XWorm RAT Technical Analysis (2024–2025 Variant)

Cybanetix

V1.0 27/06/2025

Cybanetix Limited
cybanetix.com
Registered at:
The Coade
Level 9
98 Vauxhall Walk
London
SE11 5EL
Company Registration: 10558582
VAT Number: GB 262502430

CYBANETIX

Table of Contents

XWorm RAT – Technical Analysis (2024–2025 Variant)

Background

XWorm is a remote access Trojan (RAT) first discovered in 2022. It is a versatile malware tool that enables attackers to steal sensitive information, gain remote control of systems, and even deploy additional payloads. XWorm is sold as malware-as-a-service on underground forums and via Telegram, making it easily accessible to cybercriminals. Its appeal lies in a wide range of dangerous features, ranging from typical RAT functions like surveillance and data theft to destructive actions like file encryption (ransomware behavior) and DDoS attacks. The malware's multifaceted nature has attracted both financially motivated actors and state-sponsored groups. In fact, multiple threat groups have adopted XWorm: for example, the North Korean APT Kimsuky has been observed using XWorm in recent campaigns, and cybercrime groups like NullBulge have leveraged XWorm alongside other RATs to ultimately deploy ransomware (LockBit). XWorm's continual development has yielded new versions (the latest known version is XWorm 6.0 as of mid-2025) that introduce enhanced stealth, persistence, and anti-analysis capabilities. The following report provides an in-depth technical breakdown of XWorm's latest variant – covering its infection chain, payload behavior, C2 infrastructure, obfuscation and anti-analysis techniques, persistence mechanisms, data exfiltration methods, and the TTPs associated with attackers deploying XWorm in 2024–2025.

Infection Chain and Payload Delivery

Initial Access

Attackers typically deliver XWorm to victims via social engineering and phishing lures. Campaigns often use malicious email attachments or links to initiate the infection. For instance, one observed tactic is sending a ZIP archive (disguised as a business document or other lure) that contains an obfuscated script loader. In recent cases, adversaries have employed unusual file formats like SVG images with embedded JavaScript to deliver XWorm: the SVG, when rendered (e.g. in a phishing page or a vulnerable viewer), triggers an embedded script that kicks off the infection chain. More commonly, XWorm campaigns have used malicious Windows Script Files (WSF), Visual Basic scripts (VBS), batch scripts (BAT), or LNK shortcuts as initial droppers. These droppers are heavily obfuscated to evade detection – for example, a malicious WSF

might include decoy text and a hidden VBScript segment with hex-encoded commands to appear benign to static scanners.

Multi-Stage Loader Execution

Once a user opens the booby-trapped file, a multi-stage execution chain unfolds. In one documented infection flow, a WSF dropper (delivered via phishing) executes a hidden VBScript that downloads a second-stage PowerShell script from a paste site (e.g. Paste.ee). Using trusted sites to host malicious code helps the attacker stay under the radar of network defenses. The PowerShell script in turn creates additional files and tasks: for example, it may drop an additional VBScript, a batch file, and another PowerShell module onto the system (often placing them in innocuous-looking directories like C:\ProgramData\Music\Visuals). It then sets up a scheduled task or registry autorun to persist and execute the next stage. In the cited case, the PowerShell created a scheduled task named "MicroSoftVisualsUpdater" to run the dropped VBScript after a short delay and repeatedly every 15 minutes. This VBScript triggers the BAT file, which then runs the final PowerShell loader script.

At the final stage, XWorm's core payload is loaded filelessly into memory. Rather than writing the RAT executable to disk, the loader (often a PowerShell or .NET-based stub) will reflectively load the XWorm payload or inject it into a legitimate process. For example, researchers observed a PowerShell loader script that stored the XWorm binary and a DLL injector ("NewPE2") as hex-encoded strings in its code (to avoid easy detection). This loader used .NET reflection to load the injector DLL from memory, then invoked its Execute method to inject the XWorm payload into a legitimate Windows process (in this case, RegSvcs.exe, a signed Microsoft .NET service process). As a result, XWorm began running under the context of a trusted process, with the original PowerShell script terminating to leave few obvious traces of malware in the process list. (Past variants have also employed process hollowing or similar code injection techniques to hide the malicious thread in system processes.)

In other campaigns, the infection chain may differ slightly – for example, Proofpoint reported a cluster of attacks in 2024 using Cloudflare "TryCloudflare" tunnels and WebDAV shares for staging malware. In those cases, phishing emails delivered a .URL shortcut file that, when clicked, connected to an attacker-controlled WebDAV server to retrieve a malicious LNK or VBS file. That file then executed a BAT/CMD script, which downloaded a Python installer and multiple Python scripts to eventually load the RAT payload. Regardless of delivery method, the common theme is a layered execution chain involving scripts and living-off-the-land binaries (e.g. wscript.exe for VBS, powershell.exe, cmd.exe, etc.) to download or construct the XWorm payload in memory, thereby avoiding direct disk writes of the final malware.

Exploitation of Known Vulnerabilities

In addition to phishing, attackers have occasionally exploited known code execution vulnerabilities to deploy XWorm. Notably, a wave of attacks in mid-2023 leveraged the "Follina" vulnerability (CVE-2022-30190) in Microsoft Office to drop XWorm. In those cases, malicious Office documents abused the MSDT protocol (the Follina exploit) to execute PowerShell code, which then pulled in XWorm as the final payload. This highlights that both social engineering and software vulnerabilities can serve as the initial exploitation vector for XWorm infections.

XWorm Payload Behavior and Capabilities

Command and Control (C2) Communications

Once running on a victim host, XWorm will establish contact with its command-and-control server. The RAT is usually configured with a hardcoded C2 address (domain or IP) and port, stored in its embedded configuration. In older versions, the C2 could be supplied via command-line arguments from the loader script, but in the latest variant (v6.0) the XWorm binary itself contains the C2 info encoded in its config. The config data is encrypted (for example, version 5.6 stored config values in Base64 encoded strings further encrypted with AES-ECB using a hardcoded key). On launch, XWorm decrypts its configuration to retrieve the C2 host and port, an encryption key, its campaign or group ID, and other settings. For instance, one sample's config decrypted to a domain ziadonfire[.]work[.]gd with port 7000, along with an AES key and version tag indicating "XWorm V5.6".

XWorm uses a direct socket connection to communicate with the C2. It typically opens a TCP socket to the configured IP/port (after resolving any domain). The malware is capable of handling binary data transmissions and maintains an interactive session with the C2. To keep the session alive, XWorm periodically sends heartbeat "ping" messages – for example, pinging the C2 every 10–15 seconds – and expects a corresponding "pong" from the server, checking for that response very frequently (multiple times per second). All communication is encapsulated in this socket channel, and may be further encrypted or encoded using the negotiated key from the config (XWorm's protocol uses the AES key for encrypting payload data and possibly for certain command content).

In some campaigns, actors have modified XWorm or used its builder to utilize alternate C2 channels. For example, a trojanized XWorm builder observed in late 2024 registered infected machines to a Telegram-based C2 using a hardcoded bot token. In that variant, each new victim would connect to a Telegram bot, which acted as the C2 server for receiving commands and exfiltrating data. This demonstrates the flexibility of XWorm's

C2 infrastructure – while the default is a typical host:port setup, the malware-as-a-service model allows buyers to configure custom C2 mechanisms (including public IM platforms or web APIs) when generating the payload. Indeed, XWorm's presence on darknet forums means multiple actors can run their own C2s globally, and some have been seen abusing services like Paste.ee, Discord webhooks, or Cloudflare Tunnels for staging and C2 relay.

Once connected, XWorm usually begins by sending a reconnaissance payload to the server. It gathers system information to identify the infected machine to the attacker. Data collected typically includes the hostname, current username, OS version, hardware info (CPU, GPU), the privilege level (whether admin), presence of antivirus, and even a specific marker for USB drives or files (one config included a value "USBNM USB.exe"). This info is concatenated with a delimiter string and the XWorm version, then transmitted to the C2. Some versions generate a unique victim identifier (e.g., by hashing certain system info) to tag the machine in the C2 database. After this initial beacon, the RAT enters a loop awaiting commands from the attacker.

Remote Access Functions and Modules

XWorm's latest variant offers a comprehensive arsenal of RAT capabilities, often configurable through its builder. Analysis of version 5.6 and 6.0 reveals an extensive list of supported commands (dozens in total), enabling the attacker to perform virtually any action on the compromised system. Key payload functionalities include:

- XWorm can spy on the victim's activities and steal data. It can record keystrokes (a "keylogger" command captures everything typed), capture screenshots of the desktop, and harvest credentials from web browsers (stealing saved passwords, cookies, and autofill data). In one campaign, the malware automatically stole Discord authentication tokens and gathered the system's public IP-based geolocation as soon as it infected a machine. All stolen data is exfiltrated back to the C2 (for example, screenshots are encoded to JPEG and sent via the socket channel).

- The RAT supports commands to manipulate files on the victim. It can download and execute files from the internet or C2 (LN command for "download and run"), and can upload specified files from the victim to the C2 (exfiltration on demand). It also has an uninstall command to delete its own binaries and artifacts from the system when instructed – useful for covering tracks. Notably, XWorm introduced functionality to manage plugins: it can load additional modules (DLLs or code for extra features) and also remove those plugins and their traces from the registry when commanded (the RemovePlugins command cleans up stored plugin info to wipe evidence). This modular design means XWorm's functionality can be extended with custom plugins (e.g., for specific network scanning,

credential dumping, etc.), and attackers can later remove them to reduce footprint.

- XWorm enables remote command execution and control of processes. It can spawn a hidden remote shell (RunShell) to run arbitrary system commands in the background. There are commands to run specific PowerShell scripts or shellcode: for instance, a DW command writes a provided PowerShell snippet to a temp file and executes it, and an FM command executes a compressed Base64-encoded command given by the attacker. XWorm can also open URLs on the victim's browser (visible or hidden from the user) for further exploitation or surveillance. Standard host control commands include shutting down, restarting, or logging off the system remotely (PCShutdown, PCRestart, etc.).
- While XWorm does not inherently self-propagate, it provides tools that could facilitate lateral movement by an adversary. For example, it can enumerate running processes and report back which processes are active (StartReport command) – an attacker could use this to identify security tools or pivot targets. Using its shell access, an operator could deploy credential-dumping tools or leverage Windows commands to move laterally. XWorm's ability to modify the Windows Hosts file is an interesting feature that could aid post-exploitation; it can read the hosts file and send it to the attacker, or replace it with an attacker-supplied version (Hosts and Shosts commands). By poisoning the hosts file, attackers might reroute network traffic or spoof domains as part of a broader attack on the local network.
- The malware itself does not exploit privilege escalation vulnerabilities, but it does check and take advantage of high privileges if available. If XWorm is running with administrator rights, the latest version will attempt to mark its process as a critical system process. This is done by enabling debug privileges (SeDebugPrivilege) and calling Windows APIs to flag the process as critical. A critical process cannot be terminated by regular means; if an admin user force-kills it, the system will crash (Blue Screen). Thus, by making itself critical, XWorm ensures it can't be easily killed without drastic consequences – a form of self-protection. After a crash/reboot, XWorm would restart via persistence to continue execution. In terms of obtaining admin in the first place, attackers using XWorm may employ social engineering (e.g., UAC prompts) or other tools to elevate privileges prior to deploying the RAT. The Kimsuky APT, for instance, made use of living-off-the-land binaries and scripts that could disable Windows logging and perform other tasks requiring admin rights, indicating they had or obtained elevated privileges during their multi-stage attack.
- Unusually for a RAT, XWorm can launch Denial-of-Service (DoS/DDoS) attacks from the infected host. A command StartDDos causes the victim machine to

begin flooding a target IP and port with HTTP POST requests every few seconds (with random fake user-agent strings). The attacker can specify the target and duration of the attack, effectively using the compromised system as part of a botnet for DDoS. Another harmful capability found in some XWorm versions is the ability to encrypt files on the victim system on command. The trojanized builder observed by CloudSEK researchers exposed commands like /encrypt <password> which would instruct XWorm to encrypt all files on the host with a provided password. This effectively turns XWorm into a rudimentary ransomware if the attackers choose to invoke it. Combined with its data theft features, it could be used for "double extortion" (stealing data and then encrypting files). In general, these features illustrate that XWorm is not only an espionage tool but can also facilitate sabotage and monetization (whether via ransomware or forced participation in DDoS).

Persistence Mechanisms

Maintaining persistent access to an infected system is a priority for XWorm operators, and the malware provides several methods to achieve this. The exact persistence mechanism can be selected by the attacker when building the payload. The XWorm builder offers options such as installing via registry Run keys, creating scheduled tasks, or copying itself to the Startup folder, among others. In earlier versions, a common technique was to set up a Scheduled Task that periodically re-launches the malware or loader script (as seen with tasks like "MicrosoftVisualsUpdater" running the Visual Basic script every 15 minutes). The latest observed variant (v6.0) used a simpler approach: it writes a copy of its dropper script (e.g. update.vbs) into both the %TEMP% and %APPDATA% directories, then creates a Registry Run key pointing to those locations. By adding entries under HKCU\Software\Microsoft\Windows\CurrentVersion\Run (or the equivalent HKLM path for all users), XWorm ensures that its script will execute on every user logon, thus reloading the RAT in memory.

This dual-path persistence (Temp and AppData) provides redundancy – even if one copy is removed, the other might still execute. It differs from the previously observed sample that exclusively relied on a scheduled task. The shift indicates attackers are experimenting with what is stealthier or more reliable in practice. Some XWorm infections have also leveraged LOLBAS (Living-Off-the-Land Binaries and Scripts) for persistence; for example, utilizing powershell.exe with the -WindowStyle Hidden flag and encoded commands in a registry entry, or abuse of wscript.exe to run a script on startup. The Kimsuky campaign in 2025 demonstrated a fileless persistence by staying entirely in PowerShell/memory and using encoded scripts that re-trigger via system mechanisms. In the CloudSEK-documented builder attack, XWorm established persistence by performing specific Windows Registry modifications (likely the Run key

method) as soon as it confirmed it wasn't in a virtualized environment. In summary, XWorm's persistence can vary, but common tactics are registry autoruns, scheduled tasks, startup folder copies, and occasionally WMI or service installation if the attacker so chooses. These methods ensure that even if the system reboots or the user logs off, the malware will regain execution and reconnect to its C2, maintaining the adversary's foothold.

Obfuscation and Evasion Techniques

XWorm's developers have put significant effort into obfuscation and anti-analysis to evade detection by security products and researchers. From the initial dropper stages through to the in-memory payload, the malware employs multiple layers of hiding and checks:

- The first-stage droppers (VBS, BAT, etc.) are typically highly obfuscated. For example, a XWorm v6 infection started with a VBScript that stored an array of character codes which, when iterated in reverse and converted via ChrW, produced the actual malicious script at runtime. This runtime generation of code, combined with use of Execute/Eval on the fly, thwarts simple static analysis. Batch scripts observed in 2025 campaigns were also convoluted – containing junk data or encoded payloads hidden in comments (using ::: as seen in one PowerShell-loader-in-BAT strategy). The use of fileless techniques is a core evasion strategy: rather than dropping an obvious .EXE file to disk, XWorm is often loaded in memory via PowerShell's Invoke-Expression or .NET reflection. This means traditional file-scanning antivirus might never see a standalone malware file to flag. As one report noted, the XWorm binary (XClient3.exe) was detected by many AV engines on VirusTotal, so the attackers simply chose *not to write it to disk at all*, making detection much harder.
- XWorm stages make heavy use of legitimate system tools and trusted external services to blend in. The malware's infection chains often leverage PowerShell, Windows Script Host (wscript), and trusted cloud services. By storing second-stage scripts on Paste.ee, GitHub, or cloud storage, the malware traffic appears similar to normal user or system activity (accessing a pastebin or a GitHub URL). The Proofpoint-observed campaigns abused Cloudflare tunneling to make C2 communications appear as if they were going to Cloudflare infrastructure, a trusted domain. These techniques help XWorm bypass network filters and make incident response more difficult.
- To evade detection by host-based defenses, newer XWorm variants implement direct anti-analysis patches in memory. A prime example is the Antimalware Scan Interface (AMSI) bypass introduced in XWorm 6.0's PowerShell loader. Before loading the RAT, the PowerShell script searches the running process's

memory for the DLL CLR.dll (the .NET Common Language Runtime) and the string AmsiScanBuffer within it. Upon finding that signature, it overwrites the AMSI scan function with null bytes (or a simple return instruction), effectively disabling AMSI's ability to scan any further script content. This means any malicious code loaded afterward won't be inspected by antivirus hooks that rely on AMSI. Additionally, some XWorm loader scripts disable Windows Event Tracing and logging. For instance, an analyzed batch/Powershell loader in 2025 used .NET interop to locate and patch EtwEventWrite (the Event Tracing API), neutering Windows' ability to record events for the malicious process. The Kimsuky APT's use of XWorm also involved executing PowerShell with commands to disable Windows Event Logging altogether, likely via registry or policy changes, as a defense evasion step during infection.

- XWorm takes steps to detect if it's running in an analysis environment. The latest variant notably terminates itself on finding Windows XP as the OS. This might seem counter-intuitive (as XP is outdated), but many sandbox environments and malware analysis labs use Windows XP or similarly old systems for detonating malware. By shutting down on XP, XWorm avoids running in those sandbox traps. Another check implemented is for virtualization or researcher network indicators: XWorm v6 uses an AnyRun-labeled function that actually queries an online API (ip-api.com) to check the victim's IP address category. If the IP is recognized as belonging to a cloud data center or known hosting provider (common for sandbox services like Any.Run, VirusTotal, etc.), XWorm assumes it's being analyzed and will self-terminate. Earlier observed versions (e.g., the trojanized builder variant) similarly inspected the Windows Registry for keys associated with virtual machines (like VMware/VirtualBox artifacts) and would abort if a VM is detected. These checks help XWorm avoid executing its full routine under scrutiny, thus flying under the radar of automated analysis systems.
- By injecting into legitimate processes (such as RegSvcs.exe, msiexec.exe, or other common Windows processes), XWorm obscures its presence. The injected code runs under the memory space of a trusted binary, making it harder for defenders to spot a rogue process. Some variants have been seen using process hollowing – spawning a process in suspended state and replacing its code with the malware – to hide in plain sight. Running within a Windows system process also grants the malware some free pass with firewall or EDR behavior monitoring, as those processes are often whitelisted for certain operations.
- Attackers using XWorm also employ tricks to ensure the user doesn't notice anything suspicious during infection. Kimsuky's PowerShell loader, for example, embedded C# code to call the Win32 API ShowWindow on various windows to

hide any console or script windows that might briefly appear. The PowerShell was run with -WindowStyle Hidden and other flags (-nop -w hidden -enc …) to ensure no visible artifacts alert the victim. Additionally, many campaigns display decoy documents or files to the victim as a smokescreen. For instance, while XWorm installs in the background, a benign PDF document might open (as was the case in the Cloudflare tunnel campaigns) to convince the user that the email attachment was harmless or just didn't do anything significant. Meanwhile, the malware executes behind the scenes.

Indicators of Compromise and Analysis Artifacts

Analyzing XWorm infections yields several Indicators of Compromise (IOCs) and notable artifacts that defenders can watch for:

- Although XWorm tries to stay memory-resident, the initial stages do leave some traces on disk. Look for unusual script files in user directories. For example, XWorm droppers have used file names like update.vbs (often placed in %TEMP% or %APPDATA%), or random-named PowerShell scripts such as wolf-8372-4236-2751-hunter-978-ghost-9314.ps1 (as seen in one v6.0 infection chain). The presence of multiple script files with names referencing "Labs" or random words (e.g. VsLabs.vbs, VsEnhance.bat, VsLabsData.ps1 in one case) in odd folders like C:\ProgramData\Music\Visuals can be an IOC. Any scheduled task named similarly to "MicrosoftVisualsUpdater" or containing "Visuals" in its actions might indicate this malware's persistence. Registry run keys pointing to *.vbs or *.bat files in Temp/AppData are another red flag, especially if the value name is innocuous (XWorm may use names like "Update" or others to blend in).
- XWorm C2 domains and IPs vary by campaign, but a notable indicator is unusual TCP connections to high-numbered ports (e.g., port 7000 as in one config, though ports can change). Since XWorm often uses direct sockets, the traffic might not use HTTP(S) and could appear as unknown TCP streams in network logs. Some identified C2 addresses from 2024–2025 campaigns include: ziadonfire.work[.]gd (domain), and IPs like 185.235.128.114 and 92.119.114.128 used by Kimsuky's campaign. These addresses were seen delivering or controlling XWorm payloads and should be considered indicative if found in logs. Additionally, XWorm's ping/pong traffic might be observable: the malware pings the server every few seconds, so repeated small TCP packets to a fixed destination at ~10-second intervals could hint at a beacon. In some XWorm variants (like the builder backdoor), the C2 was a Telegram bot – meaning the infected host would connect to Telegram's servers or API endpoints. Network connections to Telegram domains or IPs by systems that don't normally use Telegram could be a clue.

- If analyzing a sample statically or in memory, certain unique strings or behaviors can reveal XWorm. For example, the config decryption routine in one version used the string rZ2W67345HrmrYRB (an encryption key seed) – seeing this in a binary or script is a strong indicator. XWorm's commands often contain identifiable keywords. The CloudSEK analysis showed commands prefixed by a machine ID and an asterisk, followed by words like browsers, keylogger, desktop, encrypt, etc., as the action. Finding such patterns in malware traffic or memory (e.g., /12345*keylogger) suggests XWorm or a similar RAT is present. XWorm also tends to create or seek out certain mutexes or file markers for single-instance control – while specific mutex names haven't been publicized in our sources, behavioral analysis tools have noted XWorm checking for previously run instances (this could be via a mutex or registry flag).
- Observing a process that calls ip-api.com to check its IP or one that terminates itself when detecting certain OS or virtual machine artifacts could indicate XWorm. For instance, if a malware sample runs and immediately exits on Windows XP but not on Windows 10, it's employing a trick that XWorm v6 used. Also, any process that attempts to modify the memory of CLR.dll at runtime to patch AMSI (AmsiScanBuffer) is exhibiting behavior identical to XWorm's loader. Such behavior can be caught by advanced EDR solutions that monitor memory patching.
- Security teams have successfully extracted hardcoded tokens and kill-switches from at least one XWorm variant to disrupt its botnet. In that case, researchers found an embedded kill-switch that they activated to uninstall the malware from many victim machines, highlighting that thorough reverse engineering of XWorm can reveal countermeasures to aid defenders.

Tactics, Techniques, and Procedures (TTPs) of XWorm Attackers (2024–2025)

Attackers leveraging XWorm have demonstrated a range of TTPs across the kill chain, from initial compromise to lateral movement and defense evasion. Below is a summary of the key tactics observed in 2024–2025 campaigns involving XWorm:

- Phishing is the predominant initial access vector. Threat actors send targeted emails with either malicious attachments (Office documents, compressed archives, HTML/ZIP packages, etc.) or links to download a payload. They often craft lures relevant to the victim (e.g. invoice themes, package delivery notices, or even topics in multiple languages). In many cases, the email contains a malicious shortcut file (.LNK) or an HTML/URL file that leads to a drive-by

download. For example, one campaign enticed users to click a .URL file that would connect to a file share and retrieve a hidden LNK dropper. Another ploy has been using OneNote files or PDFs with embedded scripts to launch the infection (taking advantage of the trust in those file types). Furthermore, some advanced campaigns exploit vulnerabilities at this stage – as mentioned, the use of the Follina exploit to directly run PowerShell ensured that simply opening a rigged Office document could inject XWorm without requiring macro enablement. Across these methods, social engineering remains crucial: threat actors often accompany the malware delivery with decoy content (e.g., a harmless document or image that opens to avoid raising suspicion while malware installs in background).

- Upon gaining code execution on the victim's machine, attackers frequently use Living-off-the-Land techniques to run payloads. PowerShell is a favorite – in XWorm campaigns, encoded PowerShell commands execute the next stages (e.g., downloading payloads, injecting the RAT). Attackers employ encoded scripts, Base64 blobs, and LOLBAS like rundll32, regsvcs, or WMI to launch the malware in a stealthy way. Defense evasion at this stage is paramount: scripts are obfuscated (hex encoding, string concatenation, junk code), Windows AMSI and logging are disabled or bypassed (patching AMSI via PowerShell, turning off Event Tracing), and any visible windows or prompts are suppressed (using -WindowStyle Hidden, ShowWindow API, etc.). Fileless execution is leveraged to great effect – by injecting XWorm into memory or into a legitimate process, the attackers avoid dropping an executable that could be caught by antivirus. Additionally, packers or crypters may be used if an executable is written to disk at all. Some XWorm samples are packed or wrapped in layers of encryption to stymie analysis. Notably, attackers sometimes deploy decoy techniques for evasion: Kimsuky, for example, downloaded harmless PDF documents for the user to open, even as the malware quietly installed in the background. This misdirects the victim's attention.
- XWorm itself doesn't automatically escalate privileges, but attackers using it have shown adeptness at gaining elevated privileges when needed. If the initial code execution runs in user context, attackers might attempt a UAC bypass or use a known local exploit to get admin rights (though specific exploits weren't detailed in public sources for these campaigns). In the Kimsuky campaign, the use of certain PowerShell operations (like disabling logs) implies they had admin rights or had compromised a high-privilege account. Once they have admin, XWorm can enable its critical process protection feature, as discussed, to further cement its position. In some cases, simply convincing the user to run the initial file (e.g., a fake installer) with admin privileges via a prompt is sufficient – the trojanized XWorm builder likely required victims (the "script kiddie" hackers)

to run the builder with admin, unwittingly giving the malware full control over their systems. Post-compromise, if lateral movement is intended, privilege escalation to domain admin or similar might be pursued using credential dumping tools (which could be executed via XWorm's shell access).

- Attackers deploying XWorm often persist in a way that suits their campaign's needs. Commodity deployments aiming for broad infections might simply use registry Run keys or Startup shortcuts for persistence (quick and works on most user machines). More covert operations (espionage-driven) might prefer scheduled tasks that look legitimate or even WMI Event subscriptions for stealth persistence (though the latter wasn't explicitly observed for XWorm, it's a known technique in similar RAT campaigns). In 2025, some campaigns chose short-term persistence – for example, the CloudSEK-documented attacker had a kill-switch they later activated, indicating they didn't intend to stay indefinitely on all 18,000 infected machines. However, APT groups like Kimsuky aim for long-term stealthy access; thus, their XWorm usage was accompanied by careful cleanup (like using fileless techniques and likely removing obvious indicators once access was established). The availability of multiple persistence methods in XWorm's builder means the TTP can vary widely. Investigators have to check registry, scheduled tasks, services, startup folders, and even odd mechanisms like Office VBA autoruns or shell autorun keys, since any could be used.
- While concrete public examples of XWorm being used for lateral movement are sparse (likely because many XWorm deployments are standalone infections on end-user systems), the potential is certainly there. An attacker with a XWorm foothold can leverage native Windows tools to spread. For instance, they might collect credentials (with the RAT's keylogger or by stealing password files) and then use those to authenticate to other machines via RDP or SMB. They could deploy XWorm to another host by using its file transfer capabilities (upload command to drop a copy on a network share, etc.) and then use WMI or PsExec to run it remotely. If the target is in an Active Directory environment, the attackers might use commands via XWorm's shell to run PowerShell Remoting or schedule tasks on other hosts. Although not attributed to a specific APT publicly, it is reasonable that a determined actor would integrate XWorm into a larger toolkit for lateral movement – using it to execute network reconnaissance commands (net view, net use, whoami, etc.), then pivot. One of XWorm's internal commands (StartReport) can accept a list of process names and report which are running; an imaginative use of this could be to check if remote management services or tools are present, aiding the lateral movement strategy. In summary, lateral movement with XWorm is a manual (attacker-driven) process, not an automated worm, but XWorm gives the attacker all the remote control needed to perform it.

- Perhaps the most significant TTP category for XWorm operators is defense evasion. Beyond the technical evasion measures already described (obfuscation, anti-AV, anti-sandbox), attackers also practice operational security. For example, they often rotate infrastructure – the Proofpoint report noted that the threat actor using Cloudflare tunnels kept modifying aspects of their chain to improve evasion, and did not stick to one static infrastructure. XWorm-related campaigns also frequently encrypt or password-protect their payloads and archives to defeat perimeter scanning. Kimsuky's campaign used password-protected RAR archives and staged payloads with benign extensions (e.g. .txt for PowerShell scripts) to slip past filters. We also see "garbage" code injection as a technique – some XWorm droppers include large blocks of irrelevant text or data (for example, a long comment about Social Security Administration in a WSF file) to confuse analysts and automated scanners. On the anti-forensics side, XWorm has a command to self-delete (uninstall) and can remove plugin traces. When attackers are ready to exit, they can attempt to wipe out the malware from the host. In the case of the script kiddie botnet, the threat actor actually used a kill-switch to uninstall XWorm from many victims proactively once discovered. This indicates some attackers build in a contingency to burn their tools if needed. Finally, the use of Telegram for C2 in that case also doubles as an evasion – blending C2 traffic with normal encrypted chat traffic and making attribution harder (since commands were sent through a bot API rather than a traditional C2 panel).

Recent Notable Campaigns Involving XWorm (2024–2025)

Several high-profile campaigns and threat actor activities in 2024–2025 have involved XWorm RAT:

- Kimsuky APT's PowerShell Spy Campaign (2025): Kimsuky, a North Korean state-sponsored group, leveraged XWorm in a sophisticated espionage campaign in early 2025. The attack used *PowerShell-based fileless techniques* almost exclusively. It began with highly obfuscated, Base64-encoded PowerShell loader scripts that sequentially pulled in additional components. The attackers downloaded RAR archives and multiple payload binaries (with names like orwartde.exe, eworvolt.exe, enwtsv.exe) along with further PowerShell scripts masquerading as text files. These were retrieved from Kimsuky-controlled IPs (e.g. 185.235.128.114, 92.119.114.128) that doubled as C2 servers. One notable tactic was the use of inline C# in PowerShell to call ShowWindow and hide any console windows, keeping the attack invisible to the user. Kimsuky also

employed decoy PDF documents – while the victim was distracted with a harmless PDF opened on their screen, XWorm components executed in the background. They utilized ExecutionPolicy Bypass to run their scripts and even disabled Windows event logging to avoid detection and auditing. Persistence was achieved via *living-off-the-land* methods (likely registry or scheduled tasks created through PowerShell). Ultimately, the campaign focused on data exfiltration and covert remote access: XWorm was used to maintain access and siphon off data (such as keystrokes, documents, credentials) back to the C2. The use of XWorm by Kimsuky is significant, as it shows even APT groups with custom toolsets may incorporate commodity RATs to expedite operations or reduce development effort. Kimsuky's adoption of XWorm also underscores that XWorm had the necessary capabilities (stealth, data theft, control) to satisfy nation-state espionage requirements.

- Cloudflare Tunnel Campaigns (Mid-2024): Proofpoint tracked an unnamed cybercriminal threat cluster in 2024 that delivered XWorm at scale using Cloudflare tunnels for distribution. The actor would send phishing emails (including in English, French, Spanish, German) with links or attachments that ultimately led to One-Time Cloudflare Tunnels (via the trycloudflare service) to host their malware delivery. This innovative approach meant victims connected to a Cloudflare domain (which appeared benign) that proxied to the attacker's server. The infection chains predominantly dropped XWorm by June/July 2024, though earlier they also delivered other RATs like AsyncRAT, VenomRAT, Remcos, etc.. The multi-stage chain was complex: victims who clicked the link often got a .URL file or similar, which fetched a malicious LNK; the LNK ran a script that downloaded a Python environment and executed Python scripts to load the final RAT. In many cases a benign PDF decoy was displayed (business-themed, like invoices or tax documents) to the user while the malware installed. These campaigns were high-volume – some waves hit thousands of organizations, indicating a broad targeting typical of financially motivated actors. The use of Cloudflare's infrastructure aided their defense evasion and global reach. XWorm, being a lightweight and powerful payload, was the final stage used to remotely control infected machines for theft or further monetization (potentially selling access, etc.). Proofpoint noted the threat actors continually tweaked their TTPs (for example, increasing obfuscation in later waves, adding more stages) to avoid detection. This campaign demonstrates how XWorm has been weaponized in *"Malware-as-a-Service" fashion*—deployed en masse by cybercrime crews who value its flexibility.
- NullBulge "Hacktivist" Group (2024): NullBulge is a cybercriminal group that emerged in 2024 masquerading as hacktivists (with anti-AI rhetoric) but engaging in data theft and ransomware for profit. SentinelOne reported that NullBulge

targeted AI/gaming communities and even leaked some stolen data from a major company. Notably, NullBulge's toolset included XWorm and AsyncRAT as stage-one implants, which they used to take control of victims' systems, before ultimately deploying LockBit ransomware (using a leaked builder) on those systems. Their distribution methods included poisoning software supply chains – e.g., inserting malicious code into GitHub repositories, Python libraries, and software mods that AI developers or gamers would download. These trojanized components would drop RATs like XWorm on the machines of unsuspecting users in the AI/art community. Once the RATs phoned home, NullBulge operators could perform reconnaissance, steal sensitive data (such as browser credentials and system info via custom Python scripts that worked alongside XWorm), and then decide whether to deploy ransomware. In effect, XWorm served as a backdoor for post-exploitation; it provided access and persistence while the attackers prepared their final payload (encryption). NullBulge's use of XWorm underscores the RAT's role as a general-purpose tool in multi-stage financially motivated attacks. It also illustrates XWorm's compatibility with various delivery vectors – even less traditional ones like open-source software tampering.

- Trojanized XWorm Builder Targeting Criminals (Late 2024): In a twist on the typical scenario, one threat actor created a fake XWorm builder and released it to infect other would-be hackers. This was documented by CloudSEK and publicized in January 2025 after being active in late 2024. The malicious actor advertised a free or cracked "XWorm RAT Builder" on platforms like GitHub, file-sharing sites, Telegram channels, YouTube tutorials, etc., appealing to novice hackers who wanted to use XWorm without paying for it. Instead of generating a RAT payload, this trojanized builder silently installed XWorm on the user's own machine. The result was a botnet of nearly 18,500 infected "script kiddies" (many in Russia, US, India, etc.) who thought they were setting up a hacking tool, but became victims themselves. The XWorm variant used in this scheme had some unique C2 characteristics: it registered bots to a Telegram C2 (with a hardcoded bot ID/token) and used Telegram channels to issue commands. It was configured to automatically steal data of interest from the infected wannabe hackers – Discord tokens, saved browser passwords, IP-based location – as soon as it ran. Interestingly, this XWorm instance had capabilities tailored to stealing from hackers: for example, it took screenshots of their desktops (perhaps to gather any confidential projects or notes) and had a command to encrypt files, possibly to hold these amateur hackers for ransom or just sabotage them. The actor behind it eventually triggered a kill-switch that uninstalled the malware from many victims (likely to avoid too much attention or out of fear of exposure once CloudSEK got involved). This campaign is notable for its scale and creativity, and it highlights some static IOCs: the use of /machine_id* command syntax in

Telegram (e.g. /12345*keylogger), and the anti-VM checks (the malware halting if it detected a VM in the registry). It also underlines that there is no honor among thieves – even threat actors can fall prey to XWorm when lured by illicit tools.

- Other State-Sponsored Usage: Outside of Kimsuky, there are indications that Chinese-affiliated threat actors have dabbled in XWorm. The SOCRadar Labs reported an attack wave where XWorm was used to exploit the Follina vulnerability, suggesting a Chinese government-sponsored operation given Follina's extensive use by Chinese APTs. While details are sparse, this likely involved spear-phishing targets with malicious Office documents exploiting CVE-2022-30190 to load XWorm, thereby establishing persistent espionage access. Additionally, TA558, a threat group noted in Netskope's report, allegedly utilized XWorm in early 2024. TA558 is known for targeting the hospitality and travel sectors (previously using RATs like AsyncRAT and Loda); their inclusion suggests XWorm was seen in phishing campaigns aimed at hotels or travel agencies, possibly for financial fraud or data theft. These examples reinforce that XWorm has entered the toolset of various APT and cybercrime groups across different motivations.

Conclusion

The latest variant of XWorm (v6.0) represents an evolution of an already potent malware family, incorporating new stealth features (like AMSI bypass via CLR patching and critical process protection) and maintaining a flexible, modular toolkit for attackers. Technically, XWorm demonstrates a modern RAT's playbook: multi-stage fileless delivery, robust RAT capabilities (spanning espionage to sabotage), and layered obfuscation and anti-analysis to challenge defenders. In 2024–2025, XWorm has been a fixture in both targeted APT campaigns and mass-scale criminal operations, a testament to its effectiveness and ease of use.

For defenders, detecting XWorm requires vigilance across multiple kill chain phases – from identifying suspicious script execution and LOLBAS use, to monitoring network beacons and unusual registry persistence. Indicators such as those outlined (strange temporary VBS files, repetitive C2 pings, AMSI patch behavior, etc.) can help flag an XWorm infection. Given XWorm's ability to run completely in memory and blend with legitimate processes, a combination of endpoint monitoring (EDR) and network analytics is necessary for coverage. Furthermore, threat intelligence on XWorm's infrastructure (e.g., known C2 domains or Paste sites) and hunting for its known techniques (like PowerShell disabling AMSI) will improve detection and response.

XWorm's ongoing development and its adoption by disparate threat actors suggest it will continue to be a pervasive threat. Security teams should ensure they harden systems against the typical delivery methods (educating users on phishing, patching exploits like Follina, restricting execution of unknown scripts) and have controls in place to catch behaviors characteristic of XWorm's modus operandi. As this report illustrates, XWorm is a prime example of the modern RAT – stealthy, feature-rich, and employed in creative ways by attackers – and it will require equally creative defense strategies to counter its latest variants.