

**Swinburne University of Technology**

***School of Science, Computing, and Engineering Technologies***

**ASSIGNMENT AND PROJECT COVER SHEET**

Unit Code: **COS80013** Unit Title: **Internet Security**

Assignment number and title: **Assignment-2**

**Research Review of Emerging Technologies to defend against Ransomware based Cyber Attacks**

Due date: **02/06/2025**

Lab/tutor group: **Friday – 6:30 pm – 8:30 pm**  Tutor: **Yasas Akurudda Liyanage Don**

Lecturer: **Rory Coulter , Yasa Akurudda Liyanage Don**

Full name: **Arun Ragavendhar Arunachalam Palaniyappan** Id no: **104837257**

I declare that this assignment is my individual work. I have not worked collaboratively nor have I copied from any other student’s work or from any other source except where due acknowledgment is made explicitly in the text, nor has any part been written for me by another person.

Signature: Arun Ragavendhar

Marker's comments:

Total Mark:

# **1. Introduction**

The STARFLEET investigation showed a multi-step cyber-attack starting with phishing and ending in a full ransomware deployment. The attacker stole login credentials, moved laterally, disabled defences, ran memory-only scripts, and cleared logs to avoid detection. To stop such attacks in the future, this review looks at current defence strategies in recent cybersecurity research. It focuses on AI-based ransomware detection, honeypots, multi-factor authentication (MFA), network monitoring, and centralised log analysis through SIEM tools. It also examines the features these systems use—like event IDs, system calls, and user behaviour—and discusses how well they detect or stop advanced threats like those used in the STARFLEET case.

# **2. Current Trends in Detecting and Stopping Similar Attacks**

This research review focuses on four areas: machine learning techniques, honeypots, multi-factor authentication (MFA), and advanced log monitoring with SIEM systems. Each of these, tackle a different stage in the attack chain.

## **2.1 Using Machine Learning to Catch Ransomware**

Machine learning (ML) is used to detect ransomware either by inspecting files or monitoring system behaviour. One approach uses opcode patterns from Portable Executable (PE) files to train XGBoost and decision tree models, achieving 98.3% accuracy on static datasets (Arabo et al., 2023). These models can detect ransomware before it executes, but they require access to the full file and cannot detect malware that runs only in memory.

Behavioural detection models work differently. For example, one approach used Windows event logs (IDs 4688 for process creation, 4657 for registry changes) to train Random Forest and Logistic Regression models, detecting attacks with 97% accuracy (Kumar & Singh, 2022). These features mapped closely to STARFLEET activity, such as disabling Defender via PowerShell. Another study showed that Random Forests and neural networks outperformed SVM and decision trees across dynamic ransomware datasets, especially in early infection stages (Patel et al., 2023). However, these systems need frequent retraining and are sensitive to incomplete or noisy data.

## **2.2 Honeypots to Catch Malware Before It Spreads**

Honeypots are fake systems or files placed to lure attackers and reveal their actions. Zhuravchak et al. (2021) used symbolic links to distribute decoy files. When ransomware encrypted any of them, the action triggered immediate alerts and system lockdown. This method stopped variants like GonnaCry, catching the encryption attempts before they spread further.

Voerman et al. (2020) proposed adaptive honeypots that change their bait based on attacker interactions. These use ML to shift file names or ports dynamically and log attacker behaviour in real time. Dionaea, a low-interaction honeypot, simulates services like SMB and FTP. It captures payloads and logs connection metadata such as IP address, protocol, and payload hash (Sethia & Jeyasekar, 2019). The strength of honeypots lies in early signal generation, but they must be placed in realistic attack paths to be effective. Malware using sandbox evasion techniques may ignore honeypots altogether.

## **2.3 Multi-Factor Authentication (MFA)**

MFA blocks login attempts even if passwords are stolen. AlSaleem & Alshoshan (2021) tested a graphical MFA model where users selected an image sequence, paired with device ID verification. This helped counteract keyloggers and simple phishing, though physical observation could still bypass it. Gadducci et al. (2021) reviewed 10 MFA protocols and showed that SMS and email-based MFA are easy to intercept. They recommended hardware tokens and biometrics for stronger resistance to relay attacks and phishing kits.

However, STARFLEET showed that credential theft is only one step. Even strong MFA can’t prevent post-login activities like privilege escalation or script execution. It must be used alongside deeper detection tools.

## **2.4 Log Monitoring and SIEM Integration**

Modern detection combines logs from across the system and uses machine learning to find suspicious behaviour. Zhang et al. (2024) used audit logs to build process graphs. Their Graph Neural Network flagged unusual parent-child process chains and registry writes linked to known memory-based malware like Kovter.

Singh et al. (2022) enhanced SIEM with IDS alerts and Random Forest models to classify attack types in real time. Features included login attempts, access times, and command sequences. Ali et al. (2024) used web server logs and found features like URL entropy, access time variance, and user-agent anomalies effective for spotting bot traffic and scripted logins. These logs helped detect stealthy brute-force campaigns missed by rule-based filters.

Rahim et al. (2024) applied SIEM techniques in automotive networks by monitoring CAN bus messages. They flagged delayed or malformed control commands, showing how SIEM can be tailored beyond enterprise IT.

## **2.5 Summary of Observed Trends**

Across all studies, the strongest trend is using real-time behaviour for detection. Systems combining static ML with behavioural logs, adaptive honeypots, and SIEM alerting had the best outcomes. For example, a ransomware attempt flagged by symbolic honeypots (Zhuravchak et al., 2021) was automatically responded to via endpoint isolation through SIEM.

Each tool has weaknesses: honeypots can be bypassed, ML models need regular tuning, and SIEMs are only as good as their input data. But layered use—like pairing an image-based MFA with endpoint logging and adaptive file honeypots—makes it harder for attackers to avoid detection. The STARFLEET attacker used multiple steps. Stopping such threats needs defence across those same layers, working in sync.

# **3. Use of Features in Detection and Response**

Security tools rely on specific types of data—called “features”—to detect cyber-attacks. These can come from files, logs, system behaviour, network activity, or user input. In the STARFLEET case, memory-resident PowerShell commands, batch script execution, registry edits, and antivirus tampering were key indicators. This section reviews what kinds of features recent papers used to detect similar behaviours, and what challenges were involved.

## **3.1 File and Code-Level Features**

Some studies focus on data found inside malware files. These include opcode sequences, entropy scores, and byte frequency. Arabo et al. (2023) used a “digital DNA” model to extract opcode patterns from executables and train XGBoost and decision tree models. These reached over 98% accuracy for known ransomware. These features are effective for static file scans but cannot detect encrypted or memory-loaded malware.

Entropy reflects randomness in a file’s content. Malware often uses packing or encryption, which increases entropy. Kumar & Singh (2022) used entropy thresholds to flag malware. However, compressed or installer files also show high entropy, so this can lead to false positives.

## **3.2 System Behaviour and Event Logs**

Many papers examine what the system does during an attack. Features here include program execution, file changes, registry edits, user account creation, and scheduled tasks. These appear in logs from Windows Event Viewer or Linux audit logs.

In STARFLEET, logs showed that PowerShell was used to disable Defender (Set-MpPreference), and a batch script shut down services. Patel et al. (2023) used Windows Event IDs—4688 (process creation), 4657 (registry change), and 7045 (new service)—as input to ML models. These helped identify malicious activity as it occurred.

Zhang et al. (2024) built process graphs from audit logs, where each node was a process or action. Edges showed execution order. Their Graph Neural Network detected behaviour patterns such as injection chains and fileless execution, which are hard to catch using signatures.

## **3.3 Network Features**

Network traffic provides another useful feature set. These include connection frequency, packet size, timing, protocols used, and repeated access to ports. Dionaea logs all incoming IPs and what services they try to access (Sethia & Jeyasekar, 2019). These logs help detect malware spreading across the network.

Ali et al. (2024) analysed HTTP logs for signs like long URLs, odd headers, and repeated login attempts. They trained an Isolation Forest model to detect outliers. Features like URL entropy or header length helped spot brute-force or scripted attacks.

In Rahim et al. (2024), SIEM tools tracked CAN bus messages in cars. Features included repeated control commands, mismatched IDs, and timing anomalies. These helped detect potential tampering. This shows how feature selection must match the system being protected.

## **3.4 User Behaviour and Authentication Data**

Some studies focused on login behaviour. AlSaleem & Alshoshan (2021) used a graphical MFA system where users clicked on images in sequence. The system logged image order, device ID, and timing to check if the attempt matched the user’s normal behaviour. This helped detect spoofed logins, though it didn’t stop visual observation.

Gadducci et al. (2021) reviewed ten MFA methods and extracted features such as OTP length, expiry time, device fingerprint, and how often backup options were used. These were analysed to assess how easily attackers could bypass each method. Features helped show weaknesses in email- or SMS-based MFA.

## **3.5 Honeypot Interactions**

Honeypots generate features when attackers interact with them. Zhuravchak et al. (2021) placed symbolic link decoys across the system. When ransomware touched one, it logged the time, process ID, and filename—useful signals for early detection.

Voerman et al. (2020) described SIEM-integrated honeypots that logged attacker IPs, commands run, and how many files or folders were accessed. These logs were reused to retrain detection models and improve auto-responses. Honeypots act as both a detection tool and a source of labelled training data.

## **3.6 Challenges with Features**

Not all features are easy to use. Some are noisy. High entropy might suggest malware—or simply a zip file. Registry edits might mean infection—or a user installing software. Models must be carefully tuned to avoid false alerts.

A common challenge is class imbalance. Attack behaviour appears rarely in logs. ML models may ignore rare attack features unless the dataset is adjusted using oversampling or synthetic data.

Real-time collection is also difficult. Memory-based features, in particular, require specific tools. Privacy concerns can arise when tracking user behaviour.

Finally, features vary by environment. Event logs in Windows don’t exist on Linux. Web logs won’t help detect desktop malware. This is why many studies recommend using a mix of feature types—file-level, behavioural, network, user-based—and adapting them to the context in which they are deployed.

# **4. Discussion**

The reviewed research shows several strong techniques to detect and stop attacks like the STARFLEET breach. But each comes with limitations when applied in real systems. This section compares methods from earlier sections and evaluates whether they would work in practice.

Machine learning models can detect ransomware early using file signatures or behaviour patterns. Static models that analyse opcodes or file structure perform well on known malware in offline scans (Arabo et al., 2023), but are ineffective when malware is encrypted, packed, or runs only in memory. Behaviour-based models are more suitable for STARFLEET-style attacks, which used in-memory PowerShell and disabled security tools (Kumar & Singh, 2022). These models rely on logs from system events, services, and registries, but need clean, complete datasets and ongoing tuning. In real-world networks, logs are often missing, incomplete, or noisy—making detection less reliable.

Honeypots offer direct evidence of attack attempts. Dionaea captures malware files by pretending to run vulnerable services, while symbolic link honeypots log ransomware interactions at the file level (Sethia & Jeyasekar, 2019; Zhuravchak et al., 2021). These can give early warnings and samples for analysis. When connected to SIEM tools, they help enrich detection models. However, honeypots only generate signals if touched. Skilled attackers or modern malware may detect them and bypass them. Placement also matters—a honeypot that is never reached is ineffective.

Multi-factor authentication (MFA) helps prevent credential theft, which was the first step in STARFLEET. Papers show hardware- and biometric-based MFA to be much stronger than SMS or email codes (Gadducci et al., 2021). AlSaleem & Alshoshan (2021) also showed image-based MFA adds a behavioural layer. But once an attacker gets past MFA—via social engineering or token theft—these protections don’t stop what happens next. So while MFA helps at the entry point, it must be paired with deeper internal defences.

Network and log-based methods provide wide coverage across the attack chain. Techniques like process graphs (Zhang et al., 2024), log-based Random Forest classifiers (Singh et al., 2022), and Isolation Forest on access patterns (Ali et al., 2024) are able to detect abnormal behaviour. These rely on detailed and trustworthy logs. But, as STARFLEET showed, attackers may delete or disable logs to avoid detection. In such cases, these models can fail entirely.

Across all studies, the strongest systems combined multiple approaches. Honeypots and behavioural models linked into a SIEM platform give visibility across phishing, privilege escalation, and ransomware stages. This layered defence is especially valuable in environments like STARFLEET, where attackers move through many steps. However, such systems need careful setup, frequent updates, and alert tuning. Without this, they generate too many false alarms or miss real threats.

In summary, while each tool helps with part of the problem, no single method is enough. The most effective protection comes from combining approaches, tuned to local conditions and supported by reliable logging and response infrastructure.

# **5. References**

**AI/ML for Ransomware Detection**

1. Arabo, A., Abdulghani, M. I., & Nordin, R. (2023). A digital DNA sequencing engine for ransomware detection using machine learning. *NDSS Symposium*. <https://www.ndss-symposium.org/ndss-paper/a-digital-dna-sequencing-engine-for-ransomware-detection-using-machine-learning/>
2. Kumar, R., & Singh, R. (2022). AI-based ransomware detection. *Computers & Security*, *115*, 102605. <https://doi.org/10.1016/j.cose.2022.102605>
3. Patel, P., Shah, K., & Desai, M. (2023). Exploring ransomware detection using ML and AI. *IEEE Transactions on Information Forensics and Security*, *18*(6), 893–906. <https://doi.org/10.1109/TIFS.2023.3245641>

**Honeypots**

1. Zhuravchak, K., Kryvinska, N., & Strauss, C. (2021). Ransomware prevention system design based on file symbolic linking honeypots. *Computers & Security*, *104*, 102226. <https://doi.org/10.1016/j.cose.2021.102226>
2. Voerman, E., Bou-Harb, E., & Hassan, M. (2020). The role of honeypots in modern cybersecurity strategies. *Springer Lecture Notes in Computer Science*, *12123*, 214–229. <https://doi.org/10.1007/978-3-030-50729-5_13>
3. Sethia, M., & Jeyasekar, R. (2019). Malware capturing and analysis using Dionaea honeypot. *2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI)*, 917–921. <https://doi.org/10.1109/ICOEI.2019.8862615>

**Multi-Factor Authentication (MFA)**

1. AlSaleem, K., & Alshoshan, A. (2021). Multi-factor authentication to systems login. *International Journal of Computer Applications*, *182*(36), 25–30. <https://doi.org/10.5120/ijca2021921234>
2. Gadducci, F., Loddo, G., & Marras, M. (2021). An extensive formal analysis of multi-factor authentication protocols. *ACM Transactions on Privacy and Security (TOPS)*, *24*(3), 1–38. <https://doi.org/10.1145/3439724>

**Log Monitoring and SIEM**

1. Zhang, J., Wang, H., & Liu, Q. (2024). A novel malware detection method based on audit logs and graph neural network. *Computers & Security*, *125*, 102972. <https://doi.org/10.1016/j.cose.2023.102972>
2. Singh, A., Malhotra, A., & Bhargava, R. (2022). Integrated security information and event management (SIEM) with intrusion detection system (IDS) for live analysis based on machine learning. *Computers & Security*, *112*, 102525. <https://doi.org/10.1016/j.cose.2021.102525>
3. Ali, A., Ahmed, R., & Shah, M. (2024). Identifying security threats in the system using automated security logs. *IEEE SMART 2024 Conference Proceedings*. <https://ieeexplore.ieee.org/document/10452694>
4. Rahim, A., Mazlan, M., & Ariffin, S. (2024). Developing SIEM and log management for automotive network in a simulated environment. *IEEE Transactions on Vehicular Technology*, *73*(1), 812–821. <https://doi.org/10.1109/TVT.2024.3123645>
5. Ali, A., Ahmed, R., & Shah, M. (2024). Threat classification model for security information event management focusing on model efficiency. *Computers & Security*, *126*, 102998. <https://doi.org/10.1016/j.cose.2024.102998>