Seneca Polytechnic

CTY130 – Ethical Hacking & Vulnerability Testing

Assignment: Final Project – Part A

Group 11

Names: Farhan Ahmed & Ankit Pradhan

Professor: Ferozuddin Hyder

Due Date: December 4th, 2025

Summary

Finding	Finding Name	Affected Resource	Method of Finding	Severity
1	UnrealIRCd 3.2.8.1 Backdoor (RCE)	IRC service on port 6667	Nmap version scan identified vulnerable UnrealIRCd version	Critical
2	VSFTPD 2.3.4 Backdoor (RCE)	FTP service on port 21	Nmap banner detection identified backdoored VSFTPD version	Critical
3	PostgreSQL Default Credentials UDEV Privilege Escalation	PostgreSQL on 5432, local UDEV subsystem	Service enumeration + Metasploit + local enumeration	Critical
4	PHP CGI Argument Injection (Web RCE)	Apache/PHP-CGI on port 80	Nikto + Nmap HTTP enumeration scripts	High
5	Java RMI Registry Remote Code Execution	Java RMI service on port 1099	Nmap version scan + Metasploit RCE module	Critical

Finding 1 — Initial Foothold 1

Finding Name: UnrealIRCd 3.2.8.1 Backdoor Remote Code Execution

Affected Resource

When we were scanning the target during our enumeration phase, we noticed that port 6667 was running an IRC service. After running an Nmap version scan, the service was identified as **UnrealIRCd 3.2.8.1**. This immediately caught our attention because this version is well-known for containing a built-in backdoor that lets anyone run commands on the system with no authentication at all. Since the service was exposed to the network and running an outdated, vulnerable version, it became one of the first things we decided to investigate. The resource affected here was the UnrealIRCd application itself, which was running in an insecure and unpatched state on the target machine.

Method of Finding

We found this vulnerability simply by performing a standard Nmap scan with service and version detection. As soon as Nmap returned the version "UnrealIRCd 3.2.8.1," we already knew something was wrong because this version has been publicly documented for years as containing a serious backdoor. There wasn't any need for brute-forcing or guessing. Just seeing the version number was enough to confirm the vulnerability because there are multiple public write-ups explaining that this exact version contains a remote code execution backdoor. So Nmap gave us the version, and our knowledge of existing documentation made it clear that this service was exploitable.

Description

The reason UnrealIRCd 3.2.8.1 is vulnerable is because someone inserted a hidden backdoor directly into the software's source code before it was released. This means that anyone who connects to the service can send a specially crafted line of text, and the IRC server will treat it as a command and run it on the operating system. The server doesn't ask for a username, password, or any kind of authentication — it just runs the command. In our case, the IRC service was running as root, so once the backdoor executed our payload, we immediately gained full control of the machine. The screenshots we captured show exactly how the Metasploit module triggers the backdoor and opens a remote shell on the target.

Exploitation

To exploit this vulnerability, we used the UnrealIRCd backdoor module in Metasploit. We entered the target's IP address, set our own machine as the listener, and chose a reverse shell payload. After

everything was set, we ran the exploit. Because the backdoor doesn't require any login or permission checks, the server instantly executed our payload and opened a shell back to us. Once we had the shell, we ran commands like whoami and id, which confirmed that we were operating as the root user. This showed that the exploit worked exactly as expected and gave us complete control of the system with almost no effort. The screenshots clearly show the exploit running and the root shell being returned.

Impact

The impact of this vulnerability is extremely serious because exploiting it gives an attacker full control of the system without needing a username or password. Once the backdoor is triggered, the attacker can do basically anything a root user can do, such as changing or deleting important system files, installing their own backdoors, stealing sensitive data, or using the machine as a stepping stone to attack other devices on the network. Since the exploit gives instant root access, the entire system becomes untrustworthy, and both the confidentiality and integrity of the machine are completely compromised.

Likelihood

The likelihood of this vulnerability being exploited is very high. The backdoor doesn't require any login or special conditions, and the tools to exploit it are widely available and easy to use. Even someone doing basic scanning on the network would quickly see that the server is running UnrealIRCd 3.2.8.1, and from there it takes almost no effort to launch the exploit. Because of how obvious and easy it is, this is the type of vulnerability that attackers would identify right away.

Risk

Since the vulnerability leads directly to full system takeover and can be exploited with almost no effort, the overall risk rating for this finding is considered **Critical**. The combination of high impact and high likelihood makes this one of the most dangerous vulnerabilities on the system.

Recommendations

The best solution is to completely remove UnrealIRCd 3.2.8.1 from the system and replace it with a secure and updated version of IRC server software. This version should never be used, as it is known to be backdoored. The service should also not be exposed to the internet unless absolutely required, and if it is needed, it should be restricted behind proper firewall rules and network segmentation. Because the backdoor grants full root access, the system should also be thoroughly checked for signs of compromise, since it is possible that attackers may have used this vulnerability long before our test.

Finding 2 — Initial Foothold 2

Finding Name : VSFTPD 2.3.4 Backdoor Remote Shell

Affected Resource: The second foothold we gained came from the FTP service running on port

21 of the target machine. When we scanned the system with Nmap, it showed that the FTP server was using **VSFTPD version 2.3.4**. This version is widely known for having a built-in backdoor that was added into the source code. Because of that, simply having this service exposed on the network already put the machine at major risk. The vulnerable resource in this case was the VSFTPD software itself, which was running an unsafe version that gives attackers a way to open a root shell remotely.

Method of Finding

We discovered this vulnerability during our Nmap service and version scan. When the Nmap output showed "VSFTPD 2.3.4," we immediately recognized it because this version is infamous for containing a backdoor in its login process. We didn't need to brute-force credentials or test multiple attack methods. Just identifying the version number was enough to confirm that the service was vulnerable, because the presence of version 2.3.4 alone proves that the server can be exploited to gain remote shell access.

Description

VSFTPD 2.3.4 has a hidden backdoor that activates when someone tries to log in with a username that includes the characters ":)". When this happens, the server silently opens a new shell on port 6200 and runs it with **root privileges**. Anyone who connects to that port instantly receives a root shell without needing any password or authentication. Because of how easy it is to trigger and how powerful the result is, this vulnerability is considered **critical**. It provides a direct path from an unauthenticated state to full system takeover, which is why VSFTPD 2.3.4 is one of the most notorious vulnerable services used in penetration-testing labs. The screenshot of our exploit shows this behavior clearly as it opens a shell immediately after the exploit is run.

Exploitation

To exploit the vulnerability, we used Metasploit's vsftpd_234_backdoor module. All we needed to do was set the target IP address and run the exploit, since the backdoor does not require any additional configuration. As soon as we executed the module, VSFTPD automatically triggered the backdoor and created a root shell for us. Our machine connected to that shell, giving us full access. When we ran commands like id and whoami, the output confirmed that we were operating as the **root user**, meaning we had complete control over the system. The exploitation required almost no effort, which shows how dangerous this vulnerability is in a real environment.

Impact

The impact of this vulnerability is extremely severe because it gives an attacker full control of the system without needing any login credentials. Once the backdoor is triggered, an attacker can change system settings, view or steal files, install their own malicious software, or even completely break the operating system. Since the shell that opens runs as root, there is nothing preventing the attacker from doing whatever they want on the machine. This makes the system completely untrustworthy the moment the exploit succeeds.

Likelihood

The likelihood of this vulnerability being exploited is extremely high. VSFTPD 2.3.4 is one of the most well-known backdoored versions of an FTP server, and the exploit for it is included in almost every penetration-testing toolkit, including Metasploit. There are no special requirements, no authentication steps, and no complicated setup. Anyone performing a simple scan would quickly recognize the version and be able to exploit it in seconds.

Risk

Because this vulnerability leads directly to a full root shell and is extremely easy to exploit, the overall risk is considered **Critical**. The combination of total system compromise and extremely high exploitability makes this one of the most dangerous findings.

Recommendations

The only safe approach is to completely remove VSFTPD 2.3.4 from the system and replace it with a secure, up-to-date FTP server. This version should never be used in any real environment because it is intentionally backdoored. The system should also disable anonymous FTP access and make sure FTP is only available to users who need it. Because an attacker could easily gain root access through this version, the machine should be considered potentially compromised and should be inspected or rebuilt to ensure no hidden malware or backdoors remain.

Finding 3 — Privilege Escalation

Finding Name: Privilege Escalation via PostgreSQL Remote Code Execution Followed by UDEV Netlink Local Kernel Exploit

Affected Resource

We found this privilege escalation path by first noticing that the PostgreSQL service on port 5432 was using the default username and password. Because of that, we were able to log in and use it to get a low-privilege shell as the postgres user. Once we were inside the system, we checked the operating system and saw that it was running an old Linux kernel and an outdated version of udev. Both of these are known to be vulnerable to a local privilege escalation exploit. Since we already had access as a normal user through PostgreSQL, the outdated udev service gave us a way to turn that into full root access.

Method of Finding

We first found PostgreSQL during our Nmap scan, which showed that the service was open. We tried using the default credentials “postgres:postgres,” and surprisingly, it worked. After that, we used Metasploit to run commands through PostgreSQL and got a shell on the machine as the postgres user. Once we had that low-privilege access, we started checking the system for escalation options. That’s when we saw that the kernel and udev versions were old enough to be vulnerable to a well-known local exploit. This confirmed that we could use the udev privilege escalation module with the access we already had.

Description

This escalation worked because two weaknesses were present at the same time. First, PostgreSQL was misconfigured with default login credentials, which let us get onto the system with very little effort. Second, the machine was running an outdated udev version that allows a normal user to trick

the system into running code as root. On its own, the PostgreSQL issue only gives limited access, and on its own, the udev issue requires some way to run commands locally. But together, they created a simple path from a low-privilege user all the way to full root control. By chaining them, we were able to go from no access, to postgres user, to full root permissions.

Exploitation

To carry out the privilege escalation, we first used Metasploit to exploit the PostgreSQL service and get a low-privilege shell as the postgres user. Once the module ran, we received a meterpreter session that gave us basic access to the system. After we had that access, we checked the machine and confirmed that it was using an outdated version of udev, which is known to be vulnerable. With that confirmation, we loaded the udev privilege escalation module in Metasploit and pointed it to the same session we already had. When we ran the exploit, Metasploit automatically created and executed the payload on the target system. A few moments later, we received a brand-new meterpreter session, and this time it was running as root, which showed that the privilege escalation worked.

Impact

The impact of this vulnerability is very serious because once an attacker gets any kind of basic access to the system, they can use this exploit to become the root user. That means they can read or change any file, install anything they want, and completely take over the machine. At that point, none of the normal security controls on the system matter anymore. The whole system becomes fully compromised.

Likelihood

The likelihood of this vulnerability being exploited is high because the system is running older software that is already well known to be vulnerable, and the exploit for it is easy to find and use. As long as an attacker has a low-privilege foothold—like the one we gained through PostgreSQL—they can run this exploit without much difficulty.

Risk

Since the exploit is easy to execute and instantly gives full root access, the overall risk level is **Critical**. This combination of high impact and high likelihood makes it one of the most dangerous issues on the system.

Recommendations

To fix this issue, the system needs to be updated to a newer Linux kernel version, and the udev package must be upgraded to a patched release that is not affected by this vulnerability. Keeping the system updated is important to prevent attacks like this in the future. It would also help to disable services that are not needed, since fewer running services mean fewer possible entry points for attackers.

Finding 4 — Web Vulnerability 1

Finding Name: PHP CGI Argument Injection (Remote Code Execution)

Affected Resource

While we were checking the web services running on port 80, we found that the server was using a PHP-CGI setup that wasn't handling input safely. This meant that certain requests could pass arguments straight to the PHP interpreter without being filtered. That specific part of the web server ended up being the vulnerable component we used to get remote code execution.

Method of Finding

We discovered this issue after looking at the results from Nikto and Nmap's web enumeration. Both tools pointed out that the server was running older versions of Apache and PHP and that PHP was available through CGI. Since PHP-CGI has a known argument injection problem when it isn't configured correctly, the scan results made it clear that this server was likely vulnerable. Once we saw those signs, we tested it with Metasploit to confirm it.

Description

This vulnerability happens when PHP is run in CGI mode without properly sanitizing the arguments passed to it. Because of that, attackers can craft a request that forces PHP to execute commands on the system instead of just serving web content. Anything that PHP runs ends up running as the "www-data" user, which is the typical web server user account. Even though this isn't root, it still gives attackers the ability to execute code, upload malicious files, or move further into the system, which makes the vulnerability very serious. Since it directly leads to remote code execution, it is considered high-risk.

Exploitation

To exploit this issue, we used the php_cgi_arg_injection module in Metasploit. We set the target machine's IP address, our own listener information, and the URL path needed for the exploit. When we ran the module, the server processed the malicious request and executed our payload. This gave us a meterpreter session running as the "www-data" user, which confirmed that we were able to run commands remotely on the web server through this vulnerability.

Impact

This vulnerability has a serious impact because it allows an attacker to run their own commands directly on the web server. With access as the "www-data" user, an attacker can change or upload files the web server controls, use the web server as a starting point to explore the rest of the system, or attempt additional privilege escalation attacks. Even though the initial access is not root, it still gives enough control to do real damage or move further inside the machine.

Likelihood

The likelihood of this vulnerability being exploited is high. Older servers commonly have PHP-CGI enabled, and the argument injection flaw is well-known and easy to test for. Since the exploit requires very little setup and works reliably on outdated configurations, an attacker would have no trouble taking advantage of it.

Risk

Because the vulnerability is both easy to exploit and capable of giving an attacker remote code execution on the server, its overall risk is rated as **High to Critical**. It gives enough access to seriously affect the system and can also be used as a stepping stone for further attacks.

Recommendations

To fix this issue, the server should stop using PHP-CGI entirely or restrict it so it can't be accessed directly. Updating PHP to a newer, secure version is also important to prevent this type of attack. Apache should be configured so that PHP is handled in a safer way, and the web server should follow strict least-privilege rules to reduce the damage if anything like this happens again.

Finding 5 — Web Vulnerability 2

Finding Name: Java RMI Remote Code Execution

Affected Resource

We found this vulnerability in the Java RMI service running on port 1099. This service was accepting remote requests without any authentication, which meant anyone could interact with it. Because the RMI service was exposed and not locked down, it became an easy target for remote code execution, making it one of the more serious issues we identified on the machine.

Method of Finding

We discovered it when our Nmap scan showed that port 1099 was running a Java RMI registry. Since older RMI setups are known to be insecure, we tested it using Metasploit's RMI module. The module confirmed that the service didn't have any protections in place and was vulnerable to an exploit that takes advantage of the way RMI handles remote requests.

Description

This vulnerability exists because the RMI service accepts data from remote users without checking whether the data is safe. RMI can receive objects from clients, and in older or insecure setups, those objects can contain code that the server ends up running. An attacker can use this to force the server to run a malicious payload, which leads to full remote code execution. Since this can happen without a login and can instantly compromise the system, this vulnerability is rated as critical.

Exploitation

To exploit this, we used the java_rmi_server module in Metasploit. We entered the target's IP address and set up our listener so we could receive the connection. After running the exploit, the RMI service downloaded and executed the payload we sent it. This gave us a meterpreter session running as root, which confirmed that the vulnerability allowed complete control of the system through a single remote exploit.

Impact

The impact of this vulnerability is extremely serious because it allows an attacker to get full remote root access to the system. Once someone has root control, they can do anything they want, including stealing data, changing or deleting system files, installing their own malware, or even using the compromised machine to attack others on the network. Because the exploit gives complete control with a single successful attack, the overall impact is very high.

Likelihood

The likelihood of this vulnerability being exploited is high. Older systems often run Java RMI without proper security settings, which makes them easy targets. On top of that, tools like Metasploit already

include ready-made exploits for this issue, so attackers don't need advanced skills to take advantage of it.

Risk

Since this vulnerability is easy to exploit and leads directly to full system compromise, the overall risk rating is **Critical**. The combination of high impact and high likelihood makes it one of the most dangerous vulnerabilities on the server.

Recommendations

To fix this issue, the Java RMI service should either be disabled if it's not needed or restricted so it can only be accessed internally. The service should also require authentication instead of accepting anonymous connections. Updating Java to a supported version and using firewall rules to block external access to RMI ports would help prevent this vulnerability from being exploited in the future.