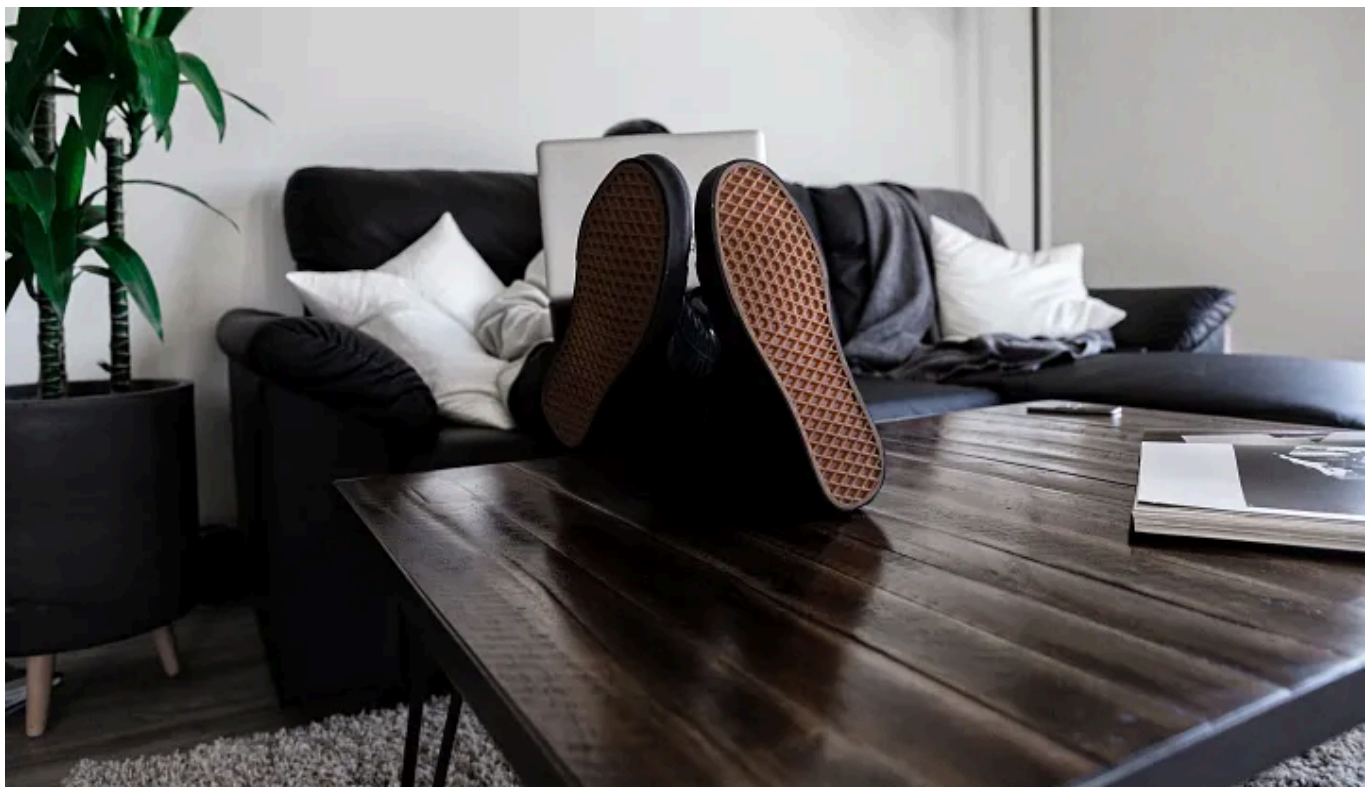


Search-ms, WebDAV, and Chill

Detecting a [Re-]emerging Initial Access Method

Micah Babinski · Follow

12 min read · Aug 1, 2023

Introduction

In this article we'll take a close look at the techniques described by Trellix researchers Mathanraj Thangaraju and Sijo Jacob in their recent piece for the Trellix blog entitled Beyond File Search: A Novel Method for Exploiting the "search-ms" URI Protocol Handler. In this post, Thangaraju and Jacob explore a recent campaign where threat actors exploit built-in Windows search capabilities in conjunction with WebDAV to trick unwitting victims into executing malware on their systems. The team at Trellix always delivers top-notch work and I appreciate how well they explain attacks like the one covered here.

Having read the article, I wanted to see for myself how the attack works, what it looks like in the logs, and how we might improve current detection concepts for this attack. If this sounds interesting to you, read on!

Our mission is simple:

🤔 Understand the threat from a technical perspective (with empathy and understanding for the victim's point-of-view as well).

👩‍🏫 Demonstrate the attack in our lab to see how it works in practice.

🔬 Analyze the resulting log data to see what is generated.

🧩 Review existing detection concepts, and create some of our own if the existing ones leave a gap.

🤸 Celebrate and share our work with the community. We never skip this part!

A Valuable Focus on Initial Access Detection

As detection specialists, we have seen lots of different methods by which attackers breach our systems. The philosophy of "assume breach" is a powerful guiding force in our work, and we are rightly accustomed to working hand-in-hand with our incident responder colleagues to address the post-compromise activity that we find.

But, think about how much headache, time, and money could be saved by catching more cyberattacks early — at the Initial Access phase or shortly after. By getting eyes on threats early in the attack chain, we stand to save staff time, remediation costs, and business productivity. So I love when researchers like the gunslingers at Trellix provide us with a new avenue to catch threats early! See my articles on OneNote and HTML smuggling for some other examples. As LD might say, catch the attackers early and $ave Dat Money!

Note: According to the Trellix researchers, the WebDAV/search-ms techniques described here serve as an initial access vector for AsyncRAT and RemcosRAT. I believe them: my goal here is only to better understand the *delivery* mechanism, and not necessarily to perform a comprehensive, deep-dive analysis of the malware itself!

Understanding the Threat

As the Trellix researchers show, the attack is a classic phish, but using a compromised or malicious web server to give the victim a view of the malicious payload that mimics what they might see in their file system in the course of normal business. By presenting the dangerous files via file explorer (a trusted, daily-use tool for most business computer users), attackers seek to trick unsuspecting victims into trusting the content more than they might if prompted to download a zipfile, click a link on a sketchy-looking web page, or insert a malicious USB drive into their work computer.

The threat relies on two technologies that some people may not be familiar with because they don't come up every day: WebDAV and the search-ms application protocol.

1. **Web Distributed Authoring and Versioning (WebDAV)** is a means by which web developers can manage and maintain data on a web server using an extra set of HTTP protocols. Practically speaking, all this means is that web developers can use a tool like Windows File Explorer to manage the files and folders on their web server directly. WebDAV has been around for a long time and has many legitimate uses. Still, it is not common for everyday business users to access data on a web server via WebDAV. I had never heard of WebDAV until 2020 when I started studying cybersecurity, and I had worked with computers since 2010! If you are new to WebDAV, you can learn all you need to know about it here.

2. The **search-ms protocol** is simply a means by which you can query the "Windows Search index" using a URI. This sounds fancy, but all it means is that search-ms provides a means to query the contents of files and folders using a component of a URL. In the case of this threat, that URL will be searching for specifically-named content on a malicious WebDAV server. There's no need to become a search-ms protocol expert to successfully detect this attack, but if you want to read more about it, you can do so here.

In combination, WebDAV and search-ms can present an end user with a file explorer window view of a malicious file, in our case a .lnk file disguised as a PDF. From there, the pattern is familiar: various built-in system utilities are used to retrieve second-stage payloads in the form of PowerShell, VBScript, or ISO archives which mount as drives and contain DLLs or EXEs with Remote Access Trojans (RAT), which steal data, keystrokes, screenshots, and other valuable resources from the victim's system.

Demonstrating the Attack

Let's hop into the lab and see how this attack looks from the victim's POV.

As a quick aside: I think it is important to consider the victim's experience and remember that, as security professionals, users are our customers, and not the cause of our problems. Without end users (yes, some of whom occasionally click on stuff they don't mean to) we would not exist, so try to show compassion for them! I find it very tiresome when security professionals bad-mouth or denigrate the end users (who spend their days creating value at their organizations).

My reaction when security pros spout that "users are the weakest link" line.

Ok, enough soapboxing. Let's click on some malicious stuff! I access a malicious web page from the Trellix blog on the win10 VM system in my lab. In this case, the URL I chose is likely contained in an invoice-themed phishing link. Note the URL ending in /index.html — hardly a suspicious file path on its own!

Using Chrome Developer tools, I am able to see the secondary request to a ######[.]webdav[.]drivehq[.]com domain, complete with text showing the use of search-ms to pre-populate the Windows search with a search term of "Invoice_BVJ0KS_pdf". Choosing "Open Windows Explorer," I am presented

with a file explorer window which displays a filtered search result showing an invoice-related filename.

If I double-click the "_pdf" link file, nothing happens. Hmm…not very satisfying. Trying another malicious URL, I am once again presented with the File Explorer window, but this time the search results are empty. However, if I remove the pre-filled search terms, I am presented with what looks like the file directory listing of a web server!

Clicking on the "Invoice_757780_PDF_034531_DeletedItem" shortcut file, I am prompted with a security warning, which I of course do not heed.

By examining the directory contents via the WebDAV connection, I can see that this lnk file prompts the execution of ca.ps1, a PowerShell script which in turn launches casxe.vbs, located in the DavWWWRoot folder.

Unfortunately, my test system was unable to access the malicious ca.ps1 PowerShell script, so I proceeded on to the next malicious URL in the Trellix blog.

WebDAV and Chill

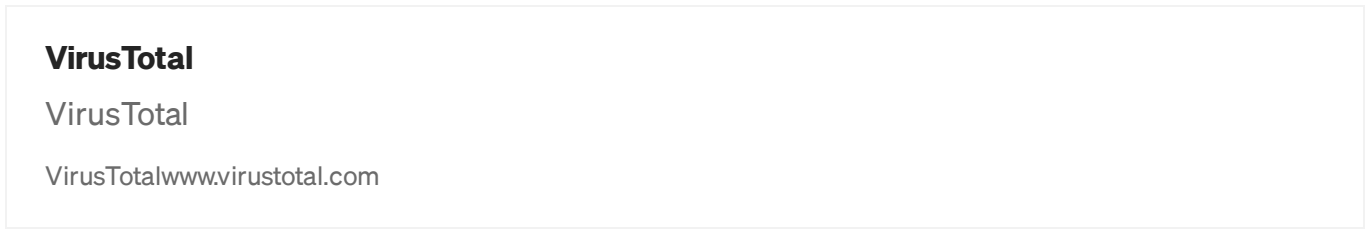
After spending a couple hours browsing these various WebDAV directory listings, deciphering the various malicious files (.lnk files, VB scripts, powershell scripts, batch files), and attempting to get something interesting to happen, I realized that I had a problem: many of the WebDAV servers shared in the Trellix article were inactive or had settings which prevented my test system from executing the various malicious scripts and programs. This threatened to prevent me from continuing my research:

D’oh!

This led me to an interesting conundrum: here I was trying to research a threat, but the infrastructure built to deliver that threat to victims was not working properly for me.

Archival footage of me encountering a setback in my lab.

To overcome this obstacle, I decided to make use of the subdomain relations of the *.webdav.drivehq.com site on VirusTotal, shown here:



VirusTotal knows about 52 subdomains of this site, several of which are flagged for malicious activity. VirusTotal knows, so therefore, so do I! Using a Python library called WebDAVClient3, I created a simple Python script to scan the webdav subdomains, list their contents, and print the modified dates of each file and directory hosted at the root level.

Behold my quick-and-dirty Python script

With an active, recently-modified .lnk file identified on a responding WebDAV server, I finally had what I needed to proceed with my research. I dutifully double-clicked the shortcut file highlighted below (note the tooltip which indicates that this actually executed wscript.exe), saw a command-line window pop-up briefly, and checked the logs in Splunk.

Best believe I use IDLE

I was not disappointed…

Me, seeing the rich tapestry of logs generated by this activity

Log Analysis Stage

With an eye towards detecting this attack, the first thing I noticed when looking at the logs from this activity in Splunk was a flood of events like this:

Just browsing and lightly-interacting with the malicious files being served up via WebDAV created a huge volume of process creation logs in my lab. These logs, while high-volume, are also somewhat informative, telling me that:

- RunDLL32.exe runs davclnt.dll, specifically a DavSetCookie function.

- The path of the WebDAV server where the file was accessed.
- The filename and extension of the remote file.

Flipping over to file creation events, I noticed some interesting-looking temporary files being created here:

```
C:\Windows\SERVIC~1\LOCALS~1\AppData\Local\Temp\TfsStore\Tfs_DAV\{6000B464-A64F-45FE
C:\Windows\SERVIC~1\LOCALS~1\AppData\Local\Temp\TfsStore\Tfs_DAV\{3D1DC9F7-6DFD-471C
C:\Windows\SERVIC~1\LOCALS~1\AppData\Local\Temp\TfsStore\Tfs_DAV\{C56C1210-6CAE-477F
```

These caught my attention because they appear to offer evidence of the remote WebDAV files being retrieved, in at least a temporary state, from the malicious server to the local system. Copying the malicious DP.vbs file to the test system Desktop, I generated a SHA1 hash of the copied file, plus the .vbs file created in the AppData\Local\Temp directory and verified that they are the same:

Following the execution of the malicious .lnk file, which used wscript.exe to execute DP.vbs, the malware dropped a highly-obfucsated batch file (temparchivo.bat) which began with a lengthy comment likely meant to confuse security analysts:

Immediately following the execution of the batch file, I saw some suspicious processes created associated with the file Temparchivo.bat.sc:

…which, sure enough, turned out to be a renamed powershell.exe:

Sneaky! Here is the full CommandLine text for the process shown above if you are curious, or [better yet] want to tell me what it does:

```
"C:\Users\vagrant\AppData\Local\Temparchivo.bat.scr "  -w hidden -c $zDHz='InvoZWVgk
```

Reviewing Existing Detection Content and Assessing Gaps

At this point, I had seen enough; I knew that my test system was thoroughly infected. A scheduled task had been created for persistence, there was some reflective code loading going on each time the PowerShell above ran, and I was well on my way to having my data stolen by something like AsyncRAT and shipped off to the attacker for them to profit off of.

Remember, my goal is not to exhaustively dissect the malware that infected my test VM. I just want to better understand this interesting/novel/emerging initial access vector, so that we can review the existing detection coverage and, as Dr. Fauci used to say, possibly give it a boost!

Who needs a boost?

Search-MS URI Protocol Abuse

One would hope that, given the attack described here uses a specific URI string of “search-ms:” or “search:” that this string would simply show up somewhere in a process creation log. However I did not find that to be the case, contrary to some resources I found online which contain hunting queries for this activity. One possibility would be to use proxy logs, which should record the text of the URI strings which pass through the forward proxy to the remote server. As shown below, we can search the URI string for the suspicious keywords, as well as an optional user-defined list of suspect search terms like “Invoice.” However, I didn’t have the opportunity to test this concept, so I can’t guarantee it will work!

```
title: Search-ms and WebDAV Indicators in URL
id: 5039f3d2-406a-4c1a-9350-7a5a85dc84c2
status: experimental
description: Detects URL pattern used by search-ms/WebDAV initial access campaign.
references:
    - https://www.trellix.com/en-us/about/newsroom/stories/research/beyond-file-sear
author: Micah Babinski
date: 2023/07/31
tags:
    - attack.initial_access
    - attack.t1584
    - attack.t1566
logsource:
```

```
        category: proxy
    detection:
        selection_search_ms:
            c-uri|contains|all:
                - 'search'
                - ':query='
                - 'webdav'
        selection_search_term:
            c-uri|contains:
                - 'invoice'
                - 'payment'
                - 'notice'
                - 'agreement'
                # add others!
        filter:
            dst_ip:
                - '127.0.0.0/8'
                - '10.0.0.0/8'
                - '172.16.0.0/12'
                - '192.168.0.0/16'
        condition: all of selection_* and not filter
    falsepositives:
        - Legitimate use of search-ms URI protocol
    level: high
```

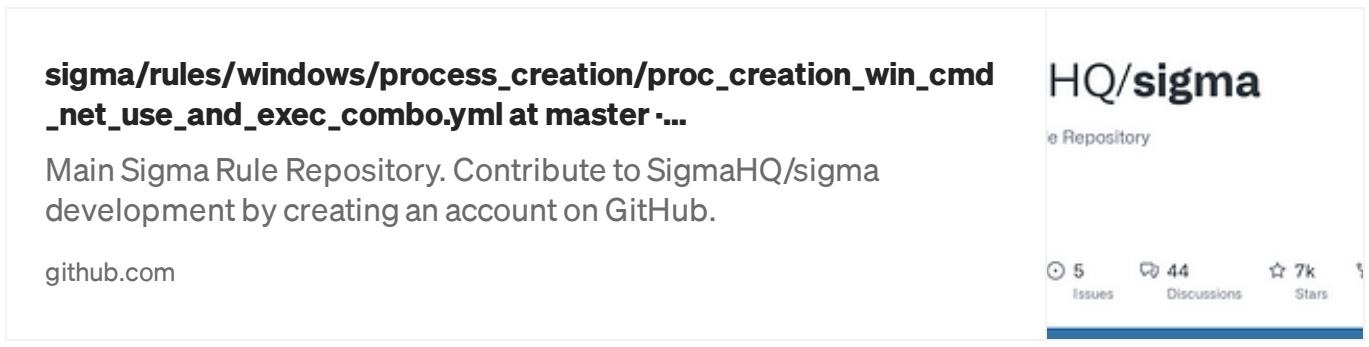
Local File Creation

The local temporary WebDAV file creation mentioned earlier creates an interesting opportunity to detect WebDAV abuse, and I was unable to find any open-source rules to detect this behavior. The Sigma rule below will alert on local creation of temporary WebDAV files with any of the suspicious file extensions I saw during the research.

```
title: WebDAV Temporary Local File Creation
id: 4c55738d-72d8-490e-a2db-7969654e375f
status: experimental
description: Detects the creation of WebDAV temporary files with suspicious extensio
references:
    - https://www.trellix.com/en-us/about/newsroom/stories/research/beyond-file-sear
author: Micah Babinski
date: 2023/07/31
tags:
    - attack.initial_access
    - attack.t1584
    - attack.t1566
logsource:
    product: windows
    category: file_event
detection:
    selection_1:
        TargetFilename|contains: 'AppData\Local\Temp\TfsStore\Tfs_DAV'
    selection_2:
        TargetFilename|endswith:
            - '.vbs'
            - '.ps1'
            - '.lnk'
            - '.zip'
            - '.ico'
            - '.bat'
    condition: all of selection_*
falsepositives:
    - Legitimate use of WebDAV in an environment
level: low
```

Process Execution

WebDAV usage is not suspicious on its own. We need some detection concepts that will help us identify the bad stuff. Double-clicking .lnk files like the malicious “Invoice_9283_pdf” shortcut shown above will result in explorer.exe spawning whatever executable the LNK file targets. Since the WebDAV-delivered .vbs script file resides in the WebDAV server directory, we know that the resulting process Command Line will contain the path to this directory, which from what I’ve seen always contains DavWWWRoot. This Sigma rule from Netron Systems covers this technique, but is limited to net use:



I adapted this rule to the specific activity observed in my research to get the following (I’ve referenced the other Sigma rule in the related section):

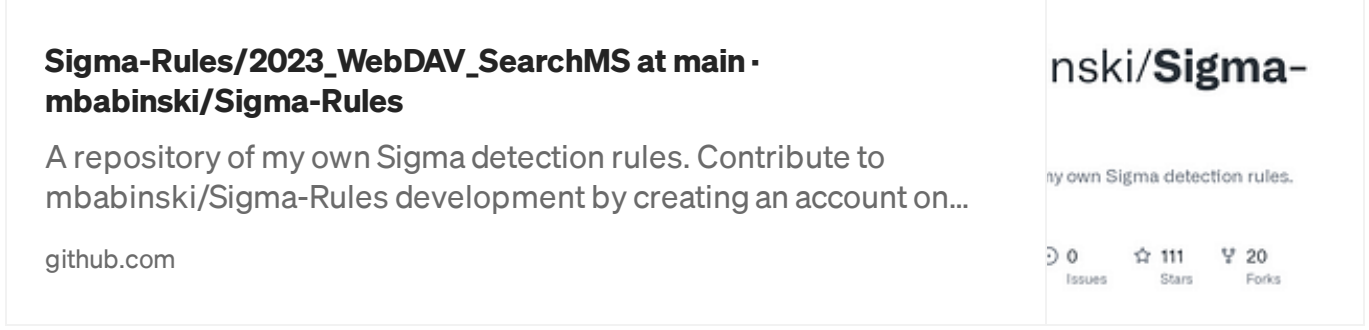
```
title: Suspicious WebDAV LNK Execution
id: 1412aa78-a24c-4abd-83df-767dfb2c5bbe
related:
  - id: f0507c0f-a3a2-40f5-acc6-7f543c334993
    type: similar
status: experimental
description: Detects possible execution via LNK file accessed on a WebDAV server.
references:
    - https://www.trellix.com/en-us/about/newsroom/stories/research/beyond-file-sear
    - https://micahbabinski.medium.com/search-ms-webdav-and-chill-99c5b23ac462
author: Micah Babinski
date: 2023/07/31
tags:
    - attack.execution
    - attack.t1059.001
    - attack.t1204
logsource:
    category: process_creation
    product: windows
detection:
    selection_img:
        ParentImage|endswith: '\explorer.exe'
        Image|endswith:
            - '\wscript.exe'
            - '\cscript.exe'
            - '\cmd.exe'
    selection_cmd:
        CommandLine|contains: '\DavWWWRoot\'
    condition: all of selection_*
falsepositives:
    - Unknown
level: high
```

Conclusion

There you have it! Another journey through a weird~~, wonderful~~ unsettling world of using obscure protocols like search-ms/WebDAV to deliver

malware. I had a lot of fun conducting this research, and particularly enjoyed using Python to analyze malicious WebDAV servers to see which ones might still be active. If you are just getting started doing this type of research, I hope you will be inspired to roll up your sleeves and consider scripting as a tool in your toolbox when presented with time-consuming, tedious tasks like seeing which malicious scripts you can get to run in your lab VM. 😂

I am 100% open to and appreciative of any feedback you might wish to offer on my work. I've really enjoyed sharing some of the research I conduct on my off-time with the security community, and am so grateful for those who choose to read and share it. 🙏 If you'd like to access the Sigma rules I've included on GitHub, you can do so here:



As always, Happy Analyzing! 🧐

Detection Engineering | Cybersecurity | Information Security | Infosec

-- 1

Written by Micah Babinski

256 Followers

Cybersecurity pro, featuring bagpiping and GIS chops. Lives with wife Quinn and son Malcolm. Loves mountains, Indian food, and mountains of Indian food.

Follow