**Synopsis of an Elite Attack**

Includes Attack Tools, How They Were Used and Concepts

In Relation to Attack on Walmart and Personal Devices

***\*Author: Mason Dyess\****

**I have comprised a list of tech, tools, repos, and related items to the attacks made on Walmart and personal devices starting on 01/01/2025 through 01/18/205.**

**Devices hacked:**

**Device 1) Walmart’s MacBook Pro**

* OS: MacOS – flavor unsure (forgot)
* Browsers
  + Chrome (logged into personal profile)
  + Safari
  + Firefox
* Extensions: Forcepoint was added to all browsers by attackers

**Device 2) ASUS i7**

* OS - originally running Windows 10 then hackers ghost imaged Windows 11 ‘
* Browsers used:
  + Chrome (personal profile logged in)
  + Edge

**Device 3) Motorola Android pone – moto 5g 2023**

* OS: Android versión 14
* Browsers
  + Chrome

**Device 4) iPhone 14**

* OS- iOS 18.2.1
* Browsers
  + Safari
  + Chrome

**GAINING ACCESS**

Remote Access Tools (RATs)

Installed via Chrome, Wifi, & especially Bluetooth Attacks

**------Taking over nearby Bluetooth devices using Blue Born ----**

BlueBorn Bluetooth attack:

* Repo: <https://github.com/ArmisSecurity/blueborne>
* Note: An MIT repo (common denominator in tools used)

Please watch this short BlueBorn example video for a 1 min explaination:

* <https://youtu.be/Az-l90RCns8>

**--------------- Remote Access Tools --------------------------------**

**One potential is Volt Typhoon :**

* **Remote Access Tech - suspected**
* **Citation**:
  + <https://techcrunch.com/2025/01/14/doj-confirms-fbi-operation-that-mass-deleted-chinese-malware-from-thousands-of-us-computers/>
* **Excerpt from above:**
  + Typhoon joins the growing list of Typhoon-monikered Chinese state-sponsored hacking groups. This list includes Volt Typhoon, a group of Chinese government hackers tasked with setting the stage for destructive cyberattacks, and Salt Typhoon, the China-backed group responsible for [the mass hacking of U.S. phone and internet companies](https://techcrunch.com/2024/10/07/the-30-year-old-internet-backdoor-law-that-came-back-to-bite/).
* Proofpoint investigates specific threats to Transportation companies
  + [Threat Actors Target Transportation and Logistics Companies in North America - Security MEA](https://securitymea.com/2024/09/30/threat-actors-target-transportation-and-logistics-companies-in-north-america/)

**Fingerprint and Tensor :**

* **Repo**:
  + [Manjushanair/fingerflow-fingerprints: FingerFlow is an end-to-end deep learning Python framework for fingerprint minutiae manipulation built on top of Keras - TensorFlow high-level API.](https://github.com/Manjushanair/fingerflow-fingerprints)
* **About:** 
  + FingerFlow is an end-to-end deep learning Python framework for fingerprint minutiae manipulation built on top of Keras - TensorFlow high-level API.
* Note: MIT Repo

**Walmart Attack Post Entry Tools**

**------------------------------ gh-ost for DBs and structure logging attacks---------------------**

While gh-ost may normally be used with traditional DBs, this tech and the concept behind it can be used with structured logs to takeover, record, reverse, and mirror OS state when exploited/misconfigured. Further, the inclusion of RSS tech and concepts in addition to this are explained below:

**gh-ost used for DB hacking:**

* Repo:
  + <https://github.com/github/gh-ost/tree/master>
* Note:
  + <https://github.com/github/gh-ost/blob/master/doc/cheatsheet.md>
  + Note tensor on link above
  + Note: MIT repo

**How this tool can be used to manipulate an OS**

* Binary log streams are primarily associated with databases and are not typically used for CPUs directly. However, the concept of logging changes and events can be applied to various systems, including operating systems (OS).
* Applying a database model concept to an OS could involve creating structured logs to track system events, changes, and operations. This could help in monitoring and analyzing system behavior, detecting anomalies, and improving security.

**Structured Logging Injection**

* **Log Injection Attack:** This attack involves manipulating input data to insert malicious content into log files. For example, an attacker might craft input containing special characters or commands that are logged by a web application 1. This can distort log entries, mask the attacker's activities, or even inject malicious code that could be executed by systems relying on those logs for automated responses
* **Citation**: <https://www.aptive.co.uk/blog/log-injection-attack/>

**Including RSS with gh-ost + log injection (Walmart attack)**

* **Question to AI:** can log injection attacks utilize RSS exploits for faster streaming and reversal
* **Log injection attacks can indeed be combined with other exploits, such as RSS exploits**, to enhance their effectiveness. Here's how this might work:
* **Log Injection Attack**: An attacker injects malicious content into log files, which can distort log entries or inject executable code.
* **RSS Exploit:** RSS (Really Simple Syndication) feeds are used to publish frequently updated information. If an application processes RSS feeds and logs their content, an attacker could exploit this by injecting malicious content into the RSS feed2.
* **Faster Streaming and Reversal**: By combining log injection with RSS exploits, an attacker can quickly propagate malicious content across multiple system**s** that consume the RSS feed 2. This can lead to faster dissemination of the attack and make it harder to detect and reverse.
* **This combination can be particularly dangerous because it leverages the automated nature of RSS feeds and log processing, allowing the attack to spread rapidly and potentially evade detection**
* \*\*Injection attacks are perpetually on the OWASp top 10 vulnerabilities
  + <https://www.barracuda.com/glossary/owasp-top-10?utm_source=50364&utm_medium=blog&utm_campaign=blog> While SQL injection, command injection, and cross-site scripting (XSS) attacks are common forms of injection attacks that typically come to mind first, log injection can still present a risk and may often be overlooked.

**Citation**: <https://www.aptive.co.uk/blog/log-injection-attack/>

**How would one go about making a log structure or schema?**

* One answer may be the use of USD (extension .usda) files such as those found on Walmart computer:
  + Universal Scene Description: Creating New Schema Classes with usdGenSchema
    - <https://openusd.org/release/api/_usd__page__generating_schemas.html>
  + USD Documentation : Generating New Schema Classes
    - <https://openusd.org/docs/Generating-New-Schema-Classes.html>

**------- gaining info from binary files as witnessed during Walmart Attack ------------------**

Linux/MacOS commands and how they are used with /usr/bin files to gain information and manipulate files for attack assistance.

Breakdown of how to use commands like file and fileprocctl to gather more information from a binary file. This can help you understand the file's structure and content.

1. file -b <filename>:
   * Use this command to identify the basic file type. For instance:
     + Bash
   * file -b example.bin
     + The output will tell you what type of file it is (e.g., "ELF 64-bit LSB executable," "JPEG image data," etc.).
2. hexdump -C <filename>:
   * This command lets you view the contents of the binary file in a human-readable format (hex and ASCII)
   * Bash
     + hexdump -C example.bin
     + It displays the file content byte by byte, which can help you find patterns or specific data within the file.
3. strings <filename>:
   * Extract readable strings from the binary file. This can reveal embedded text, error messages, or other useful information.
   * Bash
     + strings example.bin
4. objdump -d <filename>:
   * Disassemble the binary file (if it's an executable) to see its assembly code. Useful for reverse engineering.
   * Bash
   * objdump -d example.bin
5. fileprocctl -repair <filename>:
   * If you suspect the binary file is corrupted, you can use this command to attempt to repair it.
   * Bash
   * fileprocctl -repair example.bin

By using these commands, you can gather valuable information about the binary file's structure, content, and potential issues

**PERSONAL ATTACKS**

**(Phones & Other Personal Devices)**

**--------- RTO - listening on devices using radio signals -------------**

Hacking through radio signals - including police:

* <https://rtotech.org/hackrf-sdr-softwares/>

**------------------ A bit on Vectors (attacks & info) -------------------------**

**Note the use of vector attack in blue born links above**

Vector attacks  as seen on Walmart hack (Walmart - search .usda files) and in BlueBorn attack/hack

 info:

* [What are Attack Vectors: Definition & Vulnerabilities | CrowdStrike](https://www.crowdstrike.com/en-us/cybersecurity-101/threat-intelligence/attack-vector/)

**-------------android installed hack (installed via BlueBorn or similar attack )--------**

Android terminal app files from subuser 6 found with ps -ef command and the visiting process directory I find these repos:

1. <https://github.com/mishkago/userbot>
2. <https://github.com/ByCh4n/BCHackTool.git>
3. <https://github.com/guiireal> (author)

Logs from android:

(Portuguese)

cho "1 - ${CYAN}Inscreva-se no canal para mais tutoriais e novidades:${RESET}" echo "👉 ${BOLD}https://youtube.com/@devgui\_${RESET}" echo "" echo "2 - ${CYAN}Leia o tutorial completo em nosso repositório no GitHub:${RESET}" echo "👉 ${BOLD}https://github.com/guiireal/takeshi-bot${RESET}"

(English)

echo "1 - ${CYAN}Subscribe to the channel for more tutorials and news:${RESET}" echo " 👉 ${BOLD}https://youtube.com/@devgui\_${RESET}" echo "" echo "2 - ${CYAN}Read the full tutorial in our GitHub repository:${RESET}" echo " 👉 ${BOLD}https://github.com/guiireal/takeshi-bot${RESET}"

**FOUND ON MY ANDROID PHONE:**

* <https://github.com/guiireal/takeshi-bot>
* <https://guiireal.com/>
  + wrote takeshi bot found on android.

he claims he is a mobile phone hacker and owns Takashi bot linked above

-LOG FROM PHONE BELOW

*\*\*\*\*\*\****\*\*TAKSHII BOT BY GUIIREAL ON ANDRPOID - ANDROIID ITERM  LOGS  ps -ef command then process folder of subuser 6  \*\*\****\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**\*\*

(android log )

**# Códigos de cores ANSI**

RED='\033[0;31m'

GREEN='\033[0;32m'

YELLOW='\033[1;33m'

CYAN='\033[0;36m'

BOLD='\033[1m'

RESET='\033[0m'

echo "${GREEN}============================================${RESET}"

echo "🤖 ${CYAN}${BOLD}BEM-VINDO AO INSTALADOR TAKESHI-BOT${RESET} 🤖"

echo "${GREEN}============================================${RESET}"

echo ""

echo "🚀 ${YELLOW}Aguarde enquanto instalamos os módulos...${RESET}"

echo ""

echo "1 - ${CYAN}Inscreva-se no canal para mais tutoriais e novidades:${RESET}"

echo "👉 ${BOLD}https://youtube.com/@devgui\_${RESET}"

echo ""

echo "2 - ${CYAN}Leia o tutorial completo em nosso repositório no GitHub:${RESET}"

echo "👉 ${BOLD}https://github.com/guiireal/takeshi-bot${RESET}"

echo ""

echo "${GREEN}============================================${RESET}"

sleep 3

ter

- Bash Scripting

\**\*\*\*\*\*\*\*\*\*\*\*\*\*\****\*\*end log  ps -ef** *\*\****\*\****\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**\*\*

*\*\*\*\*\*\*\*\*\*\*\*\*\*\****\*\*\* ByCh4n  tool \*\*\****\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**\*\*

found on my android - repos below

* <https://github.com/ByCh4n/BCHackTool>
* Written by <https://github.com/ByCh4n>

His About Me:

Hello, I am a Security Operations Center Analyst at ING Türkiye.

I have worked and continue to work in the following areas:

- Cyber Security

- Cyber Threat Intelligence

- External Attack Surface Management

- Web Application Security

- Red Team Pentes

* RUNNING THE PROCESSES ps-ef in terminal app then opening the process tools I found on this page and this "termux" is my path with directory (result of pwd command) on Andoid
  + <https://github.com/ByCh4n/BCHackTool/blob/master/bchacktool-termux.sh>

bchacktool uses Laravel and Vite (seen in above articles) :

* <https://github.com/laravel/vite-plugin>
* It has MIT license ]
  + as gh-ost and friends have
* again we see same contributors of code found on apple, windows, android, and iPhone. A big connection exists between the devs.

**\**\*\*\*\*\*\*\*\*\*\*\*\*\*\**\*\* userbot  - malware on android \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

Discovered by iterm application command ps -ef and then going to subuser6 processes file locations and repos:

* Spamming app:
  + <https://github.com/mishkago/userbot/blob/main/main.py#L9>
  + <https://github.com/mishkago/userbot/blob/f35f836ded55018ef10c5fe5a73a936dcf12564e/main.py#L166>

**Relation to these guys and this app:**

* <https://api.spiderx.com.br/app?welcome=1>

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*awesome app - has it all \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

<https://github.com/sindresorhus/awesome#readme>

**keywords**

* Robotics
* Internet of Things
* Electronics - For electronic engineers and hobbyists.
* Bluetooth Beacons

*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\****\*\*Laravel userd in tools above** *\*\****\*\****\*\*\*\*\*\**\*\*

* <https://laravel.com/docs/11.x/middleware>
* <https://laravel.com/api/11.x/Illuminate/Foundation/Events/Dispatchable.html>

**--------------------- hacking SCRAM device for use as a WIFI beacon ------------------**

scram device hacked and used as wifi beacon

There have been instances where **devices, including SCRAM devices, have been repurposed or hacked to function as WiFi beacons**. This typically involves manipulating the device's firmware or hardware to broadcast WiFi signals, often using techniques similar to those employed in projects like ESP32-DIV or esp8266\_beaconSpam. These projects demonstrate how devices can be used to create multiple WiFi access points with custom SSIDs, which can be used for various purposes, including network monitoring and security testing.

* [cifertech/ESP32-DIV: Packet Monitor, WiFi Analyzer, Beacon Spam, Deauth Detector](https://github.com/cifertech/ESP32-DIV)
* [spacehuhn/esp8266\_beaconSpam: Creates up to a thousand WiFi access points with custom SSIDs.](https://github.com/spacehuhn/esp8266_beaconSpam)
* MIT repo again