Turla: A Galaxy of Opportunity

Warning: An indicator within this post contains offensive language.

In September 2022, Mandiant discovered a suspected Turla Team operation, currently tracked as UNC4210, distributing the KOPILUWAK reconnaissance utility and QUIETCANARY backdoor to ANDROMEDA malware victims in Ukraine. Mandiant discovered that UNC4210 re-registered at least three expired ANDROMEDA command and control (C2) domains and began profiling victims to selectively deploy KOPILUWAK and QUIETCANARY in September 2022.

ANDROMEDA was a common commodity malware that was widespread in the early 2010's. The particular version whose C2 was hijacked by UNC4210 was first uploaded to VirusTotal in 2013 and spreads from infected USB keys. Mandiant Managed Defense continues to observe ANDROMEDA malware infections across a wide variety of industries, however, Mandiant has only observed suspected Turla payloads delivered in Ukraine.

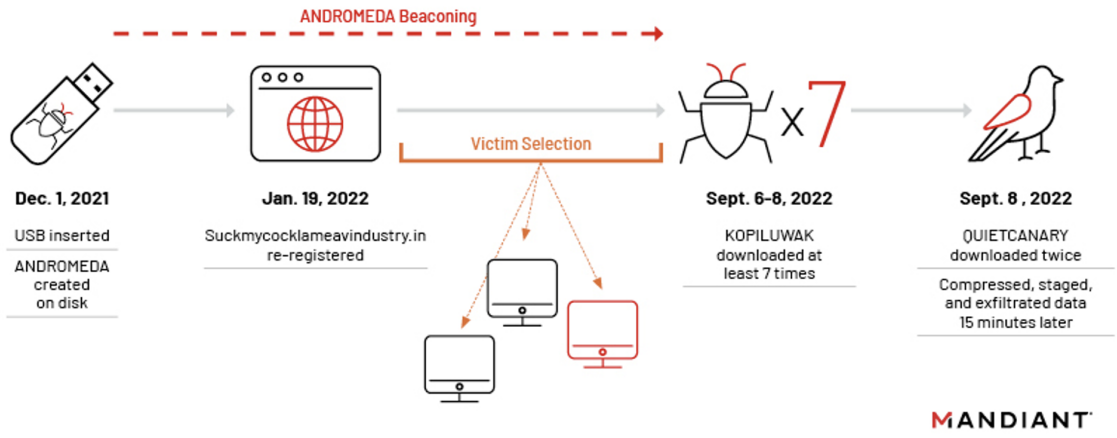

Figure 1: Timeline of ANDROMEDA to Turla Team intrusion

USB Spreading

As Mandiant recently wrote about in our blog post, Always Another Secret: Lifting the Haze on China-nexus Espionage in Southeast Asia, USB spreading malware continues to be a useful vector to gain initial access into organizations. In this incident, a USB infected with several strains of older malware was inserted at a Ukrainian organization in December 2021. When the system's user double clicked a malicious link file (LNK) disguised as a folder within the USB drive, a legacy ANDROMEDA sample was automatically installed and began to beacon out.

ANDROMEDA or 2013 Wants Its Malware Back

The version of ANDROMEDA that was installed to C:\Temp\TrustedInstaller.exe (MD5: bc76bd7b332aa8f6aedbb8e11b7ba9b6), was first uploaded on 2013-03-19 to VirusTotal and several of the C2 domains had either expired or been sinkholed by researchers. When executed, the ANDROMEDA binary established persistence by dropping another ANDROMEDA sample to C:\ProgramData\Local Settings\Temp\mskmde.com (MD5: b3657bcfe8240bc0985093a0f8682703) and adding a Run Registry Key to execute it every time the system user logged on. One of its C2 domains, "suckmycocklameavindustry[.]in," which had expired, was found to be newly re-registered on 2022-01-19 by a privacy protected registrant using Dynadot as the registrar. UNC4210 used this C2 to profile victims before sending the first stage KOPILUWAK dropper if the victim was deemed interesting.

Mandiant identified several different hosts with beaconing ANDROMEDA stager samples. However, we only observed one case in which Turla-related malware was dropped in additional stages, suggesting a high level of specificity in choosing which victims received a follow-on payload. During the time Mandiant monitored the C2s used to deliver the next stage payloads, the servers only remained up for a short period of a few days before going offline for several weeks at a time.

Recon with Ol' Reliable KOPILUWAK

After several months of ANDROMEDA beaconing without any significant activity observed, UNC4210 downloaded and executed a WinRAR Self-Extracting Archive (WinRAR SFX) containing KOPILUWAK (MD5: 2eb6df8795f513c324746646b594c019) to the victim host on September 6, 2022. Interestingly, the attackers appeared to download and run the same WinRAR SFX dropper containing KOPILUWAK seven times between September 6 and September 8. Each time the KOPILUWAK cast its net, it attempted to transfer significant amounts of data to the C2 manager.surro[.]am. It is unclear why UNC4210 did this as the profiling commands are hard coded in KOPILUWAK and would not yield different sets of data from the same host.

KOPILUWAK is a JavaScript-based reconnaissance utility used to facilitate C2 communications and victim profiling. It was first reported publicly by Kaspersky and has been tracked by Mandiant since 2017. Historically, the utility has been delivered to victims as a first-stage malicious email attachment. This is consistent with Turla's historical reuse of tools and malware ecosystems, including KOPILUWAK, in cyber operations.

The ANDROMEDA injected process "wuauclt.exe" made a GET request to "yelprope.cloudns[.]cl" with the target URL "/system/update/version." yelprope.cloudns[.]cl is a ClouDNS dynamic DNS subdomain which was previously used by ANDROMEDA and was re-registered by UNC4210. The ANDROMEDA injected process then downloaded and executed a WinRAR SFX containing KOPILUWAK to C:\Users\[username]\AppData\Local\Temp\0171ef74.exe (MD5: 2eb6df8795f513c324746646b594c019). Notably, this filename format has also been observed being utilized in Temp.Armageddon operations. Upon execution, the self-extracting archive created and executed KOPILUWAK from C:\Windows\Temp\xpexplore.js (MD5: d8233448a3400c5677708a8500e3b2a0).

In this case, UNC4210 used KOPILUWAK as a "first-stage" profiling utility as KOPILUWAK was the first custom malware used by this suspected Turla Team cluster following ANDROMEDA. Through KOPILUWAK, UNC4210 conducted basic network reconnaissance on the victim machine with whoami, netstat, arp, and net, looking for all current TCP connections (with PID) and network shares. The attackers also checked the logical disks and list of current running processes on the machine. Each command result was piped into %TEMP%\result2.dat, before being uploaded to KOPILUWAK's C2 "manager.surro[.]am" via POST requests.

QUIETCANARY in the Mine

Two days after the initial execution of and reconnaissance performed with KOPILUWAK, on September 8, 2022, Mandiant detected UNC4210 download QUIETCANARY to a host twice, but only executing commands through it on the second time. QUIETCANARY is a lightweight .NET backdoor also publicly reported as "Tunnus" which UNC4120 used primarily to gather and exfiltrate data from the victim. Please see the QUIETCANARY analysis in the annex for technical details regarding the malware.

Following the extensive victim profiling by KOPILUWAK, the ANDROMEDA injected process "wuauclt.exe" made a GET request to "yelprope.cloudns[.]cl" with the target URL "/system/update/cmu", which downloaded and executed QUIETCANARY. QUIETCANARY (MD5: 403876977dfb4ab2e2c15ad4b29423ff) was then written to disk.

UNC4210 then interacted with the QUIETCANARY backdoor, proceeding to utilize QUIETCANARY for compressing, staging, and exfiltrating data approximately 15 minutes later.

Data Theft

Mandiant observed interactive commands sent to and executed by QUIETCANARY. In one command observed, UNC4210 made a typo “netstat -ano -p tcppp” and had to reissue the command suggesting the following data theft was manual process rather than automated collection.

UNC4210 attempted to collect documents and data using WinRAR:

Data Collection Command	Primary Command Operational Choices
`rar a c:\\programdata\\win_rec.rar "%appdata%\\microsoft\\windows\\" -u -y -r -m2 -inul`	Creation of “win_rec.rar” archive containing files recursively found in directories within “% AppData%\Microsoft\Windows\”, which would have expanded to “C:\Users\ [Username]\AppData\Roaming\Microsoft\Windows\” as QUIETCANARY was executed under the compromised user’s context.
`rar a c:\\programdata\\win_rec.rar "c:\\users\\" -u -y -r -m2 -inul -n*.lnk`	Creation of “win_rec.rar” archive containing files with .lnk extension (namely Windows LNK shortcuts), recursively found in directories within “C:\Users\”
`rar a c:\\programdata\\win_files.rar "c:\\users\\" "d:\\" -u -y -r -m2 -inul -n*.pdf -n*.xls* -n*.txt -n*.doc* -hp[redacted] -v3M -ta20210101000000`	Creation of “win_files.rar” password (redacted) encrypted archive split in 3MB parts, containing files with extensions .pdf, .xls(x), .txt and .doc(x), which were modified after 2021-01-01, recursively found in directories within “C:\Users\” and “D:\”
`rar a c:\\programdata\\win_txt.rar "c:\\users" "d:\\" -u -y -r -m2 -inul -n*.txt -hp[redacted] -v3M`	Creation of “win_txt.rar” password (redacted) encrypted archive split in 3MB parts, containing files with extension .txt, recursively found in directories within “C:\Users\” and “D:\”

Notably, UNC4210 appeared to only exfiltrate files created after 2021/01/01.

ANDROMEDA C2s

As Mandiant began to review additional ANDROMEDA domains, we observed several more similarly re-registered domains from older malware. One domain “anam0rph[.]su”, was re-registered on 2022-08-12 and resolved to the same IP as yelprope.cloudns[.]cl during the same period. Mandiant assesses with high confidence that “anam0rph[.]su” is also controlled by UNC4210, though we have not observed the domain being used in UNC4210 operations.

Domain	Suspected Date of Re-Registration	IP Resolution
anam0rph[.]su	2022-08-12	212.114.52[.]24
yelprope.cloudns[.]cl	Unknown (Dynamic DNS)	212.114.52[.]24
suckmycocklameavindustry[.]in	2022-01-19	35.205.61[.]67

Conclusion

As older ANDROMEDA malware continues to spread from compromised USB devices, these re-registered domains pose a risk as new threat actors can take control and deliver new malware to victims. This novel technique of claiming expired domains used by widely distributed, financially motivated malware can enable follow-on compromises at a wide array of entities. Further, older malware and infrastructure may be more likely to be overlooked by defenders triaging a wide variety of alerts.

This is Mandiant’s first observation of suspected Turla targeting Ukrainian entities since the onset of the invasion. The campaign’s operational tactics appear consistent with Turla’s considerations for planning and advantageous positioning to achieve initial access into victim systems, as the group has leveraged USBs and conducted extensive victim profiling in the past. In this case, the extensive profiling achieved since January possibly allowed the group to select specific victim systems and tailor their follow-on exploitation efforts to gather and exfiltrate information of strategic importance to inform Russian priorities. However, we note some elements of this campaign that appear to be a departure from historical Turla operations. Both KOPILUWAK and QUIETCANARY were downloaded in

succession at various times, which may suggest the group was operating with haste or less concern for operational security, experiencing some aspect of operational deficiency, or using automated tools.

Acknowledgements

With thanks to Nick Richard and Parnian Najafi for technical review. Special thanks to all the Mandiant Consultants, Mandiant Managed Defense, and Pokemon Masters supporting Ukraine engagements.

Indicators

Family	Indicator
ANDROMEDA	bc76bd7b332aa8f6aedbb8e11b7ba9b6 TrustedInstaller.exe
ANDROMEDA	b3657bcfe8240bc0985093a0f8682703 mskmde.com
KOPILUWAK WinRAR SFX	2eb6df8795f513c324746646b594c019
KOPILUWAK	d8233448a3400c5677708a8500e3b2a0 xpexplore.js
QUIETCANARY	403876977dfb4ab2e2c15ad4b29423ff 00c3df3b.exe
QUIETCANARY	8954caa2017950e0f6269d6f6168b796 file.exe16
UNC4210 ANDROMEDA C2	suckmycocklameavindustry[.]in
UNC4210 ANDROMEDA C2	yelprope.cloudns[.]cl
UNC4210 ANDROMEDA C2	anam0rph[.]su
UNC4210 ANDROMEDA C2	212.114.52[.]24
UNC4210 KOPILUWAK C2	manager.surro[.]am
QUIETCANARY C2	194.67.209[.]186:443

YARA Rules

KOPILUWAK

```
rule M_APT_Kopiluwak_Recon_1

{

    meta:

        author = "Mandiant"

    strings:

        $rc4_1 = ".charCodeAt(i %"

        $rc4_2 = ".length)) % 256"

        $b64_1 = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/"

        $b64_3 = ".charAt(parseInt("

        $recon_1 = "WScript.CreateObject"

        $recon_2 = ".Run("

        $Arguments = "WScript.Arguments"

    condition:

        ($rc4_1 and $rc4_2 and $b64_1) and ($Arguments or ($b64_3 and $recon_1 and $recon_2))

}
```

QUIETCANARY

```
rule M_HUNTING_QUIETCANARY_STRINGS {

  meta:
```

```
        author="Mandiant"

  strings:

    $pdb1 =
"c:\\Users\\Scott\\source\\repos\\Kapushka.Client\\BrowserTelemetry\\obj\\Release\\CmService.pdb" ascii
wide nocase

    $pdb2 =
"c:\\Users\\Scott\\source\\repos\\Kapushka.Client\\BrowserTelemetry\\obj\\Release\\BrowserTelemetry.pdb
ascii wide nocase

    $pdb3 =
"c:\\Users\\Scott\\source\\repos\\BrowserTelemetry\\BrowserTelemetry\\obj\\Release\\BrowserTelemetry.pdb
ascii wide nocase

    $orb1 = {  68 00 74 00 74 00 70 00 73 00 3A 00 2F 00 2F }

    $orb2 = {  68 00 74 00 74 00 70 00 3A 00 2F 00 2F }

    $command1 = "get_Command" ascii wide nocase

    $command2 = "set_Command" ascii wide nocase

    $command3 = "DownloadCommand" ascii wide nocase

    $command4 = "UploadCommand"  ascii wide nocase

    $command5 = "AddCommand" ascii wide nocase

    $command6 = "ExeCommand" ascii wide nocase

    $command7 = "KillCommand" ascii wide nocase

    $command8 = "ClearCommand"  ascii wide nocase

      $rc4 = {21 00 62 00 76 00 7A 00 65 00 26 00 78 00 61 00 62 00 72 00 39 00 7C 00 38 00 5B 00 3F 00
78 00 77 00 7C 00 7C 00 79 00 26 00 7A 00 6C 00 23 00 74 00 70 00

6B 00 7A 00 6A 00 5E 00 62 00 39 00 61 00 38 00 6A 00 5D 00 40 00 6D 00 39 00 6E 00 28 00 67 00 67 00 2
00 40 00 74 00 74 00 65 00 33 00 33 00 6E 00 28 00 32 00 72 00 7A

00 62 00 7A 00 69 00 74 00 75 00 31 00 2A 00 66 00 61 00 00 80 E9 4D 00 6F 00 7A 00 69 00 6C 00 6C 00 6
}

  condition:

    (1 of ($pdb*)) and (1 of ($orb*)) and (all of ($command*)) or ($rc4)

}
```

Network Rules

ANDROMEDA

```
alert tcp any any -> any any ( msg:"503 irc_bot_cmd Trojan.Downloader.Andromeda AI callback-trojan
block"; content:".php HTTP/1"; nocase; content:"|0a|"; content:"|0a|"; within:4; content:"POST ";
content:"Mozilla/4.0|0d 0a|"; content:!"Referer: "; nocase; content:!"Cookie: "; nocase;
content:!"Accept-Language: "; nocase; content:!"Accept-Encoding: "; nocase; content:!"pharma";
nocase; content:!"|0d0a|TE:"; nocase; pcre:"/POST (http\:\/\/\S*\.[a-z0-9]{1,4})?/[a-z]{1,3}\.php
HTTP/"; reference:fe_date,2013-07-11; reference:a_type,mal.dsh;
reference:mal_hash,bc76bd7b332aa8f6aedbb8e11b7ba9b6; priority:90; sid:89039193; rev:3; )
```

MITRE ATT&CK

ATT&CK Tactic Category Techniques

Defense Evasion

T1027:	Obfuscated Files or Information
T1055:	Process Injection
T1070.004:	File Deletion

	T1112:	Modify Registry
	T1564.003:	Hidden Window
	T1622:	Debugger Evasion
Persistence		
	T1547.001	Registry Run Keys / Startup Folder
Discovery		
	T1010:	Application Window Discovery
	T1012:	Query Registry
	T1033:	System Owner/User Discovery
	T1049:	System Network Connections Discovery
	T1057:	Process Discovery
	T1082:	System Information Discovery
	T1083:	File and Directory Discovery
	T1518:	Software Discovery
Collection		
	T1560:	Archive Collected Data
	T1560.001:	Archive via Utility
Resource Development		
	T1584:	Compromise Infrastructure
	T1608.003:	Install Digital Certificate
Command and Control		
	T1071.001:	Web Protocols
	T1573.002:	Asymmetric Cryptography
Impact		
	T1529:	System Shutdown/Reboot

Annex: QUIETCANARY Analysis

QUIETCANARY is a .NET backdoor that can handle commands from C2. The communications between QUIETCANARY and the hard-coded C2 are RC4 encrypted and Base64 encoded over HTTPS. QUIETCANARY samples often contain the artifact “Kapushka” in the PDB path of the malware. All samples of QUIETCANARY we have identified in the wild have been nearly identical with the only differences being the hardcoded C2 and RC4 key used. Notably as well, each sample of QUIETCANARY we found contains but does not use the class ServerInfoExtractor.

QUIETCANARY Execution

Upon execution, QUIETCANARY initializes the hard-coded variables within it for C2 communication, including the RC4 key, user agent, and C2. It then attempts to connect to the C2 via GET request.

QUIETCANARY checks to see if the initial response from the C2 is longer than 13 characters and begins with the string "use." If so, it takes the substring between the third and tenth characters from the response and replaces the initial RC4 key with a new key. If this initial connection or exchange of a new RC4 key is not completed, the malware sleeps for five minutes and tries again. If the malware fails to connect to the C2 again, it quits execution. Otherwise, QUIETCANARY begins a loop that waits for a response from the C2, and, when it receives one, parses and executes the command.

QUIETCANARY does not contain a persistence method and thus relies on an external tool or technique to maintain persistence.

QUIETCANARY Commands

QUIETCANARY uses a custom parsing routine to decode the command codes and additional parameters from the C2 before executing the command routines.

QUIETCANARY expects the following structure for a command from the C2:

i <id> **s** <code val> **l** <length of parameter> **c** < parameter>

Where:

- <id> is a decimal command ID, which can be up to ten digits
- <code val> is a three-digit command code
- <length of parameter> is the length of an expected parameter, which can be up to six digits
- <parameter> immediately follows the "c" character in the array and is the length specified by <length of parameter>

For example, a command from the C2 could look like:

i123456789s220l26ccmd.exe echo "hello world"

When a command is successfully parsed, the command code is queried and returned.

QUIETCANARY can parse multiple commands in a single response from the C2. Upon receiving and successfully parsing a command, QUIETCANARY adds it to a command queue.

QUIETCANARY can handle the following command codes from the C2:

Command codes

Command Code	Command Name	Command Description
0	ClearCommand	Aborts current command execution thread and starts a new one
220	ExeCommand	Execute a command with arguments
265	TimeoutCommand	Sets a new time until C2 loop execution times out
420	UploadCommand	Uploads a command to a given path
479	DownloadCommand	Reads all bytes from a filepath and converts to Base64 encoding
666	KillCommand	Kills execution after a specified delay

Given the different encodings of each command, we detail here how QUIETCANARY processes each command from the C2 before execution.

0: ClearCommand

Aborts current thread of command execution and clears current command in the queue. Starts new command execution thread.

220: ExeCommand

Checks to see if parameter starts with the " character. If so, creates a new string that begins with the "\" character and consists of each following character in the parameter until it reaches a second " character.

If the parameter does not start with the " character, splits the parameter into substrings by space characters. Arguments directly follow the filename to execute. Then executes the process name with arguments in a hidden window.

265: TimeoutCommand

Takes an integer parameter and changes the timeout value in seconds for the execution of the C2 loop.

420: UploadCommand

Takes a "|" delimited parameter and splits into substrings. The first substring is the path to which the command should be uploaded. Converts each substring following the first from Base64 then writes to the given path on the infected machine.

479: DownloadCommand

Checks to see if the file passed to the function within the parameter exists. If it does exist, prepends "file:" to the Base64-encoded bytes of the specified filepath. If it does not, returns "not found." Results of execution are stored in memory and later sent to C2.

666: KillCommand

Takes an integer it uses to calculate a time for the malware to sleep. Then kills the process after the sleep.

QUIETCANARY separates the commands into two categories: fast commands and long commands. Fast commands are executed outside the timeout command structure. The commands will execute once received, and results will immediately be generated after execution. QUIETCANARY interprets *ClearCommand* and *KillCommand* as fast commands. Every other supported command is a long command. Long commands are held to a timeout structure. Once the commands are parsed, they are queued. Access to and execution of the commands is synchronized using the C# Monitor class.

It then generates a response string in the following format:

i<command id>**l**<length of result>**r**<result of command execution>

Where the value following the "r" character is "ok" if the command was successful, and, if the command execution was unsuccessful, the type of command followed by "error" (i.e., "kill error," "command error," etc.).

QUIETCANARY appends the response string to the string "rep" and sends a POST request to the C2. If the C2s response to the POST request is greater than 3 characters long and the first 3 characters are "per," it returns the string after the first 3 characters to be parsed.

QUIETCANARY's Unused Code

The sample of QUIETCANARY we analyzed had an unused class called ServerInfoExtractor. The class contains a function that would get some Base64-encoded value from the following registry key:

Key: HKCU\Software\Microsoft\Fax\Verification

If the value within the key is "No," the function returns the result "No." Otherwise, it Base64 decodes the contents and returns them.

We suspect this registry value may be used to store some configuration in other versions of QUIETCANARY. Since no code is present in the QUIETCANARY sample that would set this configuration value, we assume this would be set by some method outside the malware.

QUIETCANARY Network Communications

All network communications between QUIETCANARY and the C2 are RC4-encrypted and then Base-64 encoded. QUIETCANARY is proxy-aware and uses the System.Net.HttpWebRequest class to get any default proxy specified on the victim computer. The malware also dynamically generates a random PHP session ID to use when sending a GET request to the C2, which is added to the Cookie field in the request header.