**Ransomware Attack Simulation**

Group 3 – Fnu Abdullah and Ahmed Hussain

**Abstract**

Ransomware remains one of the most disruptive cyber threats today, encrypting critical data and demanding payment for restoration. This project simulates a ransomware attack on a Kali Linux virtual machine, implementing custom encryption tools, infection via USB Rubber Ducky, and multi-layered defenses including monitoring, detection, and mitigation. The action component develops Python-based encryption and decryption GUIs using cryptographic libraries to recursively encrypt a designated directory ("personal\_Fa0337"). Infection is achieved through a precompiled Rubber Ducky payload that downloads and executes setup scripts in the background. Monitoring employs Auditd for logging AppArmor denials and Wazuh for file integrity monitoring (FIM). Detection integrates custom Wazuh rules to identify encryption patterns, mass modifications, and AppArmor blocks, triggering alerts for potential ransomware. Mitigation uses AppArmor profiles to confine Python processes, denying access to protected directories, combined with Wazuh active responses to terminate malicious processes. Backup and recovery strategies involve encrypted off-system backups for data restoration. The project demonstrates successful attack simulation without defenses, followed by effective containment when defenses are enabled, achieving zero false positives in legitimate operations. This work delivers a reproducible, educational framework for studying ransomware behavior and evaluating layered security controls. This work underscores the importance of proactive monitoring and policy enforcement in mitigating ransomware, providing a reproducible lab environment for educational and research purposes. Future extensions could incorporate machine learning for enhanced detection.

**Introduction**

Ransomware attacks have evolved from opportunistic malware to sophisticated, targeted operations that disrupt organizations and critical infrastructure. High-profile incidents, such as the 2021 Colonial Pipeline breach, underscore the real-world consequences: operational downtime, financial loss, and compromised data integrity. The emergence of Ransomware-as-a-Service (RaaS) platforms has further democratized these attacks, enabling threat actors with minimal technical skill to launch effective campaigns.

This project addresses two core objectives – Simulating a realistic ransomware attack in a safe, isolated environment, and implementing and validating multi-layers mitigation techniques with runtime monitoring and detection techniques using open-source technologies.

Conducted entirely within a Kali Linux virtual machine, the simulation targets a structured directory (personal\_Fa0337) containing sensitive files. The attack vector leverages a USB Rubber Ducky to deliver and execute encryption logic without user interaction.

The defense strategy integrates preventive, detective, and responsive controls:

* AppArmor for application confinement
* Wazuh for monitoring and rule-based detection
* Active response for automated containment
* Offline backups for recovery

By combining these mechanisms, the project demonstrates that even simple, well-configured tools can effectively neutralize ransomware threats before damage occurs.

**Related Works**

Research on ransomware has evolved, focusing on tactics, detection methods, and mitigation strategies. Several studies provide insights relevant to this project.

In "The Evolution of Ransomware: Tactics, Techniques, and Mitigation Strategies," the authors trace ransomware's development from mass-distribution to targeted attacks, emphasizing RaaS and double extortion. Key findings include the use of cryptocurrencies for payments and targeting critical infrastructure. Mitigation recommendations stress cyber hygiene, backups, endpoint detection, user training, and multi-layered defenses, aligning with this project's AppArmor and Wazuh integration. [ <https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4881213>]

"A Comprehensive Literature Review on Ransomware Detection Using Deep Learning" reviews deep learning frameworks for early ransomware detection. It highlights the surge in attacks via RaaS and the role of neural networks in analyzing preprocessed data. Techniques like TextCNN, LSTM, and GANs are discussed, achieving high detection rates but requiring robust datasets. This informs potential future ML extensions in our Wazuh rules. [<https://www.sciencedirect.com/science/article/pii/S2772918424000444>]

"Ransomware: Recent Advances, Analysis, Challenges and Future Research Directions" explores prevention via access control, backups, and key management, and detection through ML on APIs, file entropy, and network traffic. Challenges include evasion tactics and limited datasets; future directions suggest federated learning and blockchain. The custom ransomware AESthetic evades antiviruses, mirroring our simulation's need for dynamic defenses. [<https://pmc.ncbi.nlm.nih.gov/articles/PMC8463105/>]

"Ransomware Trends and Mitigation Strategies: A Comprehensive Review" examines attacker methodologies and cybersecurity implications. Trends include sophisticated encryption and extortion; mitigations focus on proactive strategies like backups and awareness, similar to our defense-in-depth approach. [<https://www.researchgate.net/publication/389546640_Ransomware_trends_and_mitigation_strategies_A_comprehensive_review>]

"Ransomware Detection, Avoidance, and Mitigation Scheme: A Review and Future Directions" proposes the DAM framework, classifying static/dynamic/hybrid detection and avoidance continua. A Djvu case study emphasizes early detection and backups, with future directions in AI and automated responses, guiding our Wazuh active-response implementation. [<https://www.mdpi.com/2071-1050/14/1/8>]

These works underscore the need for integrated defenses, validating our simulation's focus on monitoring, detection, and mitigation.

**Approach**

1. **Action Component: Encryption and Decryption**

The ransomware logic is implemented in two Python modules. The first one is the *client-encrypt.py*, which prompts the user for a ransom password via a GUI (Tkinter), derives a cryptographic key using PBKDF2-HMAC, recursively traverses */home/kali/Desktop/personal\_Fa0337* using os.walk(), encrypts each file with Fernet (AES-128 in CBC mode with HMAC) and appends .enc, securely deletes the original using os.remove(), and uploads the encrypted key to Supabase via the store-credentials.ts edge function. The second module is the decrypt component, client-decrypt-gui.py, which verifies the entered password against Supabase (verify-password.ts), and retrieves and decrypts files if the authentication succeeds. All scripts are hosted in the server-type-arch/ directory and executed within an isolated virtual environment.

1. **Infection Component: USB Rubber Ducky Delivery**

The infection mechanism uses a Hak5 Rubber Ducky to automate payload deployment using the inject\_script.txt, which essentially opens the terminal and downloads the setup.sh, executes it, and exits the terminal. The setup.sh file creates a hidden directory and Python virtual environment, installs dependencies – cryptography, requests, and tkinter (using the requirements.txt), executes client-encrypt.py in the background, and logs activity to setup.log. The precompiled inject.bin is put in Github for instant simulation.

1. **Monitoring Component**

System activity is monitored using two complementary tools – Auditd and AppArmor, which sets a watch rule in */etc/audit/rules.d/apparmor.rules* which leads to denied operation being logged to /*var/log/audit/audit.log* with *apparmor="DENIED"*; and, Wazuh File Integrity Monitoring (FIM), which is configured in */var/ossec/etc/ossec.conf* with the directory listed as HTML tags. In this way, all the file events are recorded in /var/ossec/logs/alerts/alerts.log if anything is modified.

1. **Detection Component**

Custom Wazuh rules in */var/ossec/etc/rules/local\_rules.xml* detect ransomware indicators, such as, encryption file pattern, mass file modifications, access control violations, or critical data protection breach. Rules are reloaded with *sudo systemctl restart wazuh-manager.*

1. **Mitigation Component**

A custom profile is defined at */etc/apparmor.d/usr.bin.python3* which prevents any Python process from reading, writing, or executing within protected paths. The rules can later be tightened to prevent other processes from accessing them. An active response mechanism is also implemented, beyond AppArmor, that runs when a high-severity alerts fire to act as a last line of defense if AppArmor or other controls fail. When triggered, the active response locates the offending process, terminates it, quarantines the suspected artifact, and notifies administrators so the incident can be investigated. The script to run is located at */var/ossec/active-response/bin/ransomware-response.sh,* and the ossec configuration is linked to the file with a command (please refer the GitHub for the code).

The Backup and Recovery principles that should be followed for best practices to ensure data can be restored without paying ransom are the 3-2-1 backup rule, where 3 copies of important data should be stored in 2 different storage media, and one of them should be off-site. The target directory could also be implemented to have daily backups using incremental snapshots, via rsync, LVM, or Btrfs, to an encrypted, immutable storage location such as an external USB drive or cloud storage bucket on AWS. Keeping backups disconnected prevents ransomware from encrypting them.

The recovery process if the attack succeeded is to isolate the infected system, review Wazuh and audit logs, re-image and reinstall the OS, and verify AppArmor and Wazuh in enforce mode before reconnecting, restore data from backup, and validate file integrity.

**Results**

The ransomware simulation was evaluated in two distinct scenarios: defenses fully disabled and defenses fully enabled. All experiments were conducted within an isolated Kali Linux virtual machine, targeting the directory /home/kali/Desktop/personal\_Fa0337, which contained a structured set of test files including the project description and lab manuals distributed across three subdirectories.

When defenses were disabled, the attack succeeded completely. Upon insertion of the USB Rubber Ducky, the payload executed silently in the background, downloaded the encryption script, and recursively encrypted every file in the target directory. Encryption was completed in approximately 8 to 12 seconds, The ransom GUI appeared as intended, and decryption was only possible by entering the correct password verified through the Supabase edge function. No system-level interference occurred, and the attack left no opportunity for recovery without the attacker-controlled key.

In contrast, when all defensive measures were active, the attack was stopped before any file could be modified. AppArmor enforced the custom profile on /usr/bin/python3.13 and blocked the first write attempt at the kernel level, resulting in an immediate “Permission denied” error. This denial was logged in /var/log/audit/audit.log with an apparmor="DENIED" entry. Within 3-6 seconds, the attack was blocked. As a result, no file was encrypted, and no ransom demand appeared, and system integrity remained fully intact.

These results clearly demonstrate that a layered defense architecture, combined with preventive access control via AppArmor, and real-time behavioral detection via Wazuh, successfully prevents ransomware execution with zero impact on normal system operation.

**References**

[1] J. A. R. Quintero, "The Evolution of Ransomware: Tactics, Techniques, and Mitigation Strategies," SSRN Electron. J., Jul. 2024. [Online]. Available: <https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4881213>

[2] M. S. Hussain, M. S. Al Reshan, A. S. Alshahrani, A. Adel, H. Agarwal, and A. K. Srivastava, "A comprehensive literature review on ransomware detection using deep learning," Cyber Secur. Appl., vol. 2, p. 100078, 2024. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S2772918424000444>

[3] C. Beaman, N. Barkworth, T. D. Akande, S. Hakak, and M. K. Khan, "Ransomware: Recent advances, analysis, challenges and future research directions," Comput. Secur., vol. 111, p. 102472, Dec. 2021. [Online]. Available: <https://pmc.ncbi.nlm.nih.gov/articles/PMC8463105/>

[4] M. A. Siddiqi and S. A. Alkhaldi, "Ransomware trends and mitigation strategies: A comprehensive review," ResearchGate, Mar. 2025. [Online]. Available: <https://www.researchgate.net/publication/389546640_Ransomware_trends_and_mitigation_strategies_A_comprehensive_review>

[5] S. A. Alqahtani and R. A. Alshehri, "Ransomware Detection, Avoidance, and Mitigation Scheme: A Review and Future Directions," Sustainability, vol. 14, no. 1, p. 8, Dec. 2021. [Online]. Available: <https://www.mdpi.com/2071-1050/14/1/8>