PUBLISHED: JANUARY 30, 2018

DEEP DIVE: KASEYA VSA MINING PAYLOAD

By: Kyle Hanslovan

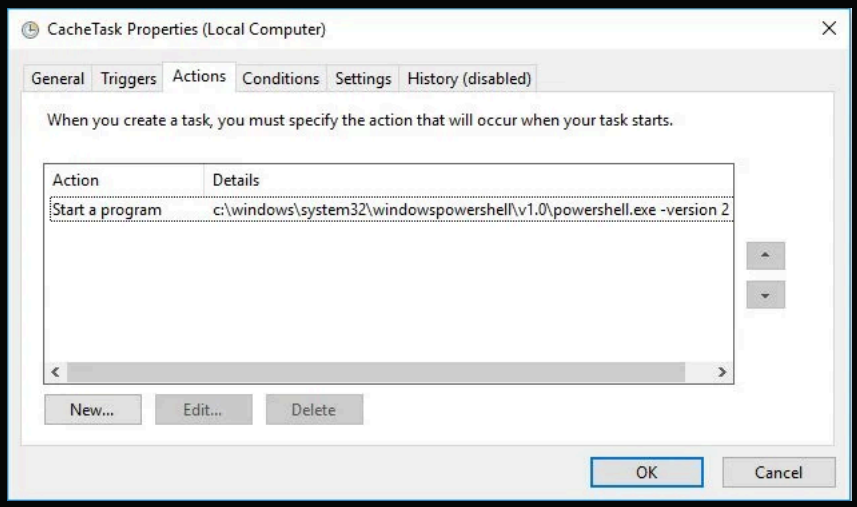
For many of us in the Managed Services Provider market, we were rocked with news of a vulnerability in Kaseya's VSA product. The purpose of this blog is to shine technical light on what the Huntress ThreatOps team observed and analyzed thus far. For official Kaseya guidance, make sure you reference their Security Update Notice and Removal Procedures (kudos Kaseya Security Team on the discovery and quick response efforts!)

BACKGROUND INFORMATION

Over the last few weeks, our team uncovered dozens of suspicious Scheduled Tasks used to execute a persistent payload with Local System privileges. When run, the footholds launched PowerShell to run Base64 encoded files stored within the Windows Registry and on Dropbox hosted domains. The original script used to infect the hosts looked like this:

Loading Gist...

MONITORING FOR POWERSHELL EXECUTION



As illustrated above, a persistent foothold is established with a single PowerShell command that downloads and executes the PowerShell script at:

- hxxps://dl[.]dropboxusercontent[.]com/s/namkbc37wtx2mdq/reg_load_subsequent.ps1

Loading Gist...

Categories

Rapid Responses

SEE HUNTRESS IN ACTION

Our platform combines a suite of powerful managed detection and response tools for endpoints and Microsoft 365 identities, science-backed security awareness training, and the expertise of our 24/7 Security Operations Center (SOC).

Book a Demo

Share

When run, the script first validates whether its payload is already running. To do this, it gathers the list of running processes and won't execute any further if more than one PowerShell process is already running.

As silly as it sounds, this particular threat could be temporarily "immunized" against by running two or more long running PowerShell processes :)

CHOOSING AND STORING 32 AND 64-BIT PAYLOADS

When delivering malware to victims, some attackers take the time to make sure their payloads match the same architecture of the compromised systems. In this case, the following code performs this check:

Loading Gist...

For those unfamiliar, the attacker chose to measure the size of an integer pointer structure. If the size is 4 bytes (32 bits), the malware downloads the content of the x86 payloads from the appropriate URLs (notice they end in _32.txt). If the structure is any other size, it assumes the host is a 64-bit system.

Loading Gist...

With this payload data, the malware stores the contents within several registry key values along with another heavily obfuscated PowerShell script. The data was stored within the following paths:

- Key: HKLM\SOFTWARE\Microsoft\Powershell\Scripts | Value: a
- Key: HKLM\SOFTWARE\Microsoft\Powershell\Scripts | Value: b
- Key: HKLM\SOFTWARE\Microsoft\Powershell\Scripts | Value: c
- Key: HKLM\SOFTWARE\Microsoft\Powershell\Scripts | Value: d
- Key: HKLM\SOFTWARE\Microsoft\Powershell | Value: ScriptInit

SCHEDULED TASKS CONFIGURATIONS

As mentioned earlier, the infection script leveraged scheduled tasks for its foothold. Many attackers use schtasks.exe and its abundant parameters to create/configure tasks to run:

- the desired payloads
- at expected intervals
- with specified privileges

However, this actor decided to create the scheduled tasks from an XML template that's also downloaded from DropBox.

Loading Gist...

HIDING SCHEDULED TASKS IN PLAIN SIGHT

Rather than simply create a scheduled task with a hard-coded name, the infection script cleverly generates a “random” name. It does this by combining the path and task name of two legitimate, randomly selected scheduled tasks with the following code:

Loading Gist...

To better illustrate how this works, let’s use these examples as the randomly selected scheduled tasks:

- \Microsoft\Windows\UPnP\UPnPHostConfig
- \Microsoft\Windows\SettingSync\BackupTask

The above code splits each of them into a “task path” and “task name”:

- Task Path #1: \Microsoft\Windows\UPnP
- Task Name #1: UPnPHostConfig
- Task Path #2: \Microsoft\Windows\SettingSync
- Task Name #2: BackupTask

Now the script joins the second task path to the first task name to create a scheduled task name that easily hides in plain sight from human analysis:

- \Microsoft\Windows\SettingSync\UPnPHostConfig

With this custom task name and the downloaded XML template, the infection script executes schtasks.exe with to create the malicious scheduled task.

> schtasks /CREATE /XML $file /TN “$fullTaskName” /F

UPDATED POWERSHELL CODE

On the evening of January 29th, 2018, the attackers updated the PowerShell command described in this blog to evade Kaseya’s Automated Removal Procedure.

Loading Gist...

Most notably, the registry key path and value name were updated from:

- Key: HKLM\SOFTWARE\Microsoft\Powershell\Scripts | Value: *
- Key: HKLM\SOFTWARE\Microsoft\Powershell | Value: ScriptInit

to:

- Key: HKLM\SOFTWARE\Microsoft\Policies\System| Value: *
- Key: HKLM\SOFTWARE\Microsoft\Policies| Value: Start

This caused Kaseya’s initial Automated Removal Script (XMR.xml) to no longer delete all of the malicious registry key values.

Loading Gist...

Unfortunately, the attacker designed this change to prevent any new Scheduled Task based footholds from being removed as the original “signature” matched on the presence of “ScriptInit” within the command’s arguments (which is now “Start”).

Loading Gist...

Lastly, the attacker got extra crafty and decided to backdoor one of Microsoft's legitimate Scheduled Tasks: "sihboot".

Loading Gist...

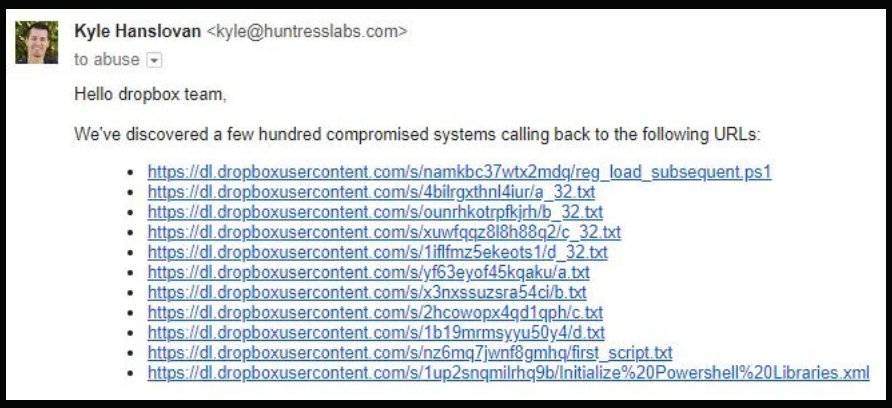
Using the Scheduled Task XML template from the URL above, the attacker added an extra Exec Action which once again runs the malicious PowerShell payload:

Loading Gist...

Douglas Sanchez from Kaseya has already updated the official Automated Removal Procedures and Manual Removal Instructions.

FILE & SCRIPT TAKEDOWN

To help mitigate the impact of these payloads, the Huntress ThreatOps team has coordinated with Dropbox to remove the following malicious URLs



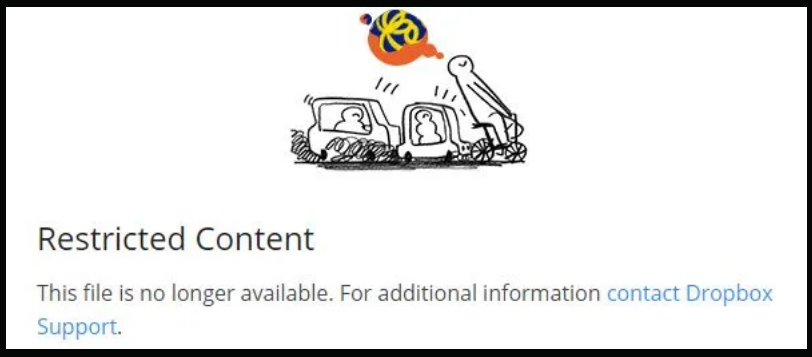
Kyle Hanslovan <kyle@huntresslabs.com>
to abuse

Hello dropbox team,

We've discovered a few hundred compromised systems calling back to the following URLs:

- https://dl.dropboxusercontent.com/s/namkbc37wtx2mdq/reg_load_subsequent.ps1
- https://dl.dropboxusercontent.com/s/4bilrgxthnl4iur/a_32.txt
- https://dl.dropboxusercontent.com/s/ounrhkotrpfkjrh/b_32.txt
- https://dl.dropboxusercontent.com/s/xuwfqqz8l8h88q2/c_32.txt
- https://dl.dropboxusercontent.com/s/1iflfmz5ekeots1/d_32.txt
- https://dl.dropboxusercontent.com/s/yf63eyof45kqaku/a.txt
- https://dl.dropboxusercontent.com/s/x3nxssuzsra54ci/b.txt
- https://dl.dropboxusercontent.com/s/2hcowopx4qd1qph/c.txt
- https://dl.dropboxusercontent.com/s/1b19mrmsyyu50y4/d.txt
- https://dl.dropboxusercontent.com/s/nz6mq7jwnf8gmhq/first_script.txt
- https://dl.dropboxusercontent.com/s/1up2snqmilrhq9b/Initialize%20Powershell%20Libraries.xml

On the afternoon of January 30th, 2018, the Dropbox team took these suckers down :)

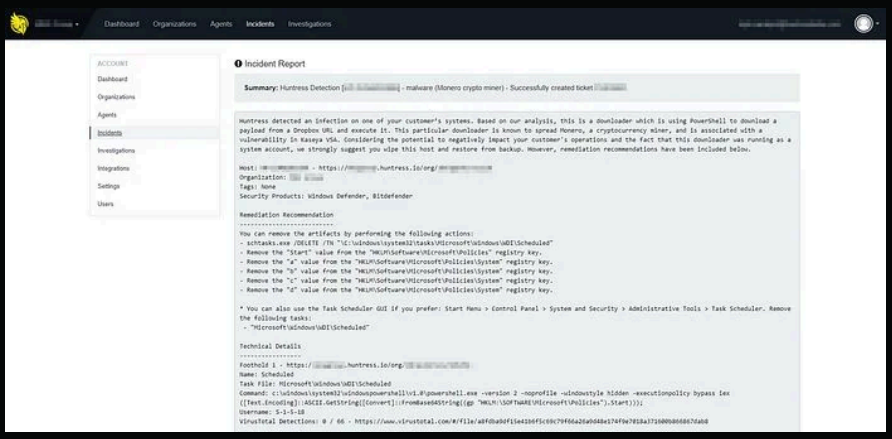


If you'd like to analyze these files and scripts for yourself, drop us a line at support[at]huntresslabs.com and we'll gladly send them your way.

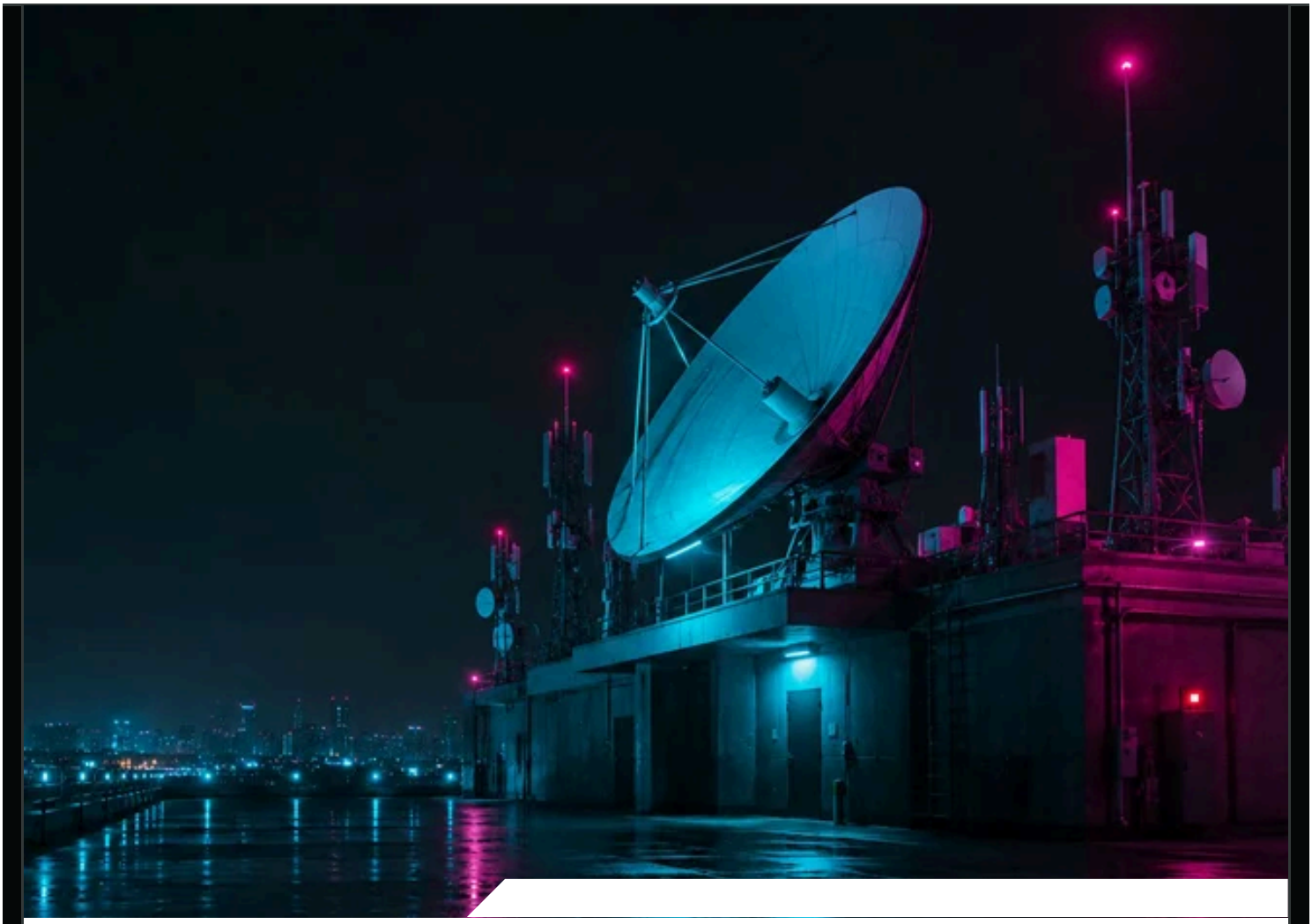
CURIOUS IF YOU'RE STILL COMPROMISED?

We offer a 21-day trial of Huntress for an unlimited number of computers. Simply deploy our agent, setup a reporting integration into your ticketing system, and we'll deliver step-by-step remediation procedures for each compromised host we discover. When the trial ends, our

team can remotely uninstall our agents with a single click (no extra cleanup). Contact sales[at]huntresslabs.com for demos and more details :)



YOU MIGHT ALSO LIKE



SIGN UP FOR HUNTRESS UPDATES

Get insider access to Huntress tradecraft, killer events, and the freshest blog updates.

Business Email*

Submit

By submitting this form, you accept our Terms of Service & Privacy Policy

CLOUDFLARE
Privacy • Terms