Scumbag Combo: Agent Tesla and XpertRAT

labs.k7computing.com/

By K7 Labs December 18, 2018

Unity is strength – this age old adage is true for just about everyone, even the bad guys.

It has become a common practice for threat actors to work in tandem for various reasons, viz. better chances of evading detection, increased magnitude or sophistication of the attack, etc., all of which are means to higher ill-gotten gains. And the availability of (malicious) source code on popular platforms like GitHub, Pastebin, etc. only makes life easier for these cyber criminals.

With this blog post we are going to explain one such recent “collaboration” which we would like to dub “The Scumbag Combo”, a true story of two malware families coming together to victimize the innocent and vulnerable.

First, an introductory pictorial representation of the infection flow (Figure 1) before going into the morbid details.

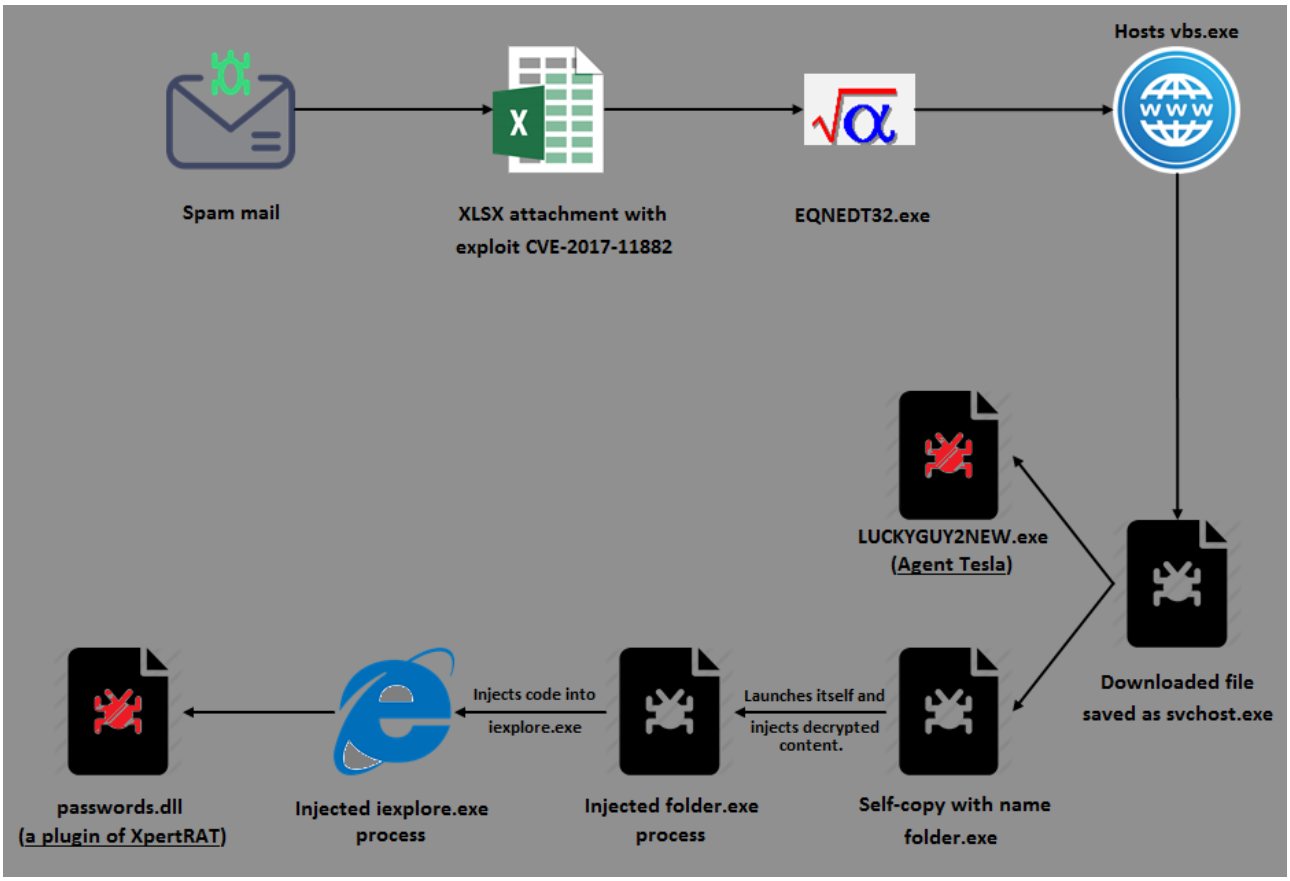


Figure 1: Infection flow

It all starts with a spam email containing an XLSX attachment that exploits the Microsoft Equation Editor's remote code execution vulnerability (CVE-2017-11882) to download the file *vbs.exe* hosted on an open directory (Figure 2), save it as *svchost.exe* under *%AppData%* directory and automatically execute it. That covers half the picture and is fairly standard stuff, but then the rest gets pretty interesting.

Figure 2: Open directory

On execution, this fake *svchost.exe* decrypts the code responsible for the delivery of the aforementioned scumbags into allocated heap memory, and transfers the control to it (Figure 3).

```
0045F779  .  6A 40          PUSH 40                  Protect = PAGE_EXECUTE_READWRITE
0045F77B  .  68 00300000    PUSH 3000                AllocationType = MEM_COMMIT|MEM_RESERVE
0045F780  .  68 1E5C0000    PUSH 5C1E                Size = 5C1E (23582.)
0045F785  .  6A 00          PUSH 0                   Address = NULL
0045F787  .  E8 006AFAFF    CALL <JMP.&kernel32.VirtualAlloc>   VirtualAlloc
0045F78C  .  8945 FC        MOV DWORD PTR SS:[EBP-4],EAX
0045F78F  .  90             NOP
0045F790  .  33F6           XOR ESI,ESI
0045F792  .  33C0           XOR EAX,EAX

0045F7B0  .  03CE           ADD ECX,ESI
0045F7B2  .  894D F8        MOV DWORD PTR SS:[EBP-8],ECX
0045F7B5  .  85C0           TEST EAX,EAX
0045F7B7  .v 75 14          JNZ SHORT 0045F7CD
0045F7B9  .  8A0A           MOV CL,BYTE PTR DS:[EDX]
0045F7BB  .  80F1 DC        XOR CL,0DC
0045F7BE  .  884D EF        MOV BYTE PTR SS:[EBP-11],CL
0045F7C1  .  90             NOP
0045F7C2  .  8B4D F8        MOV ECX,DWORD PTR SS:[EBP-8]
0045F7C5  .  8BF9           MOV EDI,ECX
0045F7C7  .  8A4D EF        MOV CL,BYTE PTR SS:[EBP-11]
0045F7CA  .  880F           MOV BYTE PTR DS:[EDI],CL
0045F7CC  .  46             INC ESI
0045F7CD  >  90             NOP
0045F7CE  .  90             NOP
0045F7CF  .  42             INC EDX
0045F7D0  .  4B             DEC EBX
0045F7D1  .^ 75 CB          JNZ SHORT 0045F79E
0045F7D3  .  90             NOP
0045F7D4  .  68 D70C0000    PUSH 0CD7
0045F7D9  .  5F             POP EDI
0045F7DA  .  90             NOP
0045F7DB  .  90             NOP
0045F7DC  .  037D FC        ADD EDI,DWORD PTR SS:[EBP-4]
0045F7DF  .  90             NOP
0045F7E0  .  FFE7           JMP EDI
```

decryption routine

Jump to decrypted code

Figure 3: Decryption routine

This decrypted code then continues to construct an import table for APIs to be used later. Additionally, it also checks for the presence of malware analysis and debugging tools (Figure 4), as well as anti-malware processes (Figure 5).

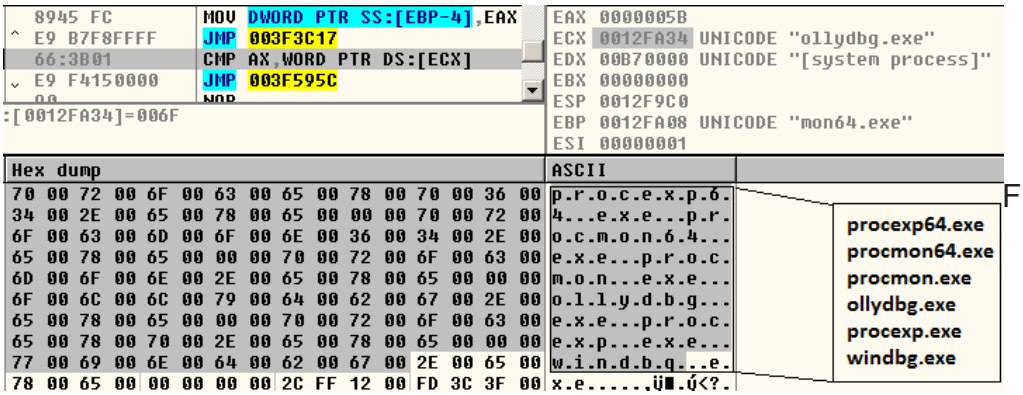


Figure 4: Malware analysis and debugging tools

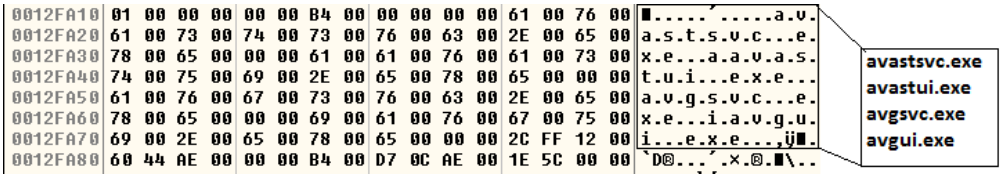


Figure 5: Anti-malware processes

It further looks for the following anti-malware processes:

- **avp.exe**
- **bdwtxag.exe**
- **bdagent.exe**
- **dwengine.exe**
- **avastui.exe**

If any of the aforementioned processes are found it terminates itself.

If suitably assuaged, it continues to create a folder called "*folder*" under *%AppData%* and copies itself to this location as *folder.exe* (Figure 6).

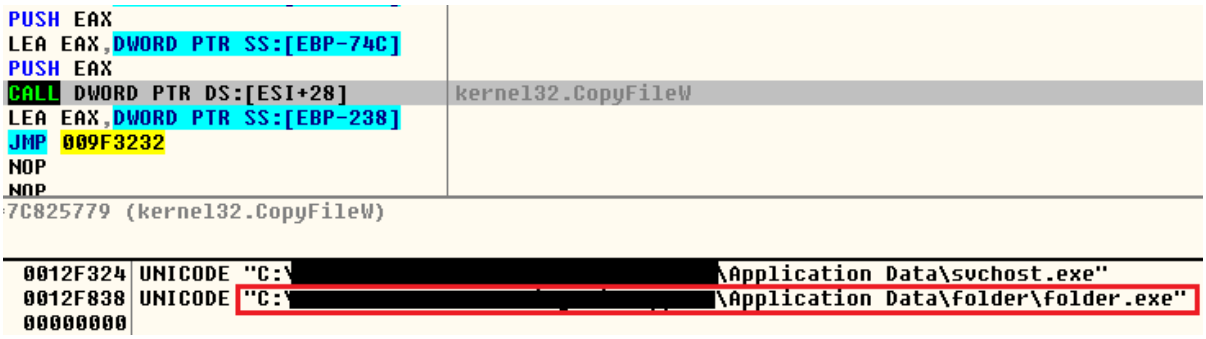


Figure 6: Self-copy as folder.exe

As the next step it decrypts a PE file *LUCKYGUY2NEW.exe* (Figure 7) into allocated heap memory, drops it under the *%temp%* folder, and executes it using the API *ShellExecuteW*.

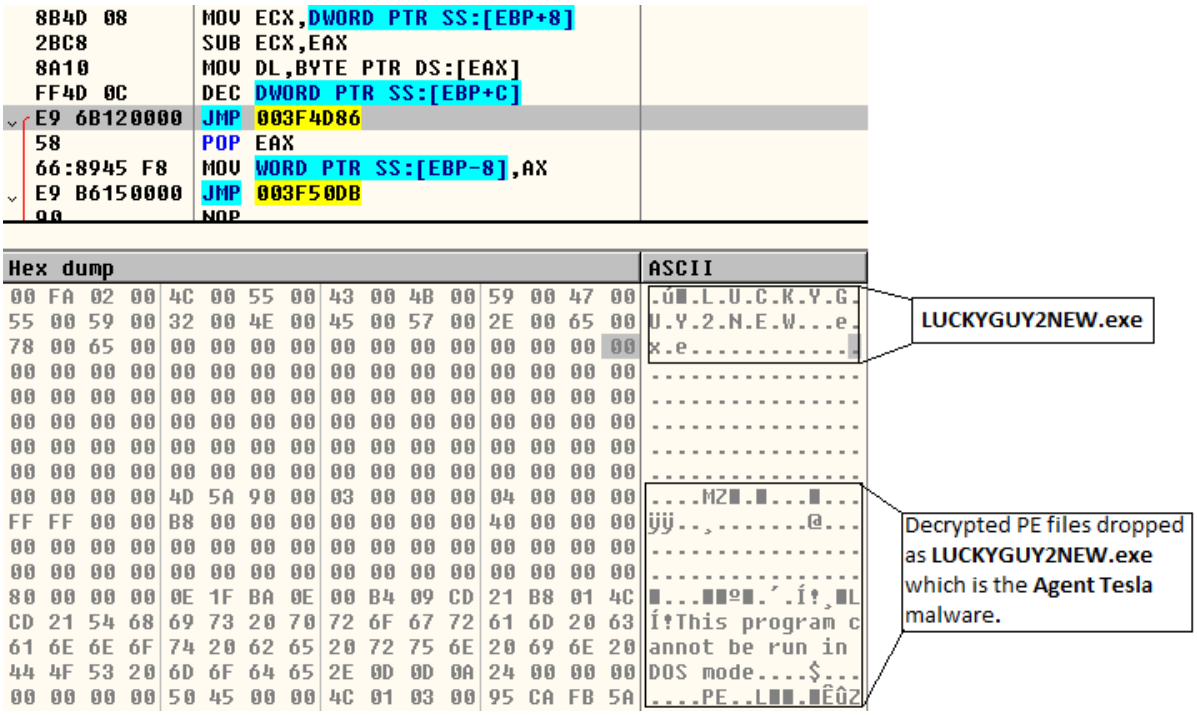


Figure 7: Decrypting LUCKYGUY2NEW.exe

This binary, *LUCKYGUY2NEW.exe*, which is found to be an MSIL file, is the first of the scumbag duo to get onto the compromised system: *Agent Tesla*. It has keylogging, screen and video capturing, and password stealing capabilities. The password stealing module can extract saved passwords (Instagram, Twitter, Gmail, Facebook, etc.) from various browsers (Figure 8), mails and FTP clients.

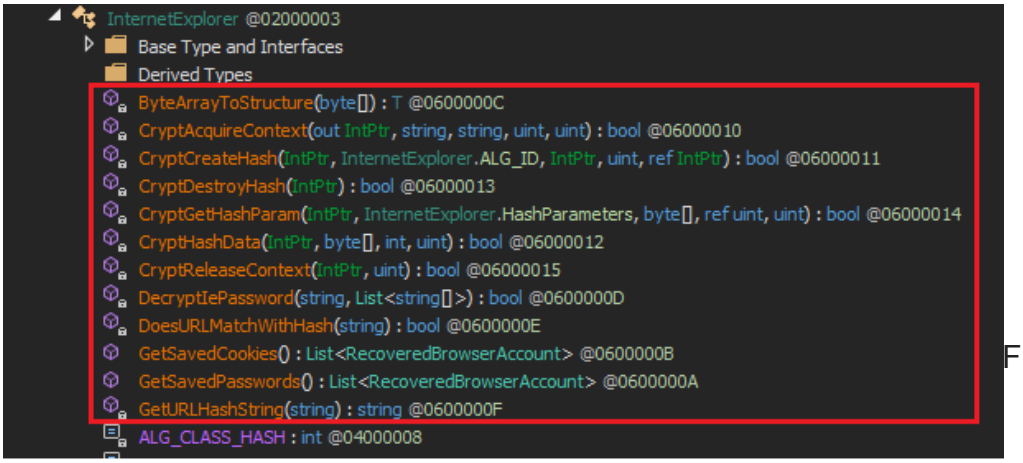


Figure 8: MSIL

methods used for stealing passwords

Having delivered the Agent Tesla component, svchost.exe goes on to execute its copy *folder.exe* from within *%AppData%\folder,* which orchestrates the dramatic entry of the second protagonist of the scumbag show: *XpertRAT*. After executing *folder.exe*, the *svchost.exe* process gets terminated.

Note, persistence of *folder.exe* is handled by a VB script *folder.vbs* dropped in the Startup directory (Figure 9).

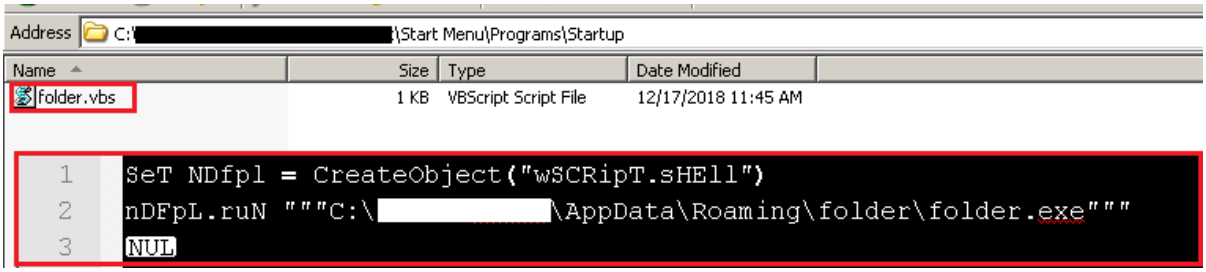


Figure 9: VBS in Startup folder

folder.exe does a redundant check for traces of the same set of malware analysis/debugging tools and anti-malware processes as depicted in Figures 4 and 5 above.

Next it decrypts yet another PE file in yet another blob of heap memory. And if you think that this is the *XpertRAT* component, well, you are plain wrong. Dumping the file from memory revealed it to be a Visual Basic compiled binary which injects into a legitimate Microsoft Internet Explorer (iexplore.exe) process.

folder.exe then creates another *folder.exe* process in a suspended state, injects the decrypted Visual Basic binary into it and resumes the thread (Figure 10). By the way, what's with these guys and the word "folder"?! No imagination. Sheesh!

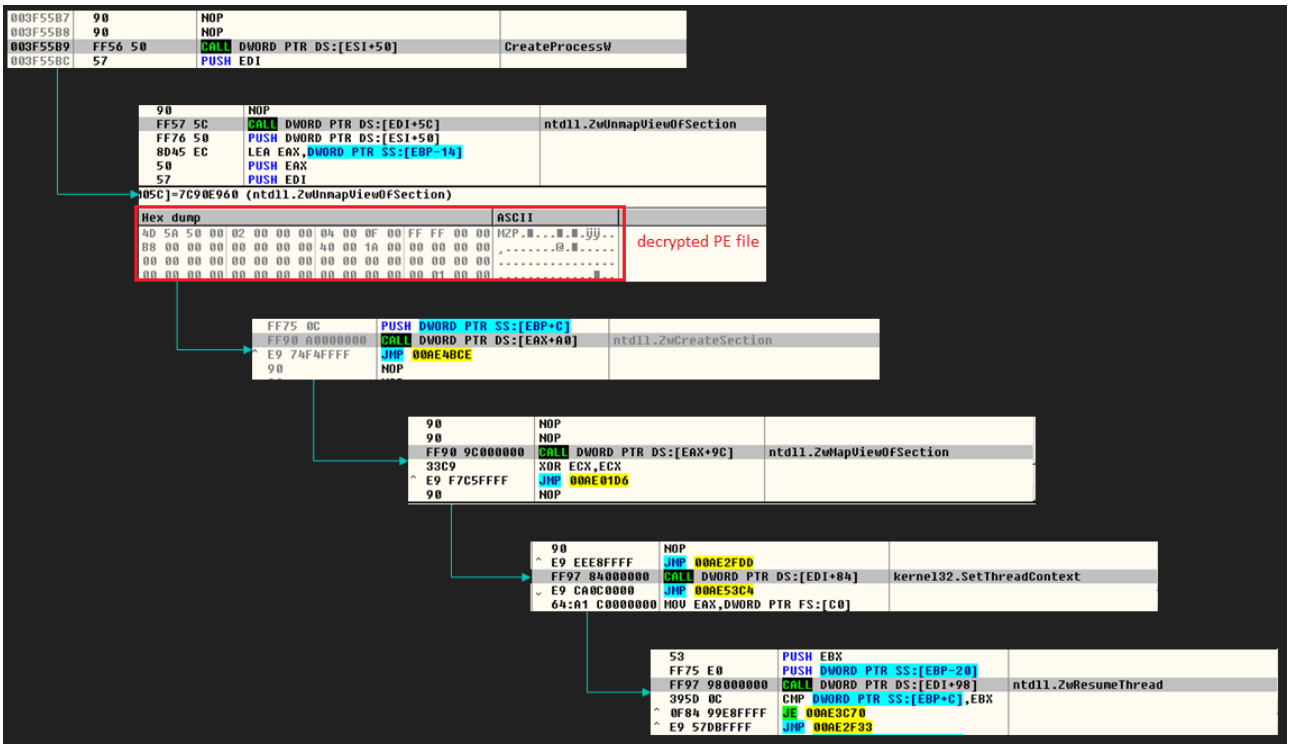


Figure 10: Injection of the latest decrypted binary

Once the injected process begins executing, it spawns the legitimate *iexplore.exe* process in a suspended state, injects its own code into it and resumes the thread. This then connects to a Command and Control server (C&C or C2) to which it sends the compromised system information (Figure 11), and requests for the Remote Access Trojan (RAT) component – *XpertRAT*.

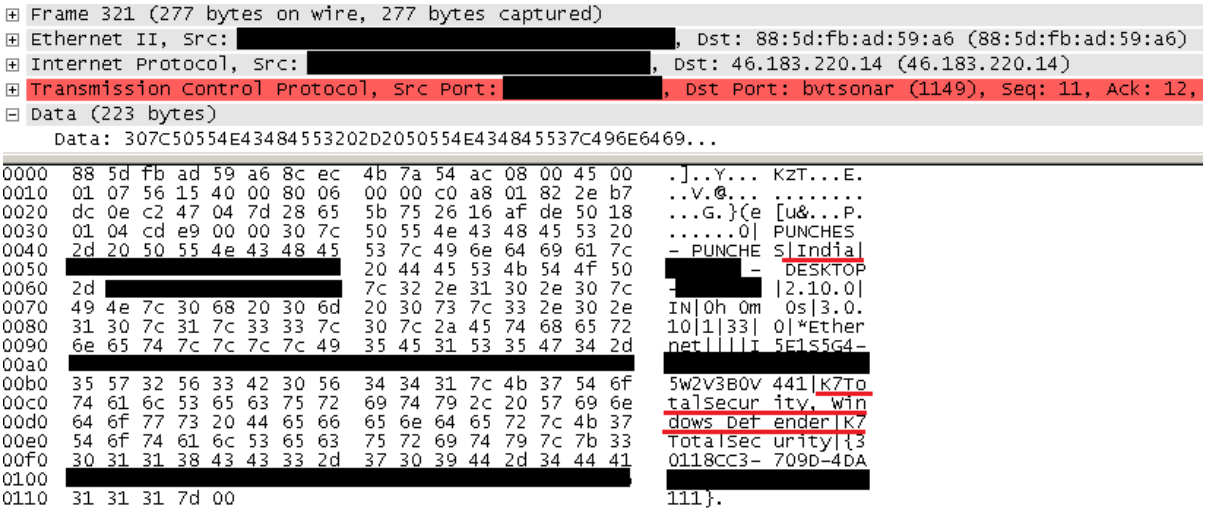


Figure 11: C&C communication (compromised system information)

The C&C server, after validating the information from the compromised system, will respond with the RAT component – *passwords.dll*, an *XpertRAT* plugin as depicted in Figure 12.

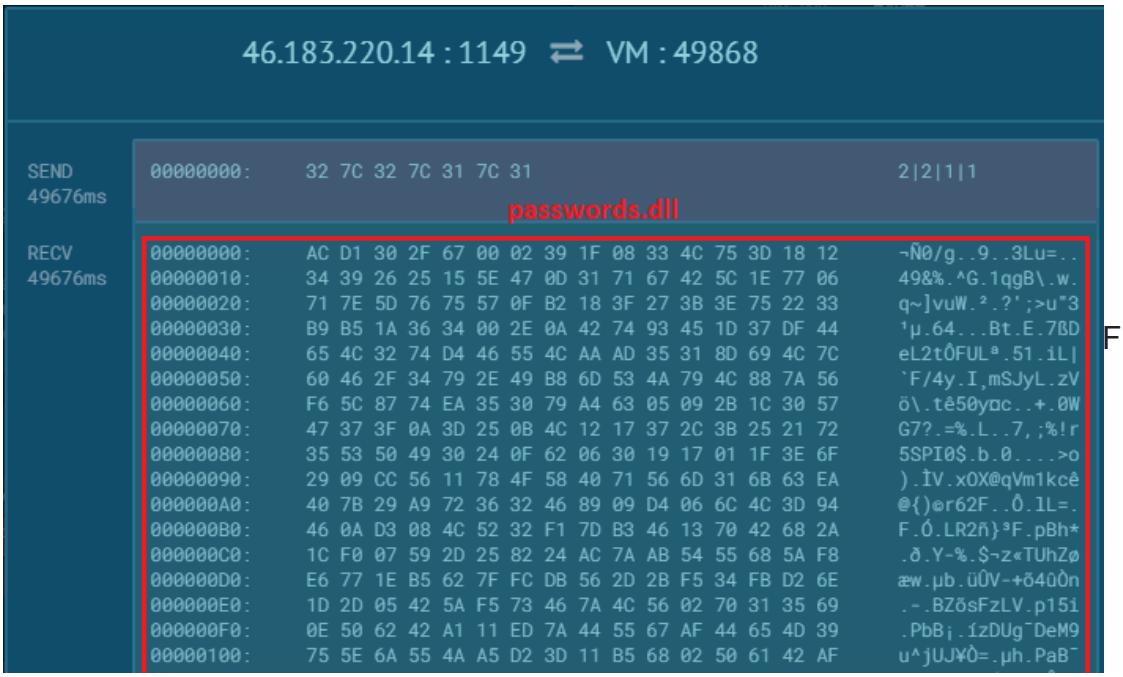


Figure 12: The XpertRAT plugin – image courtesy app.any.run

This plugin is used to retrieve all the usernames and passwords (Instagram, Twitter, Gmail, Facebook, etc.) stored in various browser caches and emails on the compromised system, which may then be stored in a text file to be either dispatched to the C&C or accessed remotely.

Lo and behold, all the actors are now on stage.

But worry not K7 users, for as always, we have you covered at every single layer of this attack! 🙂

Security Guidelines

- Install the latest service packs & hotfixes from Microsoft and enable automatic update/notification for patches on Windows.
- Cultivate the usage of spam filters.
- Do not open any email attachment that looks suspicious or that you weren't expecting.
- Check the email and make sure it is not spoofed before downloading and opening any attachments.
- Upgrade all applications to the latest • stable versions.
- Install, enable and regularly update reliable security software such as K7 Total Security.

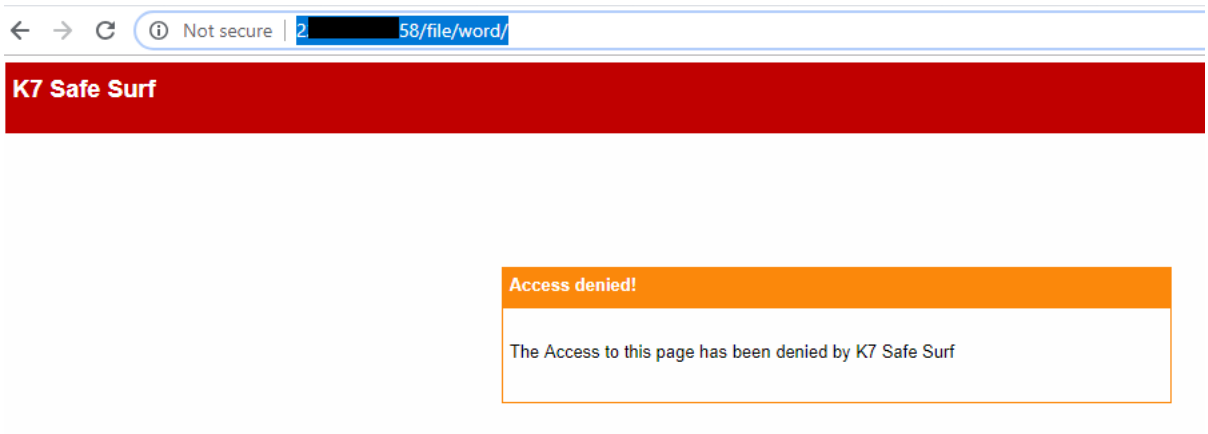
Indicators of Compromise (IoCs)

Files:

Hash	Component	K7 Detection

Hash	Component	K7 Detection
528D53B945516C8F18C63C5B8DF4695E	XLSX attachment	Trojan (0001140e1)
E0374BCC3615F00CDD9C9E3845A1EB74	svchost.exe / vbs.exe	Riskware (0040eff71)
88A93172E9BB75CE8638C36FF744BE55	LUCKYGUY2NEW.exe	Trojan (0052d5341)
9F9C272BF3372F6EE920DEAA00926689	folder.vbs	Trojan (0001140e1)
5C3E2E94AF5622A06D06EAC83CFA4C2B	VB file dumped from memory	Trojan (004be7cd1)
2EEC4FEAAD2D41A806A8D3197A4F538B	passwords.dll	Trojan (0001140e1)

URLs:



Dynamic detection:

Behaviour based detection of folder.exe process injection into iexplore.exe