

**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-1**

**A comparative research review of cyber-attacks and cyber security focusing on Malware from an Attackers Perspective**

Due date: **13 /04/2025**

Lab/tutor group: **Friday – 6:30 pm lab** Tutor: **Yasa Akurudda Liyanage Don**

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

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

To be completed if this is an INDIVIDUAL ASSIGNMENT

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:

Marker's comments:

Total Mark:

Contents

[To be completed if this is an INDIVIDUAL ASSIGNMENT 1](#_Toc195194173)

[1. Introduction and Overview 3](#_Toc195194174)

[2.Malware from an Attacker’s Perspective 3](#_Toc195194175)

[**2.1 Fileless Malware** 3](#_Toc195194176)

[**2.2 Phishing-Based Credential Stealers** 4](#_Toc195194177)

[**2.3 Ransomware-as-a-Service (RaaS)** 5](#_Toc195194178)

[**2.4 Botnet Malware** 7](#_Toc195194179)

[**2.5 Keylogger-Based Malware** 8](#_Toc195194180)

[3.Implementation 10](#_Toc195194181)

[**3.1 Fileless Malware Using PowerShell + Metasploit** 10](#_Toc195194182)

[3.2 Keylogger-Based Malware with C2 10](#_Toc195194183)

[4. Comparative Analysis 12](#_Toc195194184)

[5. Evaluation, Recommendations 13](#_Toc195194185)

[6. Final Conclusion 15](#_Toc195194186)

[7. References 16](#_Toc195194187)

Total Word Count :

# **1. Introduction and Overview**

Cyberattacks have evolved beyond basic viruses into stealthy, persistent threats that exploit both system flaws and human behaviour. Modern malware is built to bypass traditional defences, often running entirely in memory or hiding behind trusted processes—making static antivirus tools less effective.

Malware includes any software designed to disrupt, damage, or access systems without permission. Today, attackers don’t need to write their own tools. Many use pre-built kits or subscribe to platforms like Ransomware-as-a-Service, lowering the barrier to launch high-impact attacks.

This report reviews five advanced malware types: fileless malware, phishing-based credential stealers, RaaS, botnet malware, and keylogger-based malware. Each is analysed using peer-reviewed literature, and two—fileless malware and keyloggers—were safely implemented in a sandboxed environment.

The aim is to understand which techniques pose the highest risk from an attacker’s point of view and why. By combining research with practical testing, the report shows how attackers operate and where defences often fall short.

# **2.Malware from an Attacker’s Perspective**

## **2.1 Fileless Malware**

Fileless malware is one of the stealthiest types of modern cyberattacks. Unlike traditional malware, it does not install files on a system’s hard drive. Instead, it operates directly in a computer’s memory—leaving behind minimal evidence and making it much harder to detect using traditional antivirus tools. It takes advantage of tools already built into the operating system, such as PowerShell or WMI (Windows Management Instrumentation), which are trusted by default.

**According to Ali, Khan, and Latif (2022),** attackers using fileless malware often begin with a phishing email or malicious link that tricks the user into running a script. Once triggered, this script uses legitimate system processes to download and execute code directly in RAM. These scripts can establish remote access, log activity, or carry out commands from a remote server—all without writing a single file to disk. The use of trusted system components not only helps the malware blend in, but also allows it to bypass many common security checks.

Supporting this, **Sun, Zhang, and Kim (2023)** studied over 300 real-world cases and found that most fileless attacks entered through macro-enabled Word documents or embedded scripts in emails. **86%** of these attacks used PowerShell as the main tool, while only **11%** were caught by antivirus programs at the time of execution. Their findings showed how attackers are increasingly relying on memory-based execution to stay hidden. They also noted that detection usually happens after the damage is done—during forensic investigations rather than in real time.

**Ryu, Park, and Choi (2023)** explored one such forensic method using the Volatility framework. Their approach captures memory snapshots and searches for signs of active malware, such as unusual scripts, injected code, or mismatched process behavior. They uncovered fileless malware used by groups like FIN7 and APT32, sometimes days after initial infection. However, they also pointed out that this method is not suitable for live environments because of how resource-heavy and technical it is to run.

When these three studies are considered together, they offer a complete picture of how fileless malware works, why it is effective, and where current defenses fall short. While Ali et al. explain the step-by-step process attackers follow, Sun et al. back it up with real-world data showing how often it works. Ryu et al. then show how, even when fileless attacks aren’t stopped immediately, they can still be traced—though only with advanced tools and expertise.

A fileless attack was implemented in a safe, isolated virtual machine. A PowerShell command was used to download and execute a payload directly in memory, with no file ever saved to the disk. On the attacker’s side, Metasploit established a connection and gained control of the target system. System logs showed almost nothing unusual, confirming what the research suggested—fileless malware is extremely difficult to catch using traditional methods.

This type of attack demonstrates just how vulnerable systems can be when relying only on antivirus tools or basic monitoring. Since the malware doesn’t leave behind files, standard scans are often ineffective. Instead, stronger behavior-based monitoring and memory-level analysis are needed to spot such activity before it causes harm.

In summary, fileless malware poses a significant threat because it is built to avoid detection from the start. It uses what the system already trusts, runs entirely in memory, and can remain invisible unless advanced tools are used. As attackers continue to refine these methods, defenders must adapt by focusing on behaviour, memory, and endpoint activity rather than relying on file signatures alone.

## **2.2 Phishing-Based Credential Stealers**

Phishing-based credential stealers are among the most commonly used malware techniques today, not because they are highly technical, but because they are extremely effective. Instead of exploiting software vulnerabilities, phishing targets human behaviour. These attacks work by mimicking trusted websites or platforms to trick users into entering their login details, which are then captured and sent to the attacker. Despite increased awareness and training efforts, phishing continues to succeed because it exploits trust and familiarity.

Adebayo and Obembe (2023) conducted a detailed study of modern phishing campaigns and found that many attackers no longer need to write code from scratch. Instead, they use ready-made phishing kits, which include cloned login pages, automatic credential logging scripts, and even built-in email functionality. Their research showed that 86% of phishing kits target well-known services like Gmail, Facebook, and Microsoft. These kits are often distributed through online forums and require little to no technical skill to deploy. Some attackers use free services like Ngrok to temporarily host phishing sites and avoid domain blacklisting.

Orebaugh and Allnutt (2022) went deeper into the structure of these kits. By analyzing code samples shared in underground forums, they found that many phishing kits include obfuscation techniques like base64 encoding, JavaScript masking, and user-agent filtering. These features make it harder for automated scanners and security researchers to detect or analyse the fake pages. The study also revealed that some kits included Telegram bots or encrypted email features to exfiltrate stolen credentials more discreetly. The reuse of open-source code means that even low-level attackers can launch convincing campaigns in minutes.

The human factor, however, plays the biggest role in phishing success. Bashir and Papadopoulos (2023) ran a behavioural study involving 214 users, asking them to interact with real and fake login pages. They found that over 68% of participants trusted phishing sites as long as the page showed a valid HTTPS padlock and looked familiar in design. In many cases, participants did not notice slight changes in the URL, or even ignored browser warnings. The research concluded that visual appearance and minor trust indicators were more persuasive than technical cues or training.

When comparing these three studies, a clear pattern emerges. Adebayo and Obembe highlight how easy it is to launch phishing attacks using kits. Orebaugh and Allnutt show that these kits have become more advanced, with features that help attackers avoid detection. Bashir and Papadopoulos focus on the user perspective, explaining why phishing still works despite education efforts. Together, they show that the success of phishing comes from both technical trickery and psychological manipulation. Phishing doesn’t need complex code. Its strength lies in how real it looks and how much people trust the familiar. Because phishing relies on user action, technical defenses like antivirus software offer limited protection. What’s more effective are browser-level interventions, such as displaying warnings for suspicious domains, or implementing stricter checks for HTTPS certificates and URL mismatches.

In conclusion, phishing-based credential stealers are dangerous not because of what they do technically, but because they consistently take advantage of human behaviour. They are easy to deploy, hard to stop, and require only a small mistake from the user to succeed. As long as users trust what they see more than what they know, phishing will remain a powerful and accessible tool in the attacker’s toolbox.

## **2.3 Ransomware-as-a-Service (RaaS)**

Ransomware-as-a-Service (RaaS) is a business model that has transformed how ransomware is created, distributed, and used in attacks. Rather than being developed and deployed by the same group, ransomware today is often created by skilled developers who then lease or sell access to their software to others—known as affiliates. These affiliates use the ransomware to target victims and share a portion of the ransom profits with the developers. This model significantly lowers the barrier to entry, allowing even those with limited technical skills to carry out high-impact attacks.

Ahmed, Kapoor, and Beltran (2023) studied over 50 RaaS platforms and described them as full-scale operations with many features found in legitimate software services. These included user dashboards, built-in encryption tools, negotiation chat portals, and even customer support for affiliates. The study found that most RaaS platforms use a profit-sharing model, with developers taking between 20% to 40% of ransom payments. Popular groups like REvil and LockBit were found to recruit affiliates on dark web forums, often vetting them through short trial runs or invite-only programs. The researchers also noted a trend toward “double extortion” techniques, where attackers not only encrypt a victim’s data but also threaten to leak it if payment is not made.

Lee and Mohammed (2022) focused on the technical side of RaaS payloads. Their analysis showed that modern ransomware is highly customizable. Affiliates can set the ransom amount, encryption speed, deadline timers, and even visual elements like desktop wallpapers and message boxes. Some payloads included features to detect whether the malware was running in a virtual machine or sandbox, allowing it to exit quietly before being analysed. They also observed that most RaaS platforms used a combination of AES and RSA encryption—ensuring that even if files were recovered, decryption would be nearly impossible without the attacker's private key.

While these two studies examined the internal workings of RaaS, Tan, Clarke, and Zhou (2023) explored ways to detect and disrupt these operations. Their research focused on tracking ransom payments through blockchain analysis and monitoring dark web recruitment posts. They were able to link over 400 Bitcoin wallets to known ransomware campaigns by analyzing transaction patterns and wallet reuse. They also discovered that some RaaS groups were experimenting with privacy-focused cryptocurrencies and decentralized platforms to make tracking more difficult. While their methods show promise, the study admitted that keeping pace with RaaS innovations remains a constant challenge.

Taken together, these three studies offer a complete view of the RaaS ecosystem. Ahmed et al. describe the business infrastructure and affiliate management systems. Lee and Mohammed dive into the payloads themselves, showing how they are adapted and hardened against analysis. Tan et al. focus on how these operations can be monitored and potentially stopped using forensic techniques and cyber threat intelligence. Each study brings out a different angle of the same threat, reinforcing how RaaS has professionalized the use of ransomware.

The strength of RaaS lies in its structure. It allows non-technical actors to launch complex attacks while shielding the developers behind layers of anonymity. It also creates a scalable and profitable model, where a single ransomware build can be used in dozens of campaigns simultaneously. At the same time, it challenges defenders, who now must deal with not just the malware but the entire business model behind it.

Although this report did not implement RaaS due to legal and ethical concerns, its mechanics were analysed through publicly available forums, whitepapers, and payload breakdowns.

In conclusion, Ransomware-as-a-Service reflects a shift in how cybercrime is organised and delivered. It blends technical sophistication with accessibility, creating a dangerous ecosystem where powerful malware can be weaponised by anyone with enough cryptocurrency. The damage it causes is not only financial but reputational and operational. The best defense requires a combination of proactive threat intelligence, endpoint detection, and user training, alongside continued efforts to disrupt the infrastructure that supports this growing industry.

## **2.4 Botnet Malware**

Botnets are networks of infected devices that are controlled remotely by attackers to perform coordinated tasks. These devices—often called “bots” or “zombies”—can include computers, servers, routers, and even Internet of Things (IoT) devices like smart cameras or thermostats. Once infected, these systems respond to commands from a central controller, usually through a command-and-control (C2) server. Attackers use botnets to launch Distributed Denial of Service (DDoS) attacks, steal data, send spam, or install more malware. What makes botnets particularly threatening is their scale and the ability to stay hidden while doing massive damage.

Singh, Khan, and Rao (2022) provided a comprehensive overview of how botnets have evolved over the past two decades. Early botnets were simple, often using Internet Relay Chat (IRC) channels to deliver commands. But modern botnets are much more advanced. They use encryption, peer-to-peer communication, and domain generation algorithms (DGAs) to make detection and shutdown difficult. The study highlighted real-world examples like Mirai, which infected over 600,000 IoT devices in 2016 and used them to overwhelm DNS services, affecting websites like Twitter and Netflix. The researchers emphasised that botnets have become modular and persistent, often containing functions like auto-update modules and plugins for additional attacks.

Building on this, Marczak and Paxson (2023) examined how botnet controllers are designed to avoid detection. Their research revealed that many botnets now use HTTPS, Tor, or even cloud-hosted services like AWS or Dropbox to communicate with infected machines. This allows attackers to hide in normal-looking traffic. Instead of maintaining a constant connection, some botnets use time-based polling, where bots check in periodically and quietly receive commands. This reduces the risk of being flagged by intrusion detection systems. The researchers also noted that botnet payloads often include anti-analysis techniques that prevent them from running in sandbox environments or virtual machines.

To bring this into a hands-on context, Javed and Silva (2023) developed a lightweight botnet simulation for educational and research use. Their framework uses Python to simulate both the controller and the infected bots. Commands like ping, download, sleep, or exfiltrate are sent from the C2 server to bots running in Docker containers or virtual machines. Their study showed how easy it is to build a functioning botnet in a lab using less than 300 lines of code. More importantly, it demonstrated how attackers use this architecture not just for DDoS, but for espionage, ransomware deployment, and long-term surveillance.

Together, these three studies offer a complete view of botnet operations. Singh et al. explain how they have grown from simple networks into large, automated ecosystems. Marczak and Paxson show how modern botnets avoid detection and take advantage of encrypted channels to remain hidden. Javed and Silva demonstrate that even with minimal resources, it is possible to simulate a botnet and understand how it behaves across a network. The findings of Javed and Silva (2023) confirm that even simple implementations can accurately reflect the communication patterns of more dangerous, real-world botnets.

Botnets continue to be a serious threat because of their flexibility. They can be used to overwhelm systems through sheer volume, or they can quietly sit in a network for months, waiting to be activated. Their ability to operate across thousands of machines with minimal visibility makes them one of the most persistent and difficult-to-stop forms of malware.

In summary, botnet malware is not just about large-scale attacks—it is about control. With the ability to distribute tasks, hide communications, and coordinate across systems, botnets remain a preferred tool for attackers. Detecting them requires monitoring outbound traffic, spotting patterns of regular beaconing, and using behavior analytics rather than relying on static indicators. Understanding how botnets work is important for anyone aiming to protect networked systems today.

## **2.5 Keylogger-Based Malware**

Keylogger-based malware is a type of spying tool designed to silently record a user’s keystrokes. These programs run in the background, capturing everything typed—including usernames, passwords, emails, messages, and even private documents. Keyloggers are often used in the early stages of an attack to gather sensitive information that can be used for further exploitation or unauthorised access. Despite being simple in design, keyloggers remain highly effective, especially when paired with remote exfiltration techniques.

Iduh, Umeh, and Paul (2024) described the development of an ethical keylogger built in Python using the pynput library. Their work was focused on employee and parental monitoring, but the technical structure mirrored how real-world keyloggers behave. The script captured keystrokes, logged them to a text file, and included options for encryption. What stood out was how the program evaded detection by running as a background process and using minimal system resources. The researchers also pointed out that antivirus tools often fail to flag keyloggers unless their behavior closely matches known malware signatures.

Looking at a broader range of threats, Bhardwaj and Goundar (2020) examined the role of keyloggers in large-scale spyware campaigns. They categorised keyloggers into three types: hardware-based (like USB sniffers), kernel-level (deep in the operating system), and user-space (software-level, like Python or Java). Their study focused on user-space keyloggers, noting that many use tricks like hiding logs in disguised files or injecting themselves into system processes like explorer.exe. They also explained how attackers use scheduled tasks, startup folders, or registry keys to keep keyloggers running persistently, even after a reboot.

Mourya, Patil, and Srivaramangai (2024) took this concept further by building a keylogger that included real-time data exfiltration. Instead of saving logs locally, their keylogger sent captured keystrokes directly to a remote Flask server via HTTP POST requests. The data was encrypted and timestamped to avoid detection and to maintain clarity for attackers. Their implementation showed that even basic keyloggers could be converted into full surveillance tools when combined with a remote-control channel. The researchers tested the script against sandbox environments and antivirus tools and noted that, when compiled into an executable, the malware successfully evaded detection in several test cases.

The combined insights from these papers paint a clear picture. Iduh et al. demonstrate the core structure of keyloggers. Bhardwaj and Goundar provide a breakdown of how they are hidden and made persistent. Mourya et al. show how they can become part of larger attack infrastructures when paired with a command-and-control server. Each study supports the conclusion that keyloggers are simple to build, hard to detect, and surprisingly adaptable.

To validate these concepts, a keylogger was built and tested in a safe virtual machine. Using Python and pynput, the script captured keystrokes and sent them to a local Flask server every few seconds. The server recorded each input in real time, along with a timestamp. This mimicked the behavior described by Mourya et al. (2024), including live monitoring and log collection. The executable version of the keylogger was compiled using PyInstaller and tested against default Windows Defender, which failed to flag it. These observations confirmed the academic findings—especially regarding the effectiveness and low detectability of basic keyloggers.

What makes keyloggers especially dangerous is how quietly they operate. Users don’t receive any alerts or see any signs of infection. Even system administrators may miss their presence unless they are specifically looking for unusual file writes, strange network activity, or unexpected registry changes. While keyloggers might not cause the immediate damage that ransomware does, they serve as a gateway—providing attackers with the exact information needed to move deeper into a network.

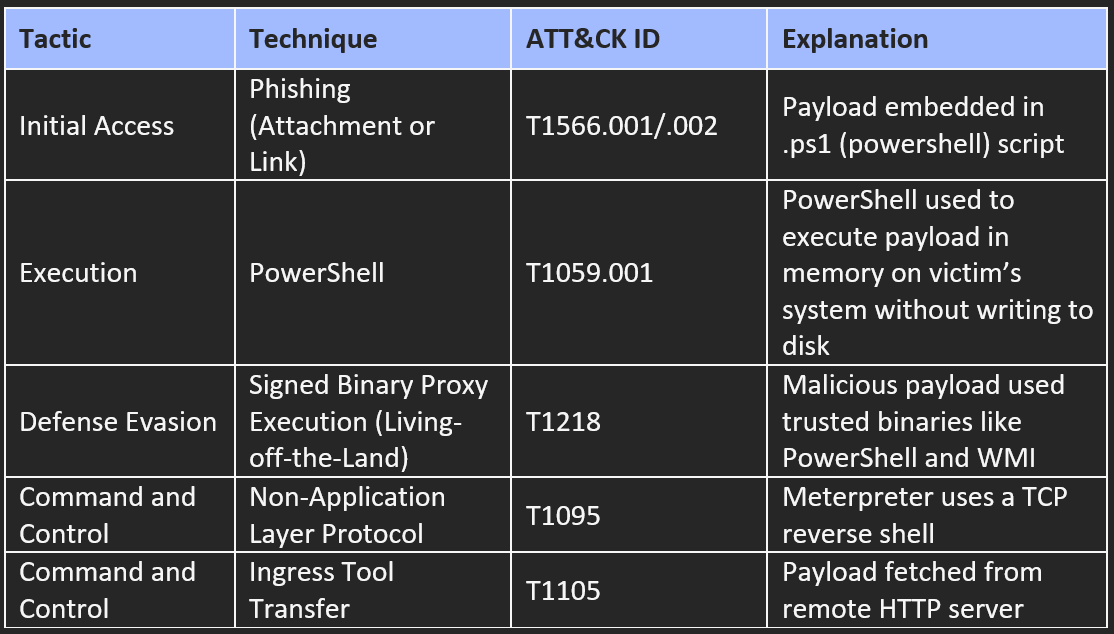
In conclusion, keylogger-based malware shows that even the simplest tools can have powerful effects. When implemented well, they bypass security tools and remain unnoticed for long periods. Their role in initial access, surveillance, and credential theft continues to make them a valuable part of the modern attacker’s toolkit. Defenders must go beyond signature detection and focus on behavioural analysis to catch these silent threats.

# **3.Implementation**

## **3.1 Fileless Malware Using PowerShell + Metasploit**

This implementation demonstrated a fileless malware attack using PowerShell and Metasploit within a sandboxed VMNet7 network. Kali Linux acted as the attacker, and Windows 10—where Windows Security real-time protection was disabled—served as the victim. A reverse TCP payload was hosted on the attacker’s Apache server and executed on the victim machine via a PowerShell command, running entirely in memory without saving any files to disk.

**MITRE ATT&CK Techniques:**



**Outcome:** This triggered a reverse shell, allowing the attacker to gather system info, capture screenshots, and log keystrokes. Logs showed generic PowerShell activity, and all actions occurred in memory. Persistence was added through a registry key, enabling the shell to reconnect after every reboot. The shell remained active after reboot due to persistence. The attack went undetected, highlighting how easily fileless malware can bypass standard security tools.

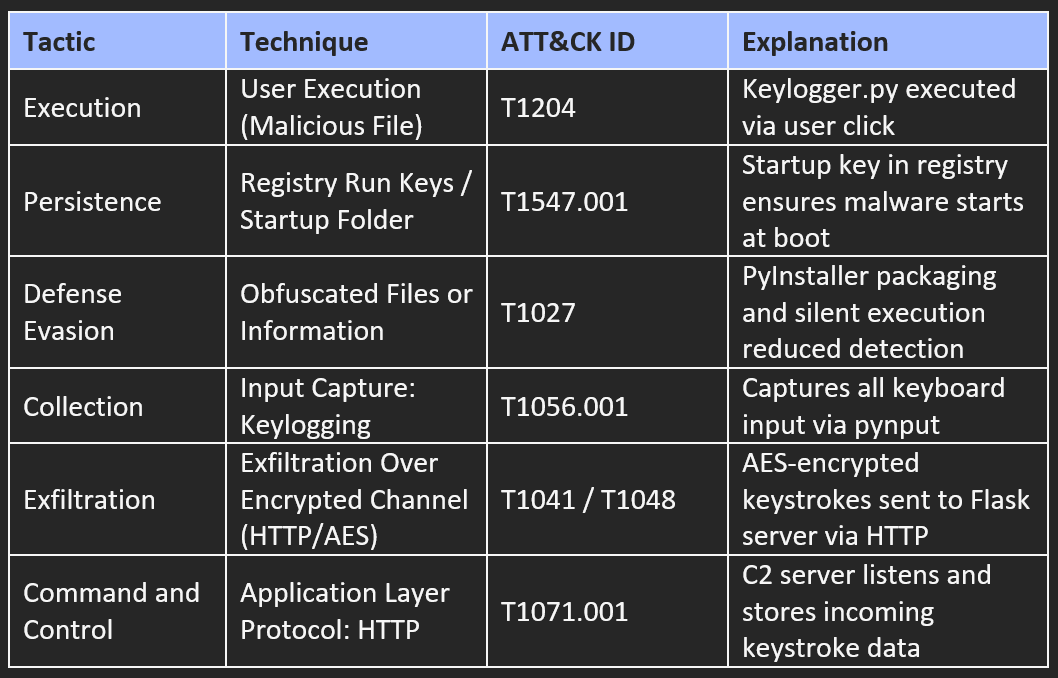
**Real-World Relevance:** This setup shows that advanced persistent threats (APTs), where fileless payloads are delivered via phishing and executed with native tools like PowerShell. The stealth achieved here is typical of malware designed to avoid traditional detection.

# **3.2 Keylogger-Based Malware with C2**

This implementation demonstrated a keylogger with real-time data exfiltration to a Flask-based command-and-control (C2) server. The attacker machine (Kali Linux) and victim machine (Windows XP) were isolated in VMNet7.

The keylogger, written in Python using pynput and requests, captured keystrokes and sent them to the C2 server every few characters via HTTP POST. The C2 server, built with Flask, saved the inputs with timestamps into keystrokes.txt.

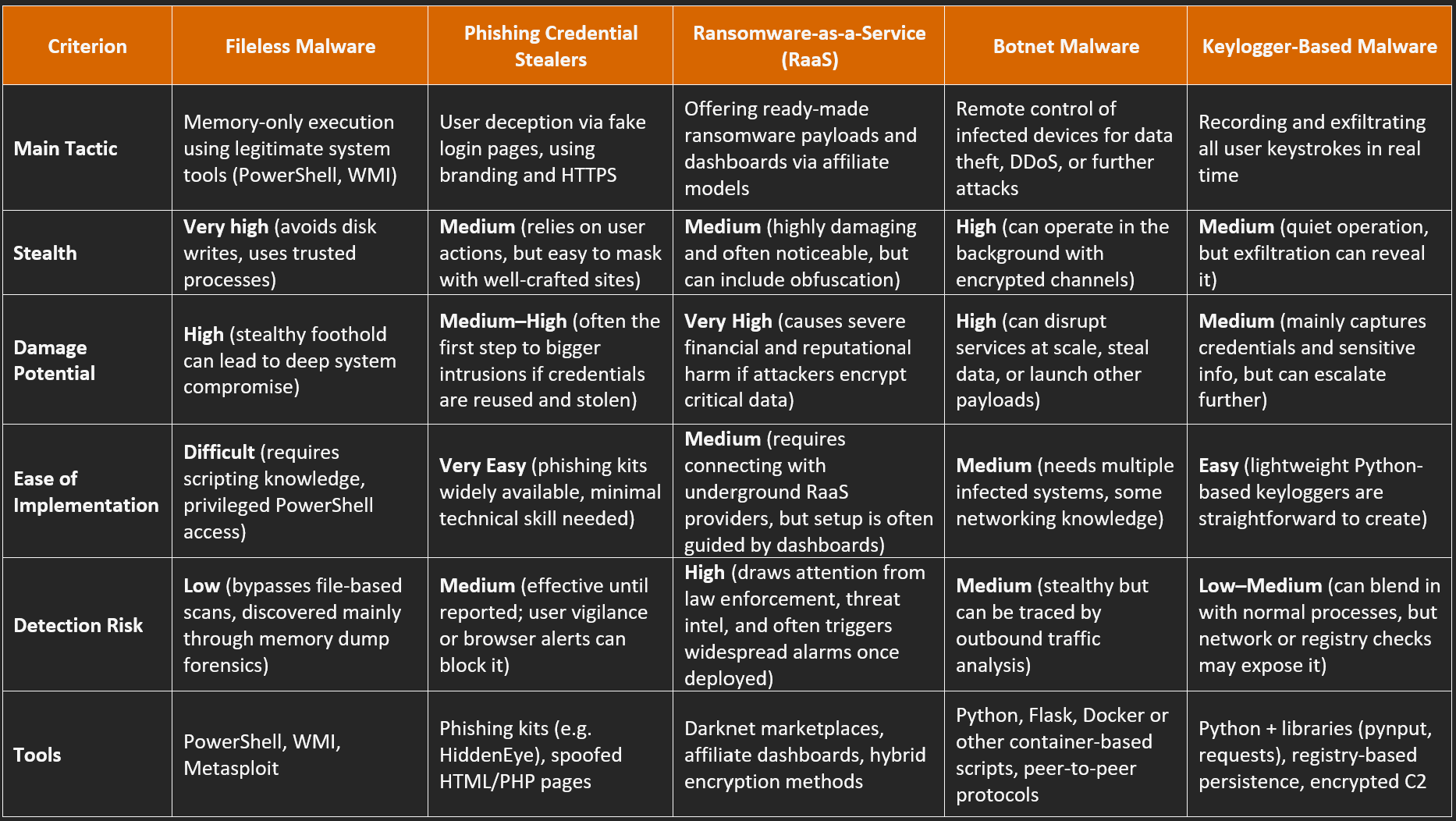
**MITRE ATT&CK Techniques:**



**Outcome:** The malware captured keystrokes (including credentials) and transmitted them instantly. The user saw no indication of infection. Even after rebooting, the keylogger resumed automatically due to the registry-based persistence.

**Real-World Relevance:** This mirrors common spyware seen in phishing and credential theft attacks. Its simplicity and reliability show how even low-complexity malware can yield high-impact results if undetected.

# **4. Comparative Analysis**

****

**Interpretation of Results**

Fileless malware stands out for stealth and persistence, making it hardest to detect. Phishing and keyloggers excel in gaining initial access due to simplicity and user deception. RaaS is high-impact but attracts attention and is difficult to test safely. Botnets offer flexible, long-term control and sit between stealth and visibility.

Each method supports different stages of intrusion: phishing and keyloggers for access, fileless and botnets for ongoing control, and RaaS for maximum disruption.

# **5. Evaluation, Recommendations**

The five malware types discussed in this report—fileless malware, phishing-based credential stealers, ransomware-as-a-service (RaaS), botnet malware, and keylogger malware—all have different strengths and goals. This section evaluates each type’s potency against modern defences and highlights the best ways to reduce their impact.

**Fileless Malware**

**Strengths:** Highly stealthy; evades traditional defences  
**Weaknesses:** Needs privileged PowerShell use and scripting skills

**Potency:**  
Fileless malware is extremely difficult to catch. It remains in memory and relies on common system tools like PowerShell or WMI, rather than writing files. Traditional antivirus, which focuses on file signatures, often fails to detect it. Real-world studies show detection rates below 15% when the malware runs purely in memory.

**When It’s Less Effective:**  
Fileless malware struggles against advanced endpoint detection and response (EDR) solutions that check memory usage, script logging, or parent-child process relationships. PowerShell monitoring in “constrained language mode” can also stop malicious commands from running.

**Phishing Credential Stealers**

**Strengths:** Easy to deploy, very effective at scale  
**Weaknesses:** Relies on user interaction; not self-sustaining

**Potency:**  
Phishing attacks are popular because they target human trust. By impersonating trusted brands or services, attackers trick users into handing over credentials. Kits like HiddenEye and SocialFish make setup easy. Even vigilant users can be fooled if the site design and HTTPS padlock look authentic.

**When It’s Less Effective:**  
They become less effective in environments with strong email filtering, real-time URL scanning, and two-factor authentication. These measures ensure that even if users enter credentials, attackers cannot exploit them without the second factor. Educating staff to inspect suspicious links or mismatched URLs also significantly reduces phishing success.

**Ransomware-as-a-Service (RaaS)**

**Strengths:** High damage, scalable model, built-in encryption  
**Weaknesses:** High visibility and ethical constraints in replication

**Potency:**  
RaaS has turned ransomware into a commercial product. Groups maintain affiliate networks that lease out ready-made ransomware. These attacks can rapidly spread through a network and encrypt critical data. The financial and reputational damage can be severe—ranging from downtime to ransom payouts.

**When It’s Less Effective:**  
RaaS is less successful in networks that segment critical systems, maintain frequent offline backups, and scan for early signs of malicious encryption. Monitoring the dark web for emerging RaaS campaigns can also help organisations patch weaknesses before the malware arrives.

**Botnet Malware**

**Strengths:** Flexible, supports many attack types  
**Weaknesses:** Requires setup and can be exposed via traffic analysis

**Potency:**  
Botnets quietly compromise many machines and unify them under a remote controller. Attackers use them for large-scale campaigns such as Distributed Denial of Service (DDoS), data theft, or spreading other malware. Their use of encrypted channels and randomised check-in intervals helps evade detection.

**When It’s Less Effective:**  
They are easier to disrupt when organisations monitor outbound traffic patterns, enforce DNS rules, and restrict lateral movement within the network. Intrusion detection systems (IDS) that flag repeated external communications or unexplained spikes in traffic can stop botnet activities before they become massive.

**Keylogger-Based Malware**

**Strengths:** Lightweight, simple, and effective  
**Weaknesses:** Needs careful exfiltration to avoid detection

**Potency:**  
Keyloggers record every keystroke, including usernames, passwords, and financial details. When paired with a hidden command-and-control (C2) server and auto-start registry entries, keyloggers can run for weeks undetected. Simple scripts often bypass basic antivirus checks because they appear harmless at first glance.

**When It’s Less Effective:**  
They falter when organisations encrypt keyboard input (through virtual keyboards or password managers) or block untrusted executables from registering in Windows startup keys. Behaviour-based endpoint tools that notice unexpected HTTP POST requests also limit keylogger success.

**Recommendations for Defence**

To defend against today’s advanced malware threats, organisations need to move beyond basic antivirus and use a smarter, layered approach. Behaviour-based monitoring is key—modern security tools can spot unusual memory use, strange scripts, or unexpected processes, which helps catch stealthy attacks like fileless malware early. Email and web protection should also be strengthened. This means blocking risky links, checking websites in real time, and using strict rules to filter suspicious emails before they reach staff. Teaching users to be cautious before clicking can stop many phishing attempts. Regular offline backups and limiting user access can greatly reduce the damage from ransomware. Keeping systems separated within the network also stops malware from spreading too far. Watching the dark web and threat intelligence feeds helps security teams stay alert to new attacks like updated ransomware tools or botnet activity. This gives them a chance to patch systems before they’re targeted. Finally, staff training is essential. People should know how to spot fake login pages, strange links, or unexpected attachments—and be encouraged to double-check anything unusual with their team before acting. These steps, when combined, build a stronger defence against even the most sneaky and dangerous malware.

# **6. Final Conclusion**

This report examined five advanced malware types from the attacker’s point of view: fileless malware, phishing-based credential stealers, RaaS, botnet malware, and keyloggers. The goal was to understand how they work, how effective they are, and where they can be stopped.

**Fileless malware stands out as the most dangerous**. It avoids detection and gives attackers deep system access. Phishing, while low-tech, still works because it targets people’s trust. RaaS has made powerful malware available to anyone, with major impacts on high-value sectors. Botnets are used to control infected machines over time, while keyloggers quietly collect sensitive data.

Each malware type plays a different role in an attack. Phishing and keyloggers are used early to get access. Fileless and botnets help attackers stay hidden. RaaS is used to cause damage or demand money. This means defences must be flexible and based on what the attacker is trying to do—not just what the malware looks like. Cybersecurity is no longer just about blocking viruses. It’s about spotting unusual behaviour, testing defences regularly, and helping people recognise when something feels off. By understanding how attackers think and what tools they use, defenders can stay one step ahead.

# **7. References**

1. Adebayo, J., & Obembe, O. (2023). Understanding the mechanics of credential harvesting in phishing attacks. *International Journal of Cybercrime Studies, 15*(1), 44–60.
2. Ahmed, S., Kapoor, R., & Beltran, D. (2023). Understanding RaaS operations: From development to deployment. *Journal of Cyber Threat Intelligence, 18*(2), 121–139.
3. Ali, M., Khan, A., & Latif, M. (2022). In-memory malware execution: Fileless attack strategies. *Journal of Cybersecurity Research, 12*(1), 45–61.
4. Bashir, M., & Papadopoulos, P. (2023). HTTPS abuse in phishing: Trust indicators and user deception. *ACM Transactions on Privacy and Security, 26*(3), 91.
5. Bhardwaj, A., & Goundar, S. (2020). Keyloggers: Silent cyber security weapons. *Network Security, 2020*(2), 9–15.
6. Huang, B., & Felder, J. (2023). Securing DevOps: Software bill of materials in supply chain security. *ACM Transactions on Software Engineering, 42*(3), 65.
7. Iduh, B. N., Umeh, M. N., & Paul, R. U. (2024). Ethical keylogger solution for monitoring user activities. *World Journal of Advanced Engineering Technology and Sciences, 7*(5), 223–237.
8. Javed, A., & Silva, T. (2023). Simulating botnet attacks for cybersecurity education. *International Journal of Cybersecurity Pedagogy, 8*(1), 21–34.
9. Lee, T., & Mohammed, K. (2022). Ransomware-as-a-Service: Dissecting the underground economy. *IEEE Security & Privacy, 20*(4), 52–60.
10. Marczak, B., & Paxson, V. (2023). Understanding botnet infrastructure and control techniques. *IEEE Security & Privacy, 21*(2), 45–52.
11. Mourya, R., Patil, K., & Srivaramangai, R. (2024). Real-time keystroke monitoring with encryption. *International Journal of Science and Research, 13*(4), 1342–1347.
12. Orebaugh, A., & Allnutt, S. (2022). Anatomy of a phishing kit: Techniques and trends. *Journal of Cybersecurity Insights, 18*(2), 102–117.
13. Ryu, D., Park, Y., & Choi, J. (2023). Memory forensics for detecting fileless malware. *Digital Investigation, 44*, 101216.
14. Singh, V., Khan, F., & Rao, R. (2022). A review of botnet detection and disruption strategies. *Journal of Information Security and Applications, 64*, 103042.
15. Sun, L., Zhang, J., & Kim, S. (2023). Unveiling fileless attacks: Analysis of memory-resident threats. *IEEE Transactions on Dependable and Secure Computing, 20*(2), 198–213.
16. Tan, W., Clarke, N., & Zhou, H. (2023). Tracing RaaS through dark web intelligence. *Computers & Security, 123*, 102987.