potentially bypassing platform security controls.

[50] Unified Extensible Firmware Interface

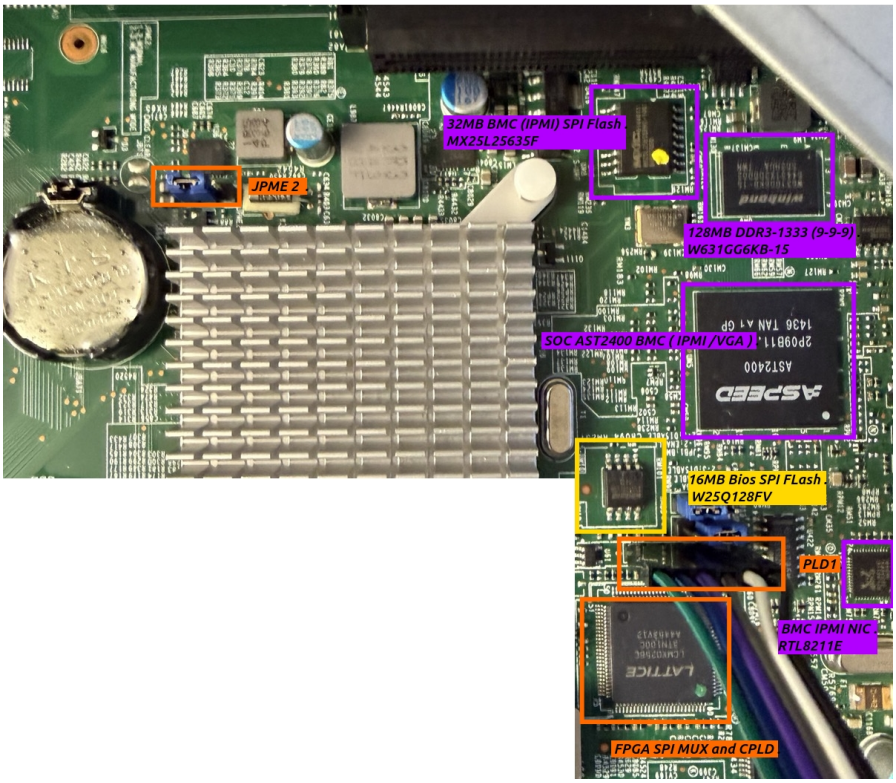

Figure 4.7: Flash Chip Locations on Motherboard

4.2.6.2 Solution Advice

X41 recommends encapsulating both flash chips with epoxy to restrict unauthorized physical access and reduce the risk of direct firmware tampering.

4.2.7 AZR-PT-25-13: Exploitation of Exposed BIOS Flash Chip - BIOS Password Recovery

Severity:	MEDIUM / 6.7
CVSS Vector:	CVSS:4.0/AV:P/AC:L/AT:N/PR:N/UI:N/VC:H/VI:H/VA:N/SC:N/SI:N/SA:N/-MVC:L/MVI:H/MSI:H
CWE:	261 – Weak Encoding for Password
Component:	Supermicro X10DRU-i+ BIOS SPI flash (W25Q128FV)

4.2.7.1 Description

During the review of the motherboard hardening, X41 observed that the BIOS flash chip (W25Q128FV) was physically accessible. X41 leveraged this access to read the BIOS contents in a stealthy manner while preserving the platform configuration used by Malwarebytes Inc..

To streamline acquisition and reprogramming, X41 implemented a simple in-situ interface on the server that enabled fast read/write access to the BIOS flash chip, as shown in Figure 4.8. X41 then collected multiple BIOS images corresponding to different BIOS password values.

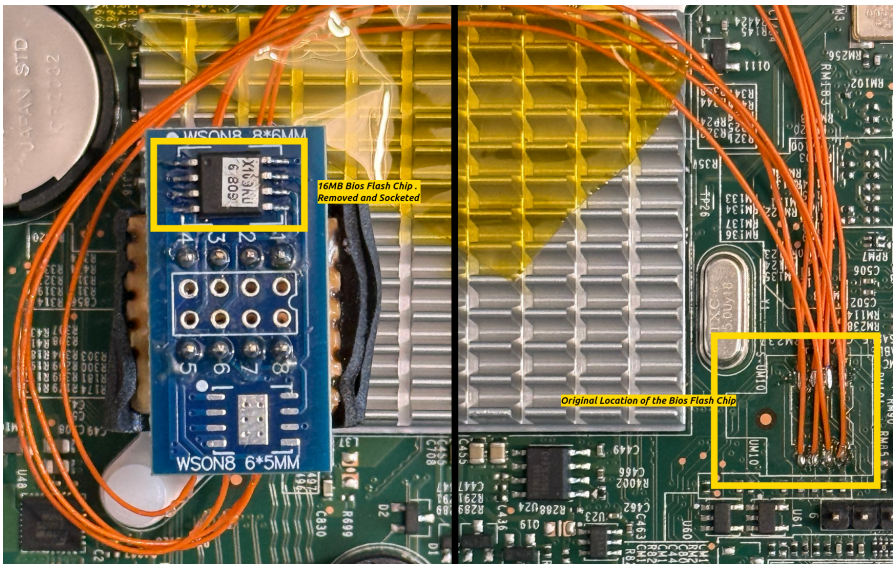

Figure 4.8: In-situ BIOS Flash Interface

The acquired images were unpacked using *UEFIExtract*[51] (UEFI Tools suite) in order to extract the underlying firmware components, including the NVRAM store.

[51] `https://github.com/LongSoft/UEFITool/releases`

By performing basic diffing across the extracted NVRAM contents from the different samples, X41 was able to identify the specific NVRAM entry used to store the BIOS password as shown in listing 4.2:

```
    GUID: C811FA38-42C8-4579-A9BB-60E94EDDFB34
    Name: AMITSESetup
```

Listing 4.2: NVRAM Entry

With this information, we returned to the previously acquired firmware dump and performed a direct hexadecimal comparison. We observed that the number of modified bytes matched the number of characters used in the configured password, as illustrated in Listing 4.3.

```

xxd -s 0x32 -l 40  rrr.bin  (password is rrr)
00000032: 2993 c426 63ba 6c4d c7e0 2274 7d07 d89a  )..&c.lM.."t}...
00000042: 332e 8ec1 e954 44e8 9f7b fa0e 55a2 b035  3....TD..{..U..5
00000052: 0bc9 665c c1ef 1c83                      ..f\....

xxd -s 0x32 -l 40  eee.bin (password is eee)
00000032: 3e93 d326 74ba 6c4d c7e0 2274 7d07 d89a  >..&t.lM.."t}...
00000042: 332e 8ec1 e954 44e8 9f7b fa0e 55a2 b035  3....TD..{..U..5
00000052: 0bc9 665c c1ef 1c83                      ..f\....

For clarity, a byte-level comparison can also be obtained using:

cmp -l eee.bin rrr.bin | awk '{printf "0x%X : 0x%X -> 0x%X\n", $1-1, strtonum("0"$2),
↪  strtonum("0"$3)}'

0x32 : 0x3E -> 0x29
0x34 : 0xD3 -> 0xC4
0x36 : 0x74 -> 0x63
```

Listing 4.3: Comparing Hex Values for Passwords `eee` and `rrr`

Only three bytes changed between the two firmware images, corresponding exactly to the three-character password length. This behavior was consistent across all tested password lengths (from 3 to 20 characters).

The modification pattern suggested a simple XOR-based transformation. Assuming that unused positions were XORed with zero, we tested this hypothesis using longer passwords.

For example, when configuring the password *eeee*, we observed that the byte at offset *0x38* had

the value *0x09*. By XORing this value with *0x6C*, which was observed at the same offset for the previous password *eee*, we obtained *0x65*, corresponding to the ASCII character *e*.

This strongly indicates that the BIOS password is stored using a reversible XOR-based encoding scheme. From this observation, we derived the missing key bytes by XORing *0x3E*, *0xD3*, and *0x74* with *0x65*, resulting respectively in *0x5B*, *0xB6*, and *0x11*.

This allowed reconstruction of the complete XOR key:

```
5B93B62611BA6C4DC7E022747D07D89A332E8EC1E95444E89F7BFA0E55A2B0350BC9665CC1EF1C83
```

We then applied this key to the original NVRAM dump:

```
0893D72672BA194DB5E067744F07ED9A332E8EC1E95444E89F7BFA0E55A2B0350BC9665CC1EF1C83
```

The resulting decoded value was:

```
53006100630075007200450032003500000000000000000000000000000000000000000000000000000000
```

Interpreting this as UTF-16 little-endian encoding (or stripping the interleaved null bytes) yields the ASCII string: **SacurE25**

4.2.7.2 Solution Advice

X41 recommends encapsulating the BIOS flash chip to prevent unauthorized physical access. Additionally, a unique BIOS password should be configured for each server to mitigate the risk of large-scale compromise in the event that a single password is recovered.

4.2.8 AZR-PT-25-14: Platform Vulnerable to Pre-Boot DMA Attacks

Severity:	MEDIUM / 6.8
CVSS Vector:	CVSS:4.0/AV:P/AC:L/AT:N/PR:N/UI:N/VC:H/VI:H/VA:N/SC:N/SI:N/SA:N
CWE:	1189 – Improper Isolation of Shared Resources on System-on-a-Chip (SoC)
Component:	Supermicro X10DRU-i+ PCIe slots

4.2.8.1 Description

During testing, X41 confirmed that pre-boot DMA attacks remain possible on this generation of Intel CPU[52] and UEFI firmware.

Using a trusted PCI Squirrel[53] device in combination with the PCILeech[54] framework, we were able to dump the first 4 GB of system memory prior to BIOS password entry. As the PCIe[55] ports were not physically hardened, the device could be connected directly, as shown in Figure 4.9.

To validate the integrity of the memory dump, we analyzed it and successfully located the NVRAM region containing the BIOS password entry, as illustrated in Listing 4.4.

```
1
2 xxd -s 0x7BA87409 -l 40 pcileech-0-100000000-20260108-145504.raw
3
4 7ba87409: 0893 d726 72ba 194d b5e0 6774 4f07 ed9a  ...&r..M..gtO...
5 7ba87419: 332e 8ec1 e954 44e8 9f7b fa0e 55a2 b035  3....TD..{..U..5
6 7ba87429: 0bc9 665c c1ef 1c83                      ..f\....
7
```

Listing 4.4: Memory Dump - NVRAM Entry of Admin Password

This demonstrates that sensitive firmware configuration data is accessible via DMA prior to authentication. An attacker could either recover the BIOS password offline or modify NVRAM entries directly in memory before credential verification occurs.

Beyond password recovery, additional NVRAM variables such as the *IntelSetup* store can be modified to disable security mechanisms including VT-d[56] and IOMMU[57] protections without altering persistent BIOS settings.

[52] Central Processing Unit
[53] https://docs.lambdaconcept.com/screamer/index.html
[54] https://github.com/ufrisk/pcileech
[55] Peripheral Component Interconnect Express
[56] Intel Virtualization Technology for Directed I/O
[57] Input/Output Memory Management Unit

After disabling DMA protections in this manner, X41 successfully performed a post-boot memory dump of the running system. While X41 was not able to fully demonstrate extraction of the WireGuard private key during the engagement due to a network issue that prevented loading the key onto the server, the successful post-boot memory dump shows that memory extraction is feasible. If the WireGuard private key were loaded in memory, extracting it would be a realistic attack scenario.

The listing below illustrates the byte-level differences required to disable VT-d and IOMMU protections within the *IntelSetup* NVRAM store:

```

cmp -l ../NVram/nvram_IntelSetup.bin ../NVram/nvram_IntelSetup_Disabled.bin | awk '{printf "0x%X : 0x%X -> 0x%X\n", $1-1, strtonum("0"$2), strtonum("0"$3)}'
0x7D   : 0x1 -> 0x0
0x2FA  : 0x1 -> 0x0
0xC64  : 0x1 -> 0x0
0x14C4 : 0x2 -> 0x0
0x15AC : 0x1 -> 0x2
0x15B1 : 0x0 -> 0x1
0x1651 : 0x1 -> 0x0

```

Listing 4.5: Disabling IOMMU and DMA Protections in IntelSetup

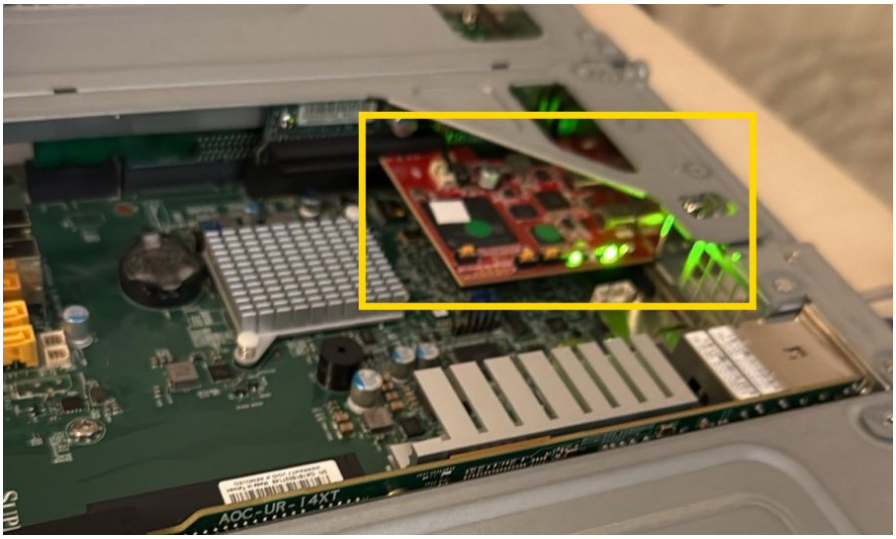

Figure 4.9: PCI Squirrel Connected Inside Chassis

4.2.8.2 Solution Advice

X41 recommends physically hardening all internal PCIe interfaces to prevent unauthorized device insertion. Additionally, software-level checks should be implemented to verify that IOMMU protections are properly enabled before loading any critical components, such as WireGuard keys.

4.3 Informational Notes

The following observations do not have a direct security impact, but are related to security hardening, affect functionality, or other topics that are not directly related to security. X41 recommends to mitigate these issues as well, because they often become exploitable in the future. Doing so will strengthen the security of the system and is recommended for defense in depth.

4.3.1 AZR-PT-25-100: Return Value Unchecked

Component: azire-wireguard/drivers/net/wireguard/receive.c:wg_receive_handshake_packet()

4.3.1.1 Description

The return value of the allocation of an instance of `struct wg_unknown_peer_net_meta` is not checked against possible *NULL* values. This may lead to a kernel error when the value is dereferenced in the following code, as shown in listing 4.6.

```
1 struct wg_unknown_peer_net_meta *net_meta = kzalloc(sizeof(*net_meta), GFP_KERNEL);
2 ...
```

Listing 4.6: Return Value of Allocation Not Checked Against NULL

The pointer is dereferenced during a function call to ***wg_unknown_peer_get_net_meta_from_skb***. This function also does not implement a *NULL* check.

4.3.1.2 Solution Advice

X41 recommends checking the return value against NULL, possibly logging the error and returning from the function early.

4.3.2 AZR-PT-25-101: Command Outside Chroot Environment

Component:	azire-pxe/wgimagetk.sh:358

4.3.2.1 Description

The command shown in listing 4.7 in the build script `azire-pxe/wgimagetk.sh` is executed on the build host, rather than in the image environment due to a missing invocation of *chroot*.

```
1 apt --quiet --assume-yes purge --auto-remove openipmi
```

Listing 4.7: Openipmi Package Removed on Build Host

As a result unneeded packages may be installed on the production image and the image is larger than necessary.

4.3.2.2 Solution Advice

X41 recommends invoking the command as a parameter to *chroot*.

4.3.3 AZR-PT-25-102: Changing Email to Unverified Address Has No Consequences

Component: azire-manager/app/Http/Controllers/Reseller/Manager/AccountController.php

4.3.3.1 Description

A user may change their email address in the account settings. A verification email is sent to the email address and the user is logged out. However, there are no consequences if the email address cannot be verified. The user may still log in and will appear to be verified according to the `verified` column of their respective row in the database's `User` table. As a consequence a User that changed their email address to a non-existing address may lose their account if they forget their password.

4.3.3.2 Solution Advice

X41 recommends resetting the `verified` field of the user model to `false` after changing the email address. Further, X41 recommends returning to the previously verified email address, should the new address not be verified within a given period of time.

4.3.4 AZR-PT-25-103: Use of Debian 12

Component:	audit/sources/azire-pxe/wgimagetk.sh

4.3.4.1 Description

The build script only supports Debian 12 (“bookworm”). Additionally, the documentation states that the build command must be run on Debian 12 (“bookworm”) or Debian 13 (“trixie”).

Debian 12 reaches the end of regular security support on 2026-06-10, after which the Debian security team will no longer provide updates. While the Debian LTS[58] team may provide extended support until 2028, the Debian project recommends[59] not relying solely on LTS for production systems.

The Debian project recommends[60] to only use the latest stable release for production.

4.3.4.2 Solution Advice

X41 recommends updating the build script to the latest stable release, Debian 13 (“trixie”), building images on Debian 13 hosts, and updating the production servers to those images.

[58] Long Term Support
[59] `https://wiki.debian.org/LTS`
[60] `https://wiki.debian.org/DebianReleases`

4.3.5 AZR-PT-25-104: Use of Shell Script to Build Images

Component:	audit/sources/azire-pxe/wgimagetk.sh

4.3.5.1 Description

A Bash script is used to build trusted images to be executed by the VPN servers. The use of shell scripts comes with inherent dangers and requires a lot of caution. It is difficult to verify that a shell script always behaves in the intended way.

For example, the *wgimagetk.sh* script makes use of `set -u` to treat unset variables as errors. While this will prevent execution of a command that contains an unset variable and will treat it as an error, it will not cause the script to exit and prevent further execution, nor prevent the use of accidentally empty variables. Similarly, a failed command substitution will not prevent execution of the command it is used in. It should be noted that the use of `set -o errexit` (or `set -e`) to exit on errors is not recommended as it is disabled entirely as soon as the call stack includes `if`, `&&`, or `||`, among others.

Shell scripts provide no type safety or type signatures, and come with many insecure defaults and caveats[61]. The Linux Documentation Project's Advanced Bash-Scripting Guide advises to not use shell scripts where security is important[62].

4.3.5.2 Solution Advice

X41 recommends replacing the shell script with a programming language that features modern security features.

When using shell scripts, a testing framework such as ShellSpec[63] could be used to detect errors and prevent some of the caveats.

[61] `https://mywiki.wooledge.org/BashPitfalls`
[62] `https://tldp.org/LDP/abs/html/why-shell.html#AEN82`
[63] `https://shellspec.info/`

4.3.6 AZR-PT-25-105: Debian Package Signature Verification Not Enforced

Component: audit/sources/azire-pxe/wgimagetk.sh

4.3.6.1 Description

In the build process for VPN server images, the `debootstrap` command is used to download Debian packages insecurely via HTTP and without enforcing verification against the Debian keyring[64] signatures.

```
1 debootstrap --verbose --cache-dir "$cache_directory" \
2     --components=main,contrib,non-free-firmware \
3     bookworm "$rootfs" http://ftp.se.debian.org/debian
```

Listing 4.8: Debootstrap Used with HTTP

It should be noted that the *README.md* file states that "the build command must be run on Debian 12 Bookworm or Debian 13 Trixie", where the keyring is available as part of the OS[65]. The `debootstrap` command implicitly uses the keyring if available, which is why this issue is filed as an informational note instead of a finding.

When the keyring is not available, an attacker on the network could intercept the connection and modify the packages, resulting in a complete compromise of the produced server image.

The output produced by the above `debootstrap` command issues a warning when no keyring file is available, but continues nonetheless, as shown below.

```
1 I: Target architecture can be executed
2 W: Cannot check Release signature; keyring file not available
  ↪  /usr/share/keyrings/debian-archive-keyring.gpg
3 I: Retrieving InRelease
4 I: Retrieving Packages
```

Listing 4.9: Debootstrap Warning about Missing Keyring File

[64] `https://wiki.debian.org/DebianKeyring`
[65] Operating System

4.3.6.2 Solution Advice

X41 recommends adding the `--force-check-sig`[66] flag (`--force-check-gpg` on Debian 12) to the `debootstrap` command in order to ensure packages are always verified, or the build command is aborted.

Additionally, X41 recommends using HTTPS to download packages as an additional security measure.

[66] `https://manpages.debian.org/trixie/debootstrap/debootstrap.8.en.html#force-check-sig`

4.3.7 AZR-PT-25-106: Immutable Log Unit Could Fail

Component:	audit/sources/azire-pxe/files/immutable-var-log.service

4.3.7.1 Description

The systemd service unit executes the shell command `shopt -s dotglob && rm -rf /var/log/* && chattr +i /var/log` in order to make the log directory immutable.

The critical `chattr` command is only executed if the previous commands were successful, which may fail in rare circumstances such as the system being out of memory or the `/var/log/*` glob expanding to a list that is longer than *`ARG_MAX`*[67].

X41 believes that these circumstances are unlikely to occur and thus files this as an informational note.

The system is likely to continue running despite the service unit failing.

4.3.7.2 Solution Advice

X41 recommends adding instructions that prevent the system from booting up if the unit fails. This may involve using the systemd unit options *`OnFailure=emergency.target`* and *`OnFailureJobMode=replace-irreversibly`*.

Alternatively, or in addition, it could be considered to mount `/var/log` as a read-only *`tmpfs`*.

[67] `https://wiki.debian.org/CommonErrorMessages/ArgumentListTooLong`

4.3.8 AZR-PT-25-107: Build Automation

Component:	audit/sources/azire-pxe

4.3.8.1 Description

Malwarebytes Inc. explained in a discussion with X41 that they currently rely on builds manually created in VMs[68] on developer machines, before eventually releasing them onto the PXE server.

This poses risks as the resulting image is hard to verify, which could be manipulated by malware on any of the authorized developer's machines, a rogue employee, or by an attacker who managed to access the developer machine.

4.3.8.2 Solution Advice

X41 recommends implementing a CI/CD[69] setup with automated builds, where the builds are created on a controlled and trusted system and are solely based on the source code pushed to the relevant repositories.

Reproducible builds[70] can help ensure that an image contains exactly what is contained in the relevant source code repositories.

[68] Virtual Machines
[69] Continuous Integration/Continuous Delivery
[70] `https://reproducible-builds.org/`

4.3.9 AZR-PT-25-108: Signed Commits

Component:	Source Code Repositories

4.3.9.1 Description

Malwarebytes Inc. explained in a discussion with X41 that commits to their source code repository are currently not signed.

This could allow an attacker who gained access to the source repository to introduce malicious changes without being noticed.

4.3.9.2 Solution Advice

X41 recommends signing all commits and tags, and configuring the repository to only accept signed commits/tags with a trusted signature. Developers should configure their local repositories to also verify signatures.

Merge commits or local rebasing can be used to avoid changing other contributor's commits, breaking the signature. Git supports[71] *gpg*, *ssh*, and *x509* signatures. In the meantime, only tags could be signed until the organisational details are decided and implemented.

The build pipeline should verify signatures before building an image.

[71] `https://git-scm.com/docs/gitformat-signature`

4.3.10 AZR-PT-25-109: Reducing Dependencies

Component:	azire-pxe/wgimagetk.sh

4.3.10.1 Description

The build script installs a list of Debian packages. There are no comments indicating what each of them are required for, and some that are no longer required were left included without notice.

Among several obvious network and configuration tools, the dependencies also included *cron*, *nano*, and development and build tools such as *build-essential*, development libraries, header files, and *git*.

4.3.10.2 Solution Advice

X41 recommends re-evaluating each of the dependencies, removing all that are not strictly required. This could become a regular practice. Additionally, a comment could be added to each dependency, describing why it is needed.

To reduce the attack surface, X41 also recommends building all binaries outside of the server, removing the need for development and build tools.

APT[72] should be configured to not automatically install recommended packages.

To take this step even further, it could be considered to build a minimal Linux distribution that comes without any interactive shells and only contains the binaries actually executed, hardening and minimizing the attack surface even further.

[72] Advanced Package Tool

4.3.11 AZR-PT-25-110: Unprotected Instant Payment Notification Endpoint

Component: https://manager.azirevpn.com/ipn/cleverbridge

4.3.11.1 Description

The Instant Payment Notification Endpoint is used by the payment provider Cleverbridge to notify Azire manager of events in the payment process of a user, e.g. when a successful payment was executed or a refund was triggered. These events are used by Azire Manager to conduct the appropriate operations on the database to enable a user to use the service or revoke a subscription. Cleverbridge provides a list of IP addresses, which can be used to establish the authenticity of such a notification. It was found that the source code under review implements appropriate checks and that it is not possible to forge a payment notification unless an attacker is able to spoof the IP address of Cleverbridge. However, a check for establishing the authenticity was apparently lacking on the production system[73] and it was possible to deliver arbitrary payment notifications. If an attacker is able to learn a user's id and a product id, they are able to forge a valid payment notification and create a subscription for the user, effectively receiving the service without valid payment. Further, with a valid *purchaseId*, an attacker would be able to trigger a refund in the Azire Manager, effectively revoking access to the service for the corresponding user despite a valid payment. The command listed in listing 4.10 is able to reproduce the issue. It returns HTTP status code 500 instead of the expected 403. The 500 is returned, because the server logic expects to parse values from the POST request and throws an exception when they do not exist.

```
1 curl -w "%{http_code}\n" -X POST https://manager.azirevpn.com/ipn/cleverbridge -o /dev/null
2 500
```

Listing 4.10: Command to Reproduce Issue

4.3.11.2 Solution Advice

This issue had been fixed in the code under review, but was exploitable in the live production systems at the time. A fix was deployed during the audit.

[73] `https://manager.azirevpn.com`

5 About X41 D-Sec GmbH

X41 D-Sec GmbH is an expert provider for application security and penetration testing services. Having extensive industry experience and expertise in the area of information security, a strong core security team of world-class security experts enables X41 D-Sec GmbH to perform premium security services.

X41 has the following references that show their experience in the field:

- Source code audit of ISC BIND9 DNS server[1]
- Source code audit of the Git source code version control system[2]
- Review of the Mozilla Firefox updater[3]
- X41 Browser Security White Paper[4]
- Review of Cryptographic Protocols (Wire)[5]
- Identification of flaws in Fax Machines[6,7]
- Smartcard Stack Fuzzing[8]

The testers at X41 have extensive experience with penetration testing and red teaming exercises in complex environments. This includes enterprise environments with thousands of users and vendor infrastructures such as the Mozilla Firefox Updater (Balrog).

Fields of expertise in the area of application security encompass security-centered code reviews, binary reverse-engineering and vulnerability-discovery. Custom research and IT security consulting, as well as support services, are the core competencies of X41. The team has a strong technical background and performs security reviews of complex and high-profile applications such as Google Chrome and Microsoft Edge web browsers.

X41 D-Sec GmbH can be reached via `https://x41-dsec.de` or `mailto:info@x41-dsec.de`.

[1] `https://x41-dsec.de/news/security/research/source-code-audit/2024/02/13/bind9-security-audit/`
[2] `https://x41-dsec.de/security/research/news/2023/01/17/git-security-audit-ostif/`
[3] `https://blog.mozilla.org/security/2018/10/09/trusting-the-delivery-of-firefox-updates/`
[4] `https://browser-security.x41-dsec.de/X41-Browser-Security-White-Paper.pdf`
[5] `https://www.x41-dsec.de/reports/Kudelski-X41-Wire-Report-phase1-20170208.pdf`
[6] `https://www.x41-dsec.de/lab/blog/fax/`
[7] `https://2018.zeronights.ru/en/reports/zero-fax-given/`
[8] `https://www.x41-dsec.de/lab/blog/smartcards/`

Acronyms