IcedID Campaign Strikes Back

blogs.juniper.net/en-us/threat-research/iceid-campaign-strikes-back

August 12, 2020

In our previous blog about IcedID, we explored some of the changes in the malware and how it tries to evade detection. We also detailed how threat actors took advantage of the COVID-19 pandemic to phish their target victims. Recently, we discovered an evolution in their phishing methods, particularly how they attempt to evade detection by implementing a password protected attachment, keyword obfuscation and minimalist macro code in their trojanized documents. This time, they also use a DLL for the second stage downloader, which shows a new maturity level of this threat actor.

Phishing Victims

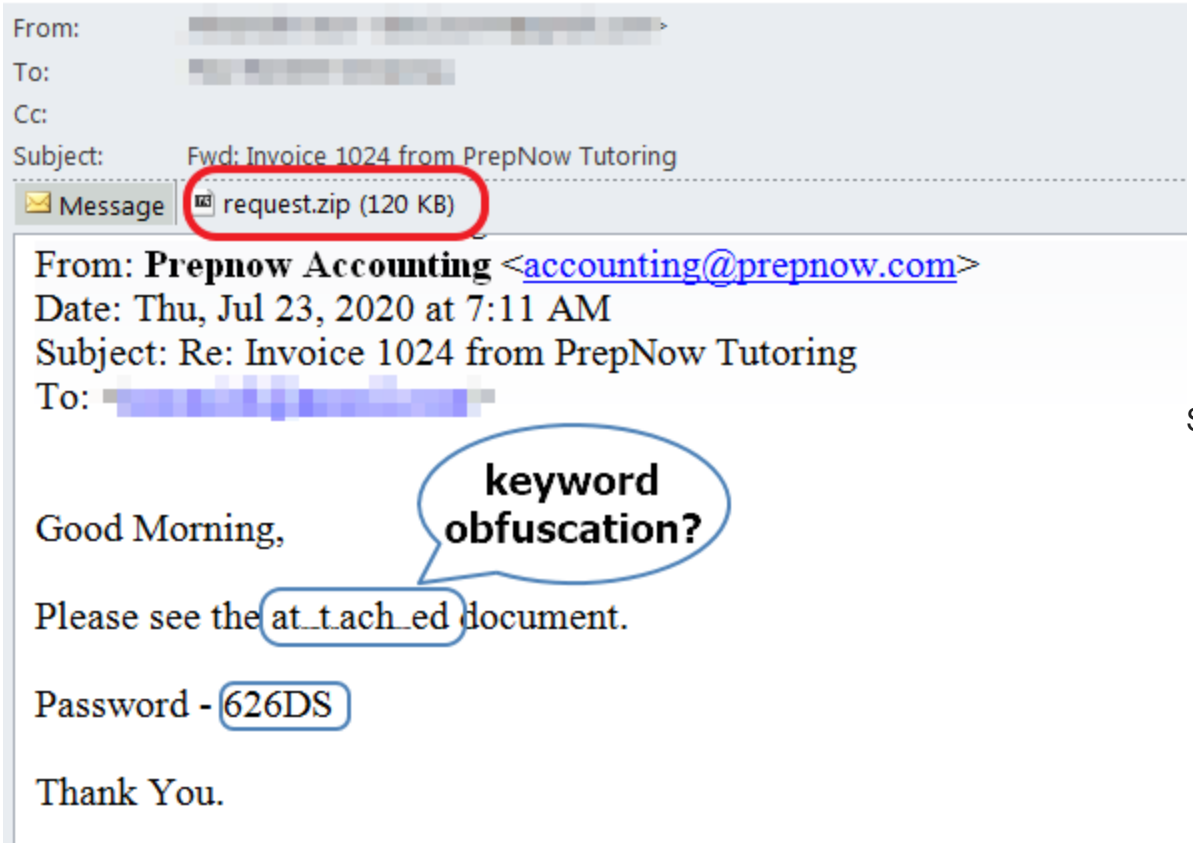
In the current campaign discovered in July 2020, an email phishing campaign is performed using compromised business accounts where the recipients are customers of the same businesses. This makes the phish that much more likely to succeed, given the sender and the recipient have an established business relationship. One example we are going to highlight is from a compromise of PrepNow.com, a private, nationwide student tutoring company with business presence in many states.

The phishing emails are sent to potential victims from the accounting department and purported to include an invoice. The attachment is a password protected zip file named request.zip. The password protection is to prevent anti-malware analysis solutions from decrypting and inspecting the attachment. The password is included in the email message body, in the hopes that the victim would read the email, locate the password and use it to open the attached file.

An interesting characteristic of these messages is the word “attached” is obfuscated in multiple ways. This may be an attempt for this phish to bypass spam filters or phishing detection systems that could be looking for such keywords. However, this is useless because there is no need for any security solution to rely on the word “attached” to figure out there is an attachment. If anything, we expected the obfuscation to obfuscate the word “password” because that’s a tell-tale sign of something phishy going on. Then again, modifying the body of the email ever so slightly may change some fuzzy hashes email security solutions calculate to identify bulk email campaigns.

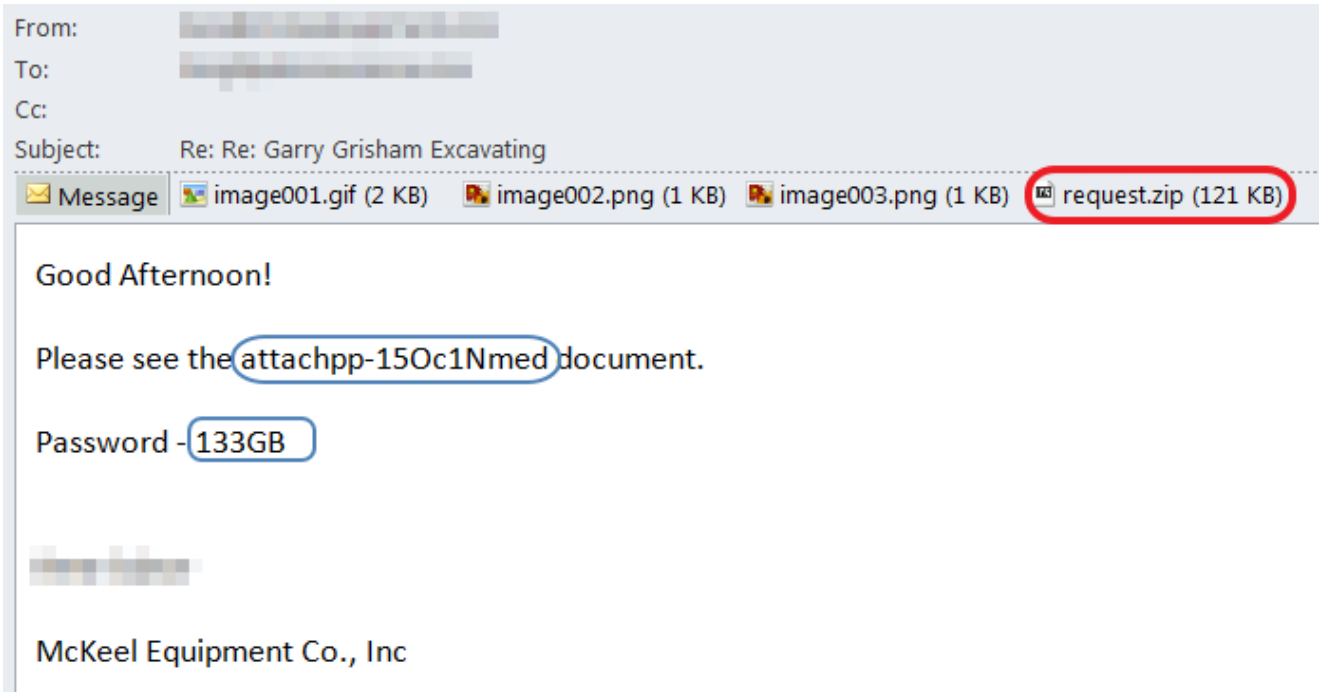
Additionally, the campaign has rotated the file name used for the attachment inside the zip file. Again, this seems futile, since the password protection should prevent most security solutions from opening and inspecting the content.

Nonetheless, this technique proved successful against Google’s Gmail security, which did not block this email.

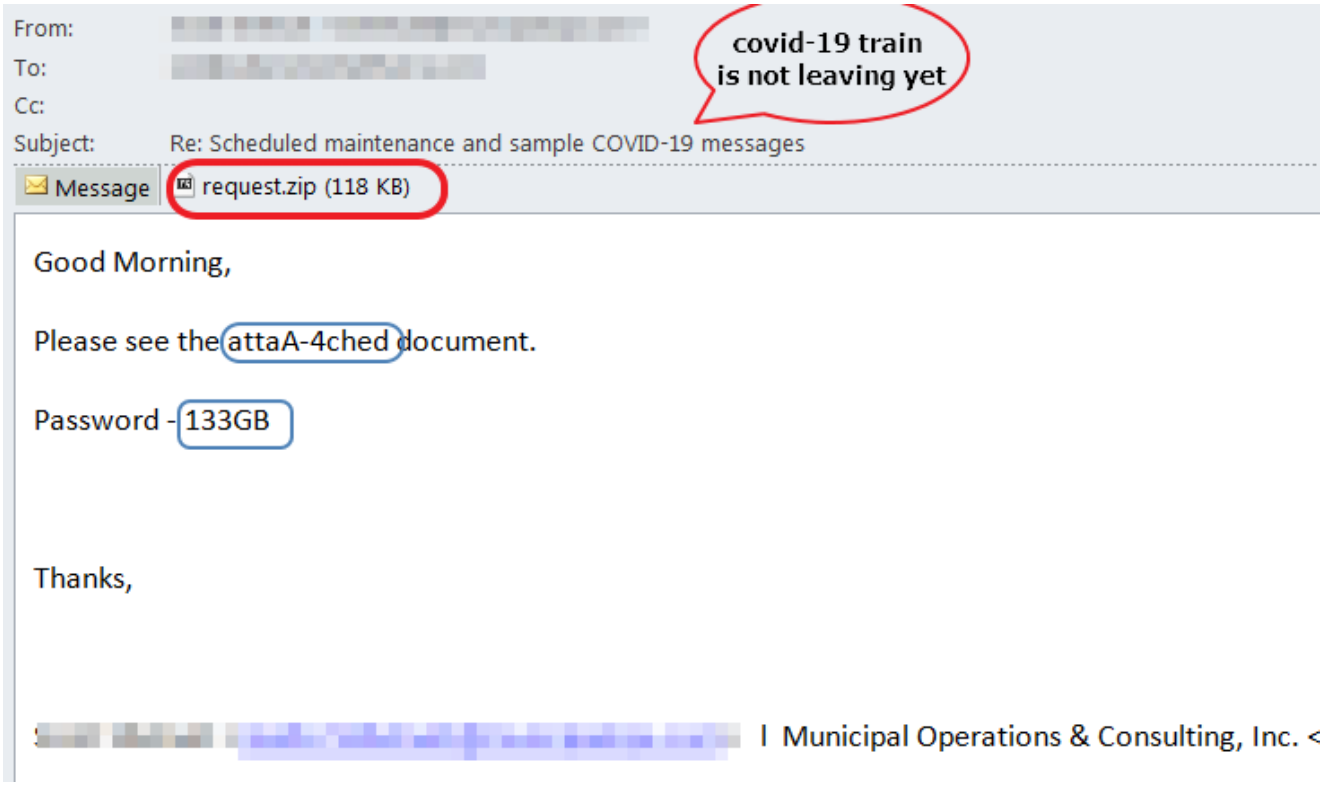


Sample email containing the password protected Request.zip sha256:

2beadfb91e794860aad159dcca1c94855a99b9bc908d03d10cea005dad652422 MS Word Document inside zip file: legal paper_07.23.2020.doc: Sha 256: 9b0ff58ddedd7a78e3b8f28c9c5a4934ea9f4dc530d57cc7715bdca6687590fc



Another example showing a slightly different obfuscation of the word “attached” Request.zip (sha256): d80dc6c07eedf0cbccedf9427accef8bcb067b9dc1eaf4f81b9ee968854eb176 legal agreement.07.20.doc (inside request.zip) dc6452b6b0683223c0d87970c600ebbda3ed6c4dab14649beff12be59842f59c

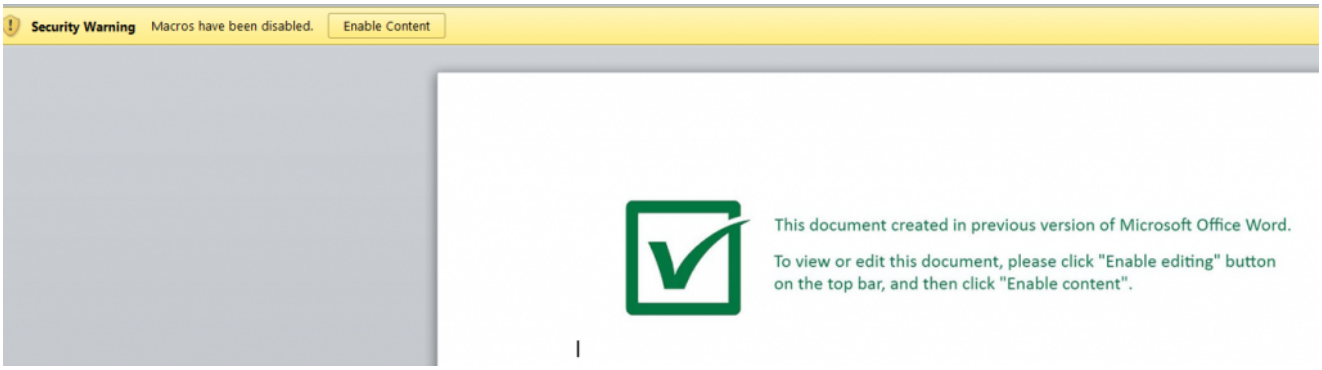


Yet another sample email with a third way of obfuscating the word attached Request.zip (sha256):

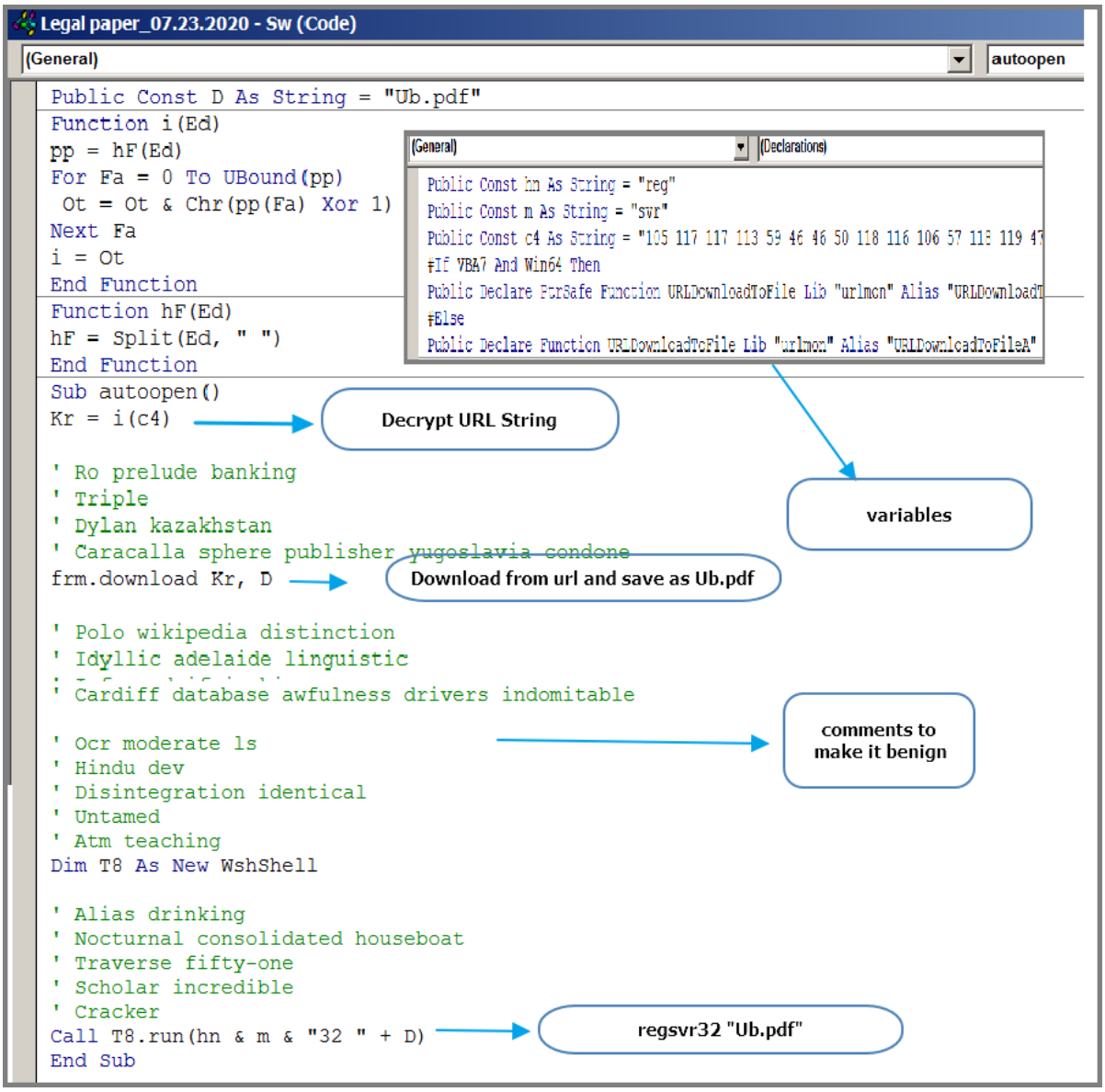
78fd08878d1f5025ecf7dcf1f0460a4d00f7c50ea281b35c190cd3f8aecf61af
Question_07.20.doc (inside request.zip)
469fc41ba6d15f2af6bcf369e39c5c06b8bb5d991c008efadbfd409d096e911b
Let's take a look at the malicious documents in the attachments.

First Stage: MS Office Documents Macro Downloader

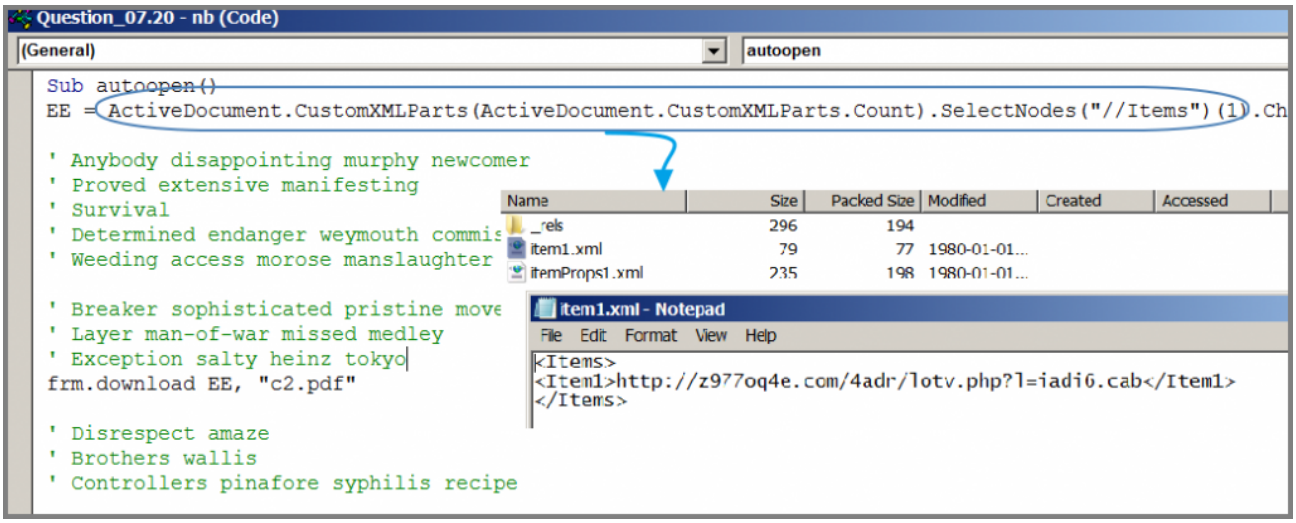
In short, once the zip file is expanded, the user finds a Microsoft Word document that contains a macro that executes upon opening the document. There is the usual social engineering attempt to get victims to enable macros, which claims the document was created with a previous version of MS Word, in this case. Once macros are enabled, the VB script will download a DLL, save it as a PDF and install it as a service using regsvr32 to guarantee persistence.



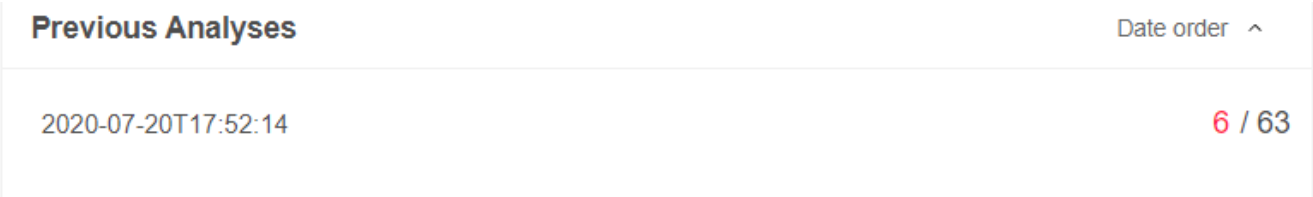
The authors have resorted to being "minimalist" in this recent campaign. The "macro" code is very simple and straightforward but they managed to add a few tricks to evade detection. For instance, all strings and function calls in the macro are obfuscated.



There are also instances where the URL is saved as an XML file inside the document.



To some extent, these few tricks worked. Virustotal hits were low at first submission on the samples from July 20.



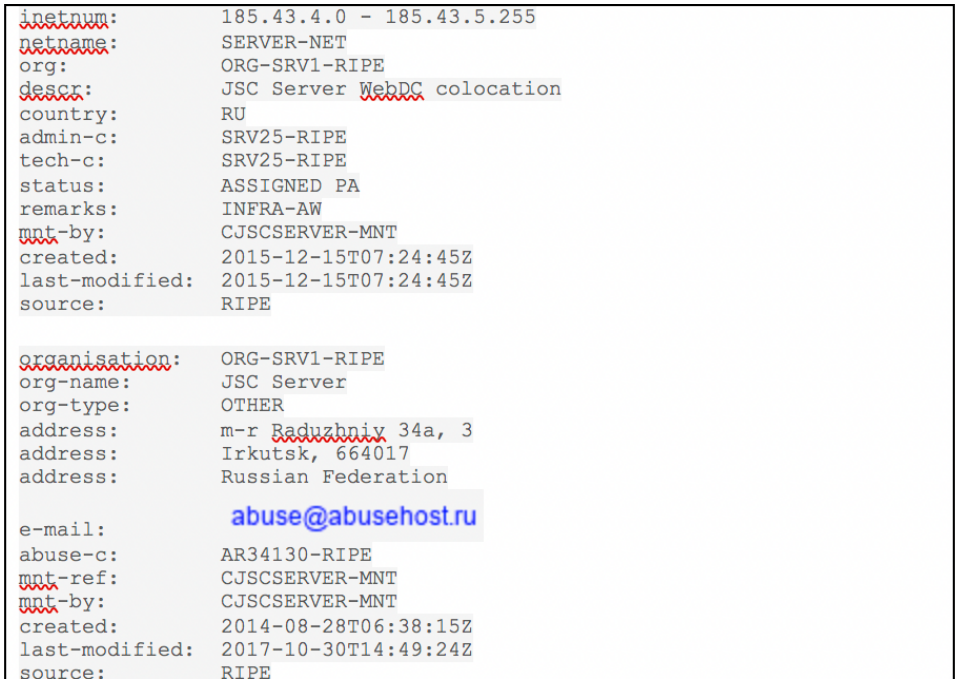
Sha256: 469fc41ba6d15f2af6bcf369e39c5c06b8bb5d991c008efadbfd409d096e911b
Source: virustotal.com

Sha256: dc6452b6b0683223c0d87970c600ebbda3ed6c4dab14649beff12be59842f59c

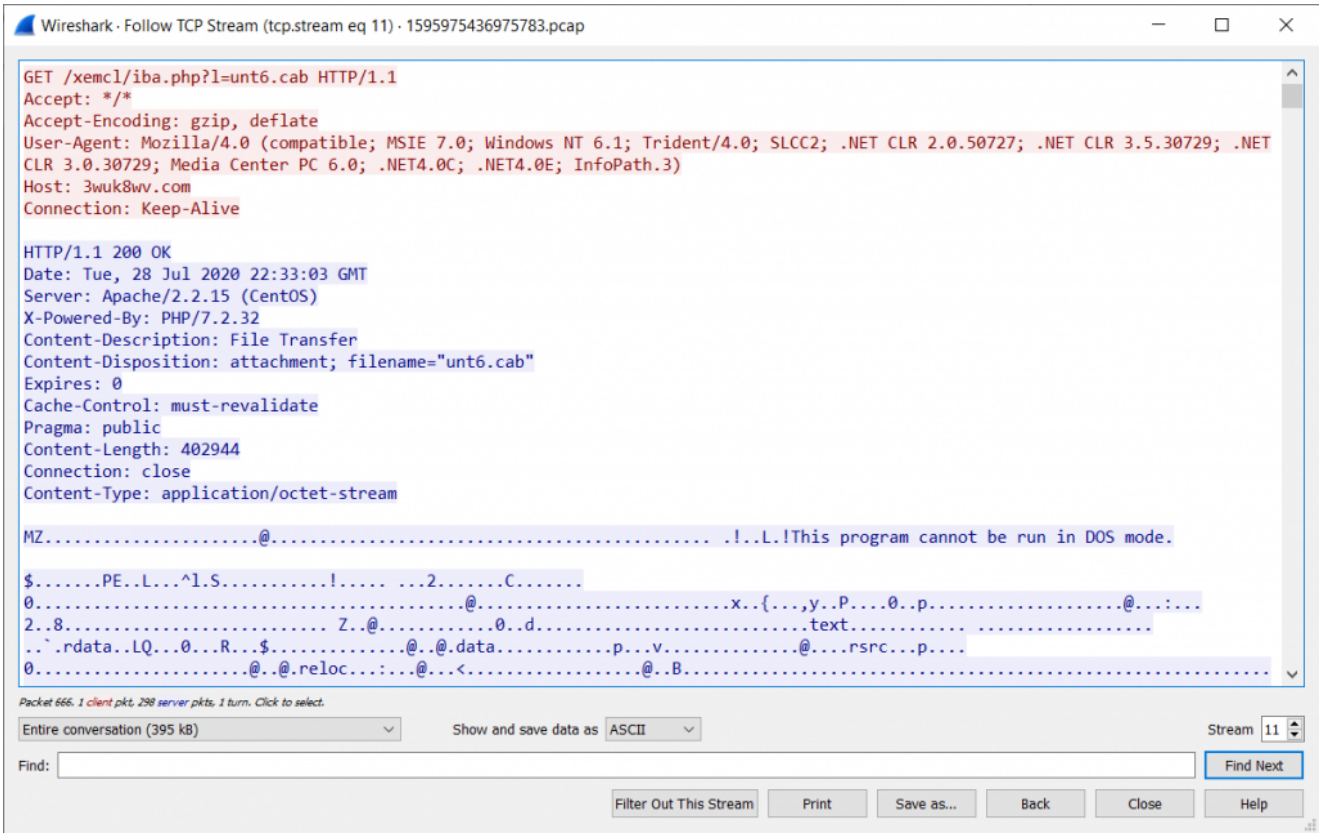
Second Stage: DLL Trojan

In our observation, the second stage payload consists of a DLL that is downloaded from 3wuk8wv[.]com or 185.43.4[.]241, which is hosted on a

hosting provider in Russian Siberia https://ispserver.com/

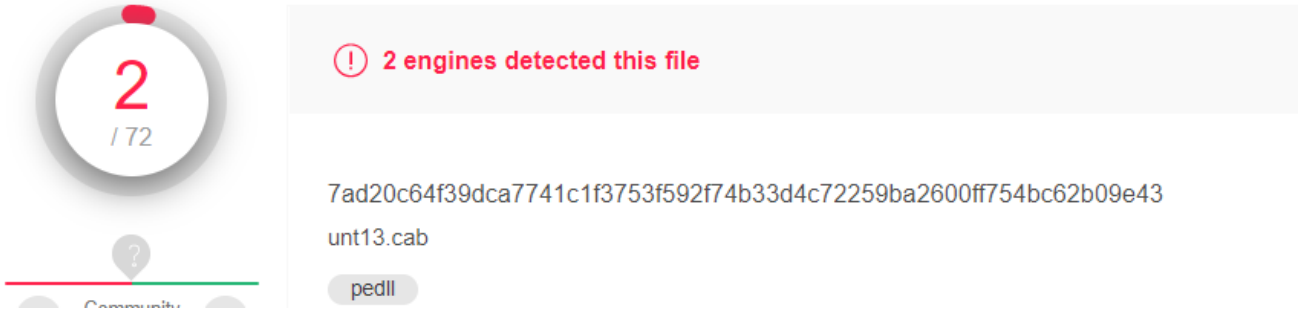


Once downloaded, the malicious DLL is saved as a pdf file, then the macro executes it via a call to regsvr32.exe.



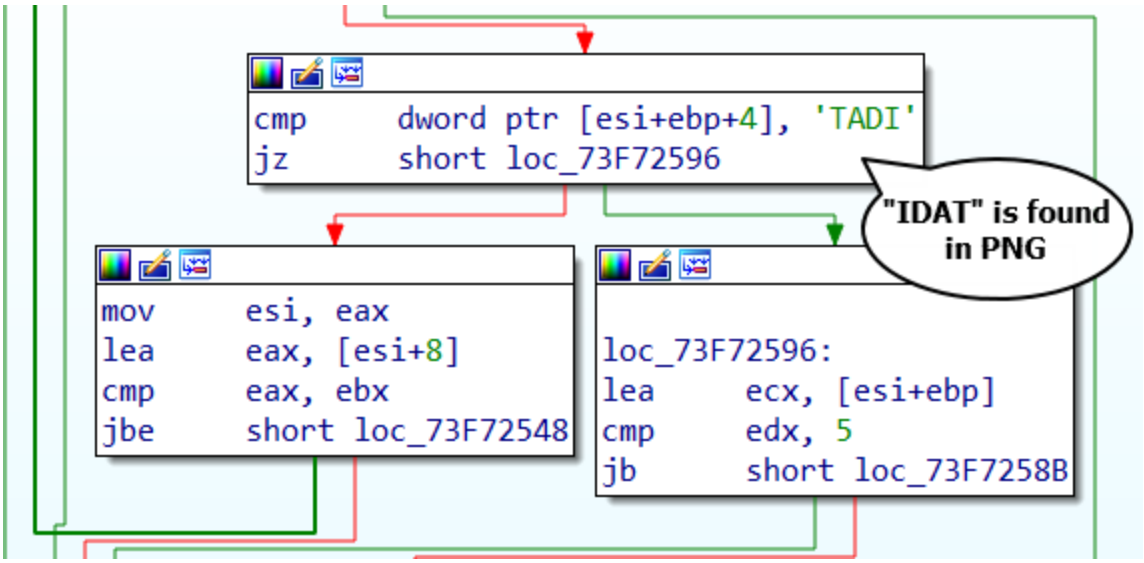
Pcap capture of downloading of DLL

Our sample has very few detection on Virustotal, upon initial submission.



Third Stage: Malicious Payload Downloader

Once launched, the DLL will download the next stage from the domain loadhnichar[.]co as a PNG file and decrypt it. Similar to the second stage loader we analyzed in our previous blog, this loader blends its traffic with requests to benign domains, such as apple.com, twitter.com, microsoft.com, etc. to look more benign to sandboxes trying to analyze it.



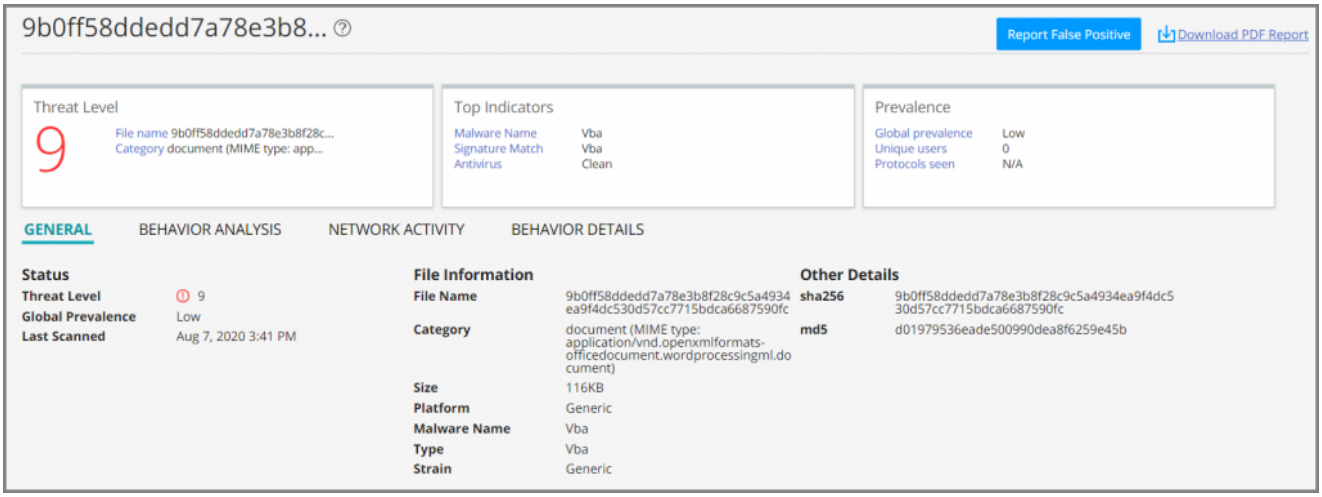
Unfortunately, at the time of our testing, the download domain, loadhnichar[.]co for the next stage is already down.

```
Standard query 0xf801 A support.oracle.com
Standard query 0xa460 A support.apple.com
Standard query 0x054f A loadhnichar.co
Name query NB LOADHNICHAR.CO<00>
Name query NB LOADHNICHAR.CO<00>
Name query NB LOADHNICHAR.CO<00>
Standard query 0xd4da A help.twitter.com
Standard query 0xdb0d A support.microsoft.com
```

Using a similar sample from malware-traffic analysis, https://malware-traffic-analysis.net/2020/07/20/index.html, we analyzed the next stages.

We have not found any changes from this stage, compared to our previous analysis. The second stage will download the third stage as a PNG file, decrypt it and run it. It will be saved as {random}.exe and will create a scheduled task for persistence. The third stage will download the IcedID main module as a PNG file, spawn a msiexec.exe process and inject the IcedID main module into it.

Juniper Advanced Threat Prevention (ATP) detects this file as malware.



Indicators of Compromise

sha256	Notes
2beadfb91e794860aad159dcca1c94855a99b9bc908d03d10cea005dad652422	request.zip
d80dc6c07eedf0cbccedf9427accef8bcb067b9dc1eaf4f81b9ee968854eb176	request.zip
78fd08878d1f5025ecf7dcf1f0460a4d00f7c50ea281b35c190cd3f8aecf61af	request.zip
9b0ff58ddedd7a78e3b8f28c9c5a4934ea9f4dc530d57cc7715bdca6687590fc	doc
dc6452b6b0683223c0d87970c600ebbda3ed6c4dab14649beff12be59842f59c	doc
469fc41ba6d15f2af6bcf369e39c5c06b8bb5d991c008efadbfd409d096e911b	doc
3wuk8wv[.]com	2nd stage
185.43.4[.]241	2nd stage

*Special thanks to **Alexander Burt** and **Mounir Hahad** from Juniper Threat Labs for participating in the analysis and writing of this blog.*