

## Lab 05 – Peripheral Payloads

**Name:** Olivia Flores

**Course:** IS 3033-ON2

**Date:** 3/10/24

### INTRODUCTION

In this lab, we will be learning about important databases associated with vulnerabilities and report on a recent Ubuntu driver associated vulnerability and a specific vulnerability called OverlayFS. To start off, I want to introduce the Common Vulnerabilities Enumeration (CVE) database, the Common Weakness Enumeration (CWE) database, and the U.S. National Vulnerability Database (NVD).

The CVE database provides a standard and unique method of identifying vulnerabilities (weaknesses in computer hardware/software) that may lead to malicious activity and in this lab, we are going to get familiar on how to conduct CVE searches and in-depth research on specific vulnerabilities. The CWE is a community-developed list of software and hardware weaknesses and vulnerabilities. The CWE system provides a standardized way to categorize and describe common security weaknesses, making it easier for software developers, security professionals, and researchers to understand and address them. The CVE feeds the U.S. National Vulnerability Database, this is a very important resource as it provide more information on remediation and impact, including the CVSS score and any CWE references which we will also be practicing in this lab. The NVD allows us to narrow down searches as well by OS, vendor, product, version, vulnerability type, severity, impact, etc. [1]

[1] Mitra, Department of Information Systems and Cyber Security. [Online] “Lab 05 – Peripheral Payloads” UTSA, 2024 [Accessed: March 11, 2024.]

### BREAKPOINT 1

Step 01: Investigate a Recent CVE

To start off looking at CVEs, I chose to check out [CVE-2023-4762](#), the **Google Chromium V8 Type Confusion Vulnerability**, which has a priority level of high, with a CVSS score of 8.8. The Google Chromium V8 Type Confusion Vulnerability refers to a security flaw found in the V8 JavaScript engine, which is used in Google Chrome and other Chromium-based web browsers. Type confusion vulnerabilities happen when the JavaScript engine incorrectly interprets the data type of an object, leading to potential memory corruption and the execution of arbitrary code. To break it down a bit better for us, V8 JavaScript engine is responsible for interpreting and executing JavaScript code within web pages, type confusion vulnerabilities like this one can be particularly dangerous [2]. An attacker could exploit this type of vulnerability by crafting a specific JavaScript code that triggers the type confusion error when executed in the browser. This could potentially allow the attacker to execute arbitrary code within the browser's context, leading to various security risks such as Remote Code Execution (RCE), Information disclosure, and Denial of service to name a few. RCE allows attackers to exploit the vulnerability and execute arbitrary code on a victim's system, potentially leading to unauthorized access, data theft, or other malicious activities. Type confusion vulnerabilities can be leveraged to leak sensitive information from the victim's system or manipulate the browser's memory to disclose sensitive data stored within the browser or other applications running on the system. A denial of service involves the attacker exploiting the vulnerability to crash the browser or cause it to become

unresponsive, leading to a denial-of-service condition for legitimate users attempting to access the affected web page or application. Nonetheless, Google and the V8 development team release security updates and patches to address vulnerabilities like the V8 type confusion vulnerability. It is strongly encouraged to keep our web browsers up to date with the latest security patches to mitigate the risk of exploitation [3].

[2] *What Is Chrome V8?* | Cloudflare, [www.cloudflare.com/learning/serverless/glossary/what-is-chrome-v8/](https://www.cloudflare.com/learning/serverless/glossary/what-is-chrome-v8/). [Accessed March 16, 2024]

[3] "CVE-2023-4762 Type Confusion in V8 in Google Chrome." *WNEsecurity*, 7 Feb. 2024, [wnesecurity.com/cve-2023-4762-type-confusion-in-v8-in-google-chrome/](https://wnesecurity.com/cve-2023-4762-type-confusion-in-v8-in-google-chrome/). [Accessed March 16, 2024]

As I researched more on this vulnerability, I found that it was reported by an anonymous researcher through Google Git on 8-16-2023 as stated in the Chrome releases Google blog. I understand that there are still security updates being conducted and details regarding the bug fixes or software updates are being kept restricted until most users receive the update [4]. In addition, NIST also suggests a required action to apply mitigations per vendor instructions or to discontinue use of the product if mitigations are unavailable so from this information, I can make a conclusion that this vulnerability is still being researched [5]. Referring back to the google blog, I can see that there are historical CVEs that precede this one. One of those CVEs is known as **CVE-2023-4763: Use after free in Networks**, also reported by anonymous on 8-3-2023 and **CVE-2023-4764: Incorrect security UI in BFCache**, which was reported by Irvan Kurniawan (sourc7) on 5-20-2023. From looking at these CVEs, I made an inference that they were also related issues found in Chromium, which lead to this vulnerability, potentially leading to execution of arbitrary code, denial of service or information disclosure [4].

[4] "Stable Channel Update for Desktop." Google Chrome Releases Blog, Google, 14 Sept. 2023, <https://chromereleases.googleblog.com/2023/09/stable-channel-update-for-desktop.html> [Accessed March 16, 2024]

[5] National Institute of Standards and Technology. "CVE-2023-4762." NIST National Vulnerability Database, September 5, 2023, <https://nvd.nist.gov/vuln/detail/CVE-2023-4762> [Accessed March 16, 2024]

I wanted to learn more about execution chains and how they are used to gain access to system information so I went ahead and asked ChatGPT the following question: **"what is the execution chain to gain access to system information?"** This helped me break down the vulnerability that I introduced in this step in a more understanding way, and I am able to see why the severity of this vulnerability is rated as High.

ChatGPT answered, "The execution chain to gain access to system information can vary depending on the specific vulnerabilities and attack vectors involved. However, here's a generalized example of how an attacker might execute such an attack:

1. **Initial Reconnaissance:** The attacker conducts reconnaissance to gather information about the target system, such as its operating system, network architecture, software versions, and potential vulnerabilities. This information helps the attacker identify potential entry points and weaknesses in the system.

2. **Identifying Vulnerabilities:** Using the information gathered during reconnaissance, the attacker identifies specific vulnerabilities in the target system. These vulnerabilities could include software bugs, misconfigurations, weak passwords, or insecure network services.
3. **Exploiting Vulnerabilities:** The attacker exploits the identified vulnerabilities to gain initial access to the target system. This could involve using techniques such as remote code execution, SQL injection, buffer overflows, or exploiting weak authentication mechanisms.
4. **Privilege Escalation:** Once inside the system, the attacker attempts to escalate their privileges to gain higher levels of access than initially obtained. This could involve exploiting additional vulnerabilities or misconfigurations to gain root/administrator access.
5. **Information Gathering:** With elevated privileges, the attacker collects system information such as user credentials, network configurations, software configurations, and other sensitive data. This information can be used to further escalate privileges, perform lateral movement within the network, or carry out other malicious activities.
6. **Exfiltration:** The attacker exfiltrates the stolen system information to a remote server or storage location controlled by the attacker. This could involve transferring sensitive files, credentials, or other data over the network using various exfiltration techniques, such as encrypted channels or hidden communication channels.
7. **Covering Tracks:** To avoid detection, the attacker attempts to cover their tracks by deleting logs, modifying timestamps, and removing any evidence of their presence on the system. This helps to minimize the likelihood of detection and attribution.

It's important to note that this is a high-level overview, and the specific techniques and tools used in each step can vary widely depending on factors such as the attacker's skill level, the target system's configuration, and the overall security posture of the organization. Defending against such attacks requires a multi-layered approach, including regular patching, network segmentation, access control measures, intrusion detection systems, and security awareness training for users.”[6]

[6] “what is the execution chain to gain access to system information?” *ChatGPT, GPT-3.5 January 2022 version, OpenAI*, <https://chat.openai.com/chat>. [Accessed: March 16, 2024.]

The attacker can use reconnaissance to gather information on things like software versions, operating systems, and other system configurations, that can lead to potential weaknesses in the system. Then the attacker can use this information to identify and exploit these vulnerabilities. The exploitation technique used in this vulnerability looks like it involves Remote Code execution, leading the attacker to have unauthorized access. This then gives them the ability to grant themselves escalated privileges, leading to additional vulnerabilities and misconfigurations to gain root access to the system. Having those types of privileges, the attacker can be very dangerous, where they can gather information, exfiltrate the stolen system, then cover their tracks by deleting logs making it unknown to others that there was any type of attack done.

From my understanding, the vulnerability I am introducing involves a specific software version that allows a remote attacker to execute arbitrary code through a crafted HTML page. To mitigate a vulnerability of this type, security professionals can implement measures to reduce the risk of

exploitation and prevent unauthorized execution of arbitrary code. On the National Vulnerability Database, there are a few resources available for this specific vulnerability, one of which is the Debian Security Advisory, letting us know that these problems have been fixed in version 116.0.5845.180-1~deb12u1. It is recommended to users to upgrade chromium packages for security as well [7].

From this information and other resources, I can gather that potential mitigations that can be done are patches or updates by the software vendor, who in this case is Chromium. Since this vulnerability is also still being researched, the release of information is still pending and currently no workarounds have been discovered. On the other hand, there were vendors like Microsoft that conducted a release for a Security Update to Microsoft Edge because it is Chromium-based, which now is no longer vulnerable due to the update. Another mitigation for this vulnerability could be web content filtering which blocks access to malicious sites that may host crafted HTML pages designed to exploit the vulnerability.

This vulnerability could apply to me and many other people, when using chrome books, google chrome or any chromium open-source projects, as we continuously browse the web, use google email, and download files regularly. I did not make this a priority before, but I learned that it is important for us to keep our browser up to date with the latest security patches and updates that Google announces. These updates include fixes for vulnerabilities such as this V8 JavaScript engine [4].

[7] [Security] [DSA 5491-1] Chromium Security Update, [lists.debian.org/debian-security-announce/2023/msg00183.html](https://lists.debian.org/debian-security-announce/2023/msg00183.html). [Accessed March 16, 2024.]

## BREAKPOINT 2

### Step 02: Report on OverlayFS

Now, let's move on and talk about the OverlayFS vulnerability. The vulnerabilities, **CVE-2023-2640 & CVE-2023-32629** were both found in the Ubuntu kernel's OverlayFS module, this is known as the **Ubuntu OverlayFS vulnerability**. The CVE-2023-2640, having a priority and CVSS score of 7.8 (high), affects specific Ubuntu kernels, allowing unauthorized users to assign privileged extended attributes to files on mounted drives without proper security checks. Essentially, this means an unauthorized user could grant special permissions to files without being restricted by security measures, leading to unauthorized access or unspecified risks. On the other hand, the CVE-2023-32629, having a priority and CVSS score of 7.8 (high), is a local privilege escalation vulnerability that skips permissions checks when performing certain operations on Ubuntu kernels. Both are vulnerabilities leading to an attacker gaining full control of an affected system [8]. Ubuntu fixed the vulnerabilities on July 24th, 2023, but it is recommended that users update their kernels to the latest version. These CVEs generally stem from the previously disclosed Linux kernel vulnerability from 2020, CVE-2021-3493, whose exploits work for the discovered vulnerability without any changes [9].

[8] 0xRave. "Ubuntu GameOver(Lay) Local Privilege Escalation CVE-2023-32629 and CVE-2023-2640." Medium, 31 Oct. 2023, <https://medium.com/@0xrave/ubuntu-gameover-lay-local-privilegeescalation-cve-2023-32629-and-cve-2023-2640-7830f9ef204a>. [Accessed March 17, 2024.]

[9] "GameOverlay Vulnerability Impacts 40% of Ubuntu Workloads | Wiz Blog." Wiz.io, 27 July 2023, <https://www.wiz.io/blog/ubuntu-overlayfs-vulnerability>. [Accessed March 17, 2024.]

Stonejajia, Shir Tamari and Sagi Tzadik found that the OverlayFS implementation in the Ubuntu Linux kernel did not properly perform permission checks in certain situations. A local attacker could possibly use this to gain elevated privileges [10]. (**CVE-2023-2640 & CVE-2023-32629**)

[10] *USN-6250-1: Linux Kernel Vulnerabilities | Ubuntu Security Notices ...*, [ubuntu.com/security/notices/USN-6250-1](https://ubuntu.com/security/notices/USN-6250-1). [Accessed March 17, 2024.]

As mentioned in the previous step of this lab, there are several steps to gain access to system information. As I refer to the OverlayFS, there have been multiple vulnerabilities, many which have been similar to one another. There are flaws discovered in Ubuntu's modifications such as allowing the creation of specialized executables, which then grants the ability to escalate privileges to root on the affected machine. Normally, only the root user can have special permissions while running programs, but it has been discovered that it's possible to create an executable file with "scope" file permissions and trick the Ubuntu system into giving these permissions to anyone who runs/executes it [8]. There are several mitigations for this vulnerability displayed through resources on the NVD website, one of them being the Wix blog. To exploit these vulnerabilities mentioned above, attackers need to set up a user namespace and an OverlayFS mount. This means they would already need access to run code on the affected system, making it unlikely for attacks to happen remotely. Ubuntu has released updates to fix these vulnerabilities in affected kernels and it's important for affected Ubuntu users to update to these new versions. If this is not possible, another option is to limit user namespace usage to users with lower privileges to prevent potential exploitation [9].

## BREAKPOINT 3

### Step 03: Check Your Own Ubuntu System for OverlayFS

In this step of the lab, I will take you along with me to test the OverlayFS vulnerability on my own Ubuntu system. First, I started by checking which Ubuntu version I am on.

I was able to do this using the command: **`cat /etc/os-release`**

Down below you can see the Ubuntu version I am on is 22.04.3 LTS (Jammy Jellyfish). This Ubuntu version may be vulnerable depending on the kernel version being used, which according to one of the resources introduced earlier, 15.19.0 and 6.2.0 kernel versions are shown to be vulnerable due to the code change made only to Ubuntu's OverlayFS implementation. This then introduced a second vulnerable path with the same underlying issue because they were Ubuntu specified changes and resulted in flaws not being present in the upstream Linux Kernel or other Linux Distributions [11].

[11] KL, Arun. "How to Fix Gameover(Lay)- Two Local Privilege Escalation Vulnerabilities in Ubuntu Linux Kernel?" *The Sec Master*, 22 Oct. 2023, [thesecmaster.com/how-to-fix-gameoverlay-two-local-privilege-escalation-vulnerabilities-in-ubuntu-linux-kernel/](https://thesecmaster.com/how-to-fix-gameoverlay-two-local-privilege-escalation-vulnerabilities-in-ubuntu-linux-kernel/). [Accessed March 17, 2024.]

```

Last login: Sat Mar 16 19:11:04 UTC 2024 on tty1
olivia-flores@olivia-flores:~$ cat /etc/os-release
PRETTY_NAME="Ubuntu 22.04.3 LTS"
NAME="Ubuntu"
VERSION_ID="22.04"
VERSION="22.04.3 LTS (Jammy Jellyfish)"
VERSION_CODENAME=jammy
ID=ubuntu
ID_LIKE=debian
HOME_URL="https://www.ubuntu.com/"
SUPPORT_URL="https://help.ubuntu.com/"
BUG_REPORT_URL="https://bugs.launchpad.net/ubuntu/"
PRIVACY_POLICY_URL="https://www.ubuntu.com/legal/terms-and-policies/privacy-policy"
UBUNTU_CODENAME=jammy
olivia-flores@olivia-flores:~$

```

Figure 1: Checking which Ubuntu version I am using.

Next, to check the kernel version that I am using, I used the command: `uname -a` for a detailed description of the kernel version and `uname -r` for only the kernel version. As you can see from the figure below, I am on kernel version 5.15.0-91-generic kernel version for Ubuntu 22.04 LTS. According to research done through the Wiz blog, I have determined that this kernel version was not vulnerable or impacted [9].

```

olivia-flores@olivia-flores:~$ uname -a
Linux olivia-flores 5.15.0-91-generic #101-Ubuntu SMP Tue Nov 14 13:30:08 UTC 2023 x86_64 x86_64 x86_64 GNU/Linux
olivia-flores@olivia-flores:~$ uname -r
5.15.0-91-generic
olivia-flores@olivia-flores:~$

```

Figure 2: Checking which kernel version of Ubuntu I am using.

Now, I want to check if OverlayFS is available on my Ubuntu machine. I went ahead and followed the steps provided in the KL site. I started this by downloading the exploit from the git using the command: `$ wget https://github.com/briskets/CVE-2021-3493/blob/main/exploit.c`

After that was done, I wanted to give executable permissions to the exploit.c source code, so I ran the command `$ sudo chmod +x exploit.c`

Next, I ran the `gcc exploit.c -o exploit` command to compile the C source code file shown in figure 3, showing only part of the output. Directly after this I attempted to run the compiled file, known as `./exploit` but I was unsuccessful as displayed in figure 4, this meaning that I was unable to run OverlayFS and accessing root.

```

olivia-flores@olivia-flores:~$ gcc exploit.c -o exploit
olivia-flores@olivia-flores:~$

```

Figure 3: Partial compiled C source code.

```
olivia-flores@olivia-flores:~$ ./exploit
-bash: ./exploit: No such file or directory
olivia-flores@olivia-flores:~$ _
```

Figure 4: Failure attempting to execute source code file.

## BREAKPOINT 4

### Step 04: Research Another Ubuntu-Related CVE

Now to dive into **Ubuntu Jammy Jellyfish** and its vulnerabilities, I used the Canonical Ubuntu search engine to look for CVEs related to this version (22.04.3 LTS) of Ubuntu. I narrowed down the search by looking for High priority CVEs and a Status of Released. I chose to look into CVE-2024-0646, which was published on 17 Jan 24. This vulnerability involved an out-of-bounds memory write flaw in the Linux kernel's Transport Layer Security (TLS) functionality. This happens when a user calls the splice function with a ktls socket as the destination. This vulnerability then allows a local user to either crash the system or potentially escalate their privileges on the system. This vulnerability has a CVSS score of 7.8 with a High priority due to the reasons mentioned above, but you will be able to see that under the Ubuntu package "Status" that they released Linux kernel version 5.15.0-97.107 (shown in green) which addressed this CVE [12]. If using this kernel version or later, you should be good to go. I, on the other hand, need to update my system immediately.

[12] "CVE-2024-0646." *Ubuntu*, [ubuntu.com/security/CVE-2024-0646](https://ubuntu.com/security/CVE-2024-0646). [Accessed March 17, 2024.]

I went back to the National Vulnerability Database and researched the CVE-2024-0646 to look into it deeper. I was able to find the git commit showing more information on this vulnerability, here you can see that this vulnerability was Reported-by: Jann Horn [jannh@google.com](mailto:jannh@google.com) [13]. To access system information with this vulnerability in place, local users must first understand that there is an out-of-bounds memory write flaw that I introduced earlier in this step, initializing the execution chain. These local users with basic access could exploit this flaw to gain administrative privileges on the system, leading to a compromised system, and potentially a system disruption, for example cause a denial-of-service attack. To avoid such attacks, there are mitigations set in place and many resources available to help avoid this vulnerability. Some recommendations by security professionals include updating your system immediately to kernel version 6.7-rc5 or later, where the fix for CVE-2024-0646 is included and monitoring the system regularly also would help prevent this from happening [14].

[13] *NET: TLS, Update Curr on Splice as Well - Kernel/Git/Torvalds/Linux.Git - Linux Kernel Source Tree*, [git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=c5a595000e2677e865a39f249c056bc05d6e55fd](https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=c5a595000e2677e865a39f249c056bc05d6e55fd). [Accessed March 17, 2024.]

[14] "Linux Kernel Flaw (CVE-2024-0646) Exposes Systems to Privilege Escalation - Op Innovate." *OP Innovate - Premium Application Penetration Testing and Incident Response*, 20 Feb. 2024, [op-c.net/blog/linux-kernel-flaw-cve-2024-0646-exposes-systems-to-privilege-escalation/](https://op-innovate.com.net/blog/linux-kernel-flaw-cve-2024-0646-exposes-systems-to-privilege-escalation/). [Accessed March 17, 2024.]

## BREAKPOINT 5

### Step 05: Map an OWASP Mobile Top Ten with a CWE

In this step of the lab, I will go over one of the OWASP Mobile Top Ten releases for 2024. This list includes the most critical web application security risks selected this year and has been carefully reconstructed since the last release in 2016. I am going to introduce what the “M10: Insufficient Cryptography” vulnerability is and then we will go ahead and map it to a CWE.

**M10: Insufficient Cryptography** refers to the vulnerabilities taking place from the improper use or implementation of cryptographic functions. Moreover, insufficient cryptography in a mobile application introduces security weaknesses that can take over the effectiveness of cryptographic measures and compromise the confidentiality and integrity of sensitive data. To name a few of these weaknesses, this may include weak encryption algorithms, improper handling of encryption keys, or flawed implementation of cryptographic protocols, all of which can be easily overlooked [15]. As a system administrator, it is important to remember proper cryptographic techniques and to constantly use best practices because they are crucial for protecting data confidentiality and integrity through web applications.

Now, let’s go over a CWE related to this vulnerability. **CWE-327** is related to this vulnerability in that it uses a broken or risky cryptographic algorithm or protocol, essentially an insufficient cryptography. Just like the vulnerability introduced by OWASP, this CWE involves weak cryptographic algorithms and can lead to potential exposure to sensitive information, unauthorized data modification or spoofing. Although cryptographic algorithms were once thought of to be strong, this can change when cryptography lives in an ever-changing environment, it is common for them to be considered unsafe due to new attacks and increased computing power. Managing weaknesses in hardware-based cryptography is even more challenging now, as flaws may require product recalls and adversaries computing power increases over time [16].

To mitigate insecure cryptography, developers can follow some best practices such as the cryptographic standards published by NIST, avoiding custom algorithms, utilizing trusted library implementations of cryptographic code, and constantly educating ourselves on configuring cryptographic parameters [17]. On the other hand, some mitigations users can keep in mind is making sure to encrypt all sensitive data at rest and in transit, storing passwords with strong adaptive and salted hashing functions such as Argon2, and verify configuration and setting independently, to state the least of the mitigations that should be practiced [18].

[15] “M10: Insufficient Cryptography.” *M10: Insufficient Cryptography | OWASP Foundation*, [owasp.org/www-project-mobile-top-10/2023-risks/m10-insufficient-cryptography.html](https://owasp.org/www-project-mobile-top-10/2023-risks/m10-insufficient-cryptography.html). [Accessed March 18, 2024.]

[16] “Common Weakness Enumeration.” *CWE*, [cwe.mitre.org/data/definitions/327.html](https://cwe.mitre.org/data/definitions/327.html). [Accessed March 18, 2024.]

[17] Craig, Sam. “Exploring CWE-327: Use of a Broken or Risky Cryptographic Algorithm.” *Ubiq*, 8 Sept. 2022, [www.ubiqsecurity.com/exploring-cwe-327-use-of-a-broken-or-risky-cryptographic-algorithm%ef%bf%bc/](https://www.ubiqsecurity.com/exploring-cwe-327-use-of-a-broken-or-risky-cryptographic-algorithm%ef%bf%bc/). [Accessed March 18, 2024.]

[18] “A02:2021 – Cryptographic Failures.” *A02 Cryptographic Failures - OWASP Top 10:2021*, [owasp.org/Top10/A02\\_2021-Cryptographic\\_Failures/](https://owasp.org/Top10/A02_2021-Cryptographic_Failures/). [Accessed March 18, 2024.]



## **CONCLUSION**

In conclusion to this lab, I was able to get more familiar using the CVE database, CWE database, and the National Vulnerability Database, all which are very helpful resources as cybersecurity is an everchanging environment and new attacks and vulnerabilities are discovered constantly. Some challenges I faced were trying to understand the OverlayFS vulnerability and the resources I was referring to were different than my experience, but I came to a conclusion that the Ubuntu and kernel version I was using was not vulnerable to OverlayFS. Later in the lab, I was able to discover that the Ubuntu kernel version I was running is vulnerable after all, but it was involving an out-of-bounds memory write flaw in the Linux kernel's Transport Layer Security. As I wrap up the lab, I am now able to apply everything that I learned to my daily security practices and know where I can look to dive deeper into today's most recent vulnerabilities to help keep myself and others aware.

## **COLLABORATION**

In this lab, I made use of the lab instructions provided to us and some generative AI tool interactions as I was attempting to get familiar with certain terminology. I was able to understand how to conduct in-depth research through the CVE database, CWE database and NVD, as well as other resources presented throughout the lab.