Rhadamanthys: The “Everything Bagel” Infostealer

research.checkpoint.com/2023/rhadamanthys-the-everything-bagel-infostealer/

March 27, 2023

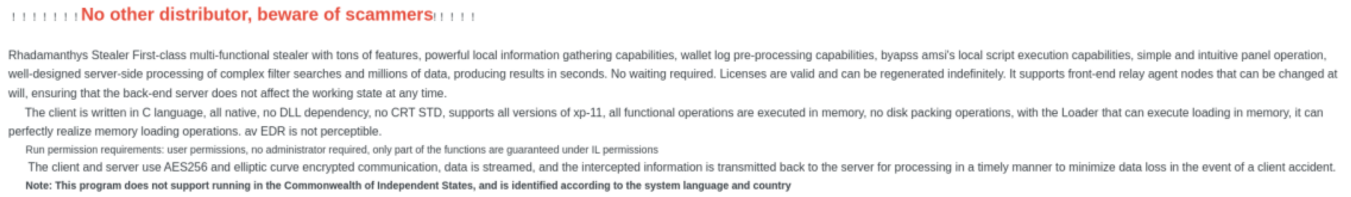
Key Takeaways

- Rhadamanthys is an advanced infostealer which debuted on the dark web in September of last year to a warm critical reception by cybercriminals.
- A maximalist approach to features: functionality is added for its own sake, never mind the effort required or expected payoff.
- Campaigns by default target countries indiscriminately, excluding the commonwealth of independent states. This is typical of this kind of malware.
- Multiple-stage loader/shellcode execution has been researched in prior publications and has made it difficult to reach a proper interactive disassembly workflow with the actual information-stealing logic.
- We provide highlights of the Dark Web ‘buzz’ surrounding this malware.
- We share telemetry insights which confirm that by the nature of how the malware is used, large orgs are also being subjected to incidental drive-by attacks that have a theoretical potential to escalate.
- We present a method of forensically resolving API calls of homebrew function tables in “orphaned” memory dumps from concluded sandbox runs, using the in-memory addresses alone.
- We use this method to convert a memory dump of Rhadamanthys information stealing code into a workable interactive disassembly database with resolved API calls, and showcase the newly available level of analysis by presenting a step-by-step disassembly breakdown of how the malware compiles its own database of stolen Google Chrome information in order to send back to the C2 server.

Background

What causes a man to wake up one day and say, “I’m going to build my own malware and go sell it to cybercriminals on the dark web”? After all, the market is saturated with competitors, and the product is judged on the one sole metric of how many victims it has successfully parted with their funds and personal data. Statistically, during the past 5 years, someone must have created what would have been the great malware strain to stun the entire industry, but the first two criminals to actually try out the thing had a weak spam game, got weak results, and that was that.

All this must have been acutely clear to an individual who on September 24, 2022, under the alias “kingcrete2022”, posted the following to the appropriate channels:


! ! ! ! ! ! ! No other distributor, beware of scammers ! ! ! ! !

Rhadamanthys Stealer First-class multi-functional stealer with tons of features, powerful local information gathering capabilities, wallet log pre-processing capabilities, byapss amsi's local script execution capabilities, simple and intuitive panel operation, well-designed server-side processing of complex filter searches and millions of data, producing results in seconds. No waiting required. Licenses are valid and can be regenerated indefinitely. It supports front-end relay agent nodes that can be changed at will, ensuring that the back-end server does not affect the working state at any time.

The client is written in C language, all native, no DLL dependency, no CRT STD, supports all versions of xp-11, all functional operations are executed in memory, no disk packing operations, with the Loader that can execute loading in memory, it can perfectly realize memory loading operations. av EDR is not perceptible.

Run permission requirements: user permissions, no administrator required, only part of the functions are guaranteed under IL permissions

The client and server use AES256 and elliptic curve encrypted communication, data is streamed, and the intercepted information is transmitted back to the server for processing in a timely manner to minimize data loss in the event of a client accident.

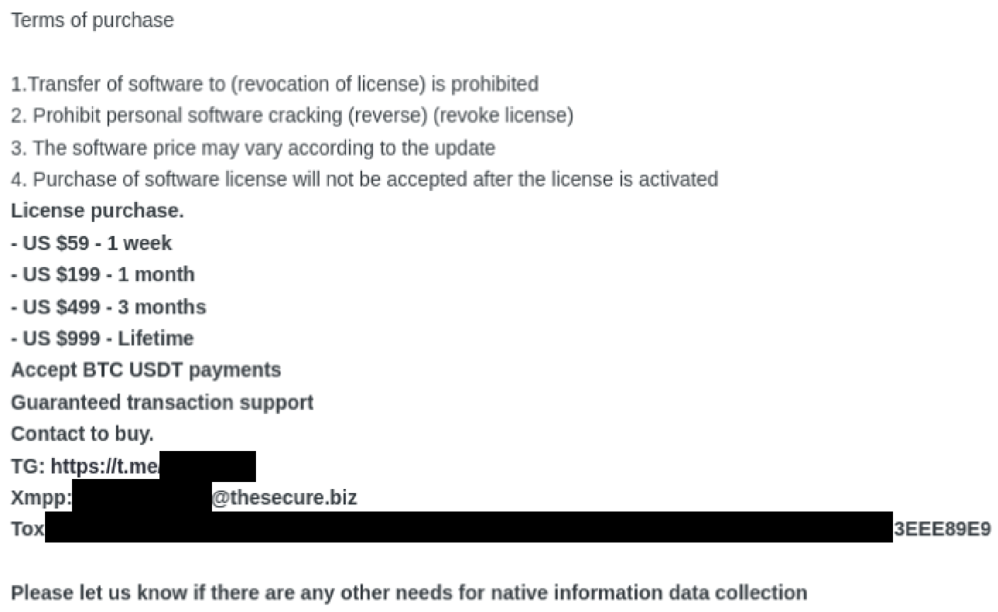
Note: This program does not support running in the Commonwealth of Independent States, and is identified according to the system language and country

The author did not rush into this venture. They had already spent half a year lurking in the forum as "kingcrete2022", and possibly more than that under other aliases. Builds of the malware would later surface that were already polished enough to see the light of day, yet had been compiled a full month before the official launch. It would not be surprising if this person had already spent a long time operating in the cybercrime sphere, even before the debut of the "King Crete" persona – earlier this year Analyst1 published an astoundingly in-depth report on the LockBit gang, with a repeated emphasis on how the ransomware-sphere was much smaller and incestuous than people tend to assume; it's easy to imagine the same incentives at work dictating that one does not go from being a nobody to publishing an offering like the above after just six months of familiarity with the cybercrime landscape, and that this isn't the King's first rodeo.

Following this announcement, KingCrete aggressively went to work, determined to prove that his was the superior product. Curious would-be clients who nonchalantly remarked "seems interesting, I will check it out" earned an immediate response and a request to post a review and follow up with comments. A manic stream of version updates checkered the forum thread for the coming months, adding a very long litany of features, sub-features and sub-sub-features, and providing support in both English and Russian. One update reads, "repaired *major* security vulnerability that the panel session is not affected by password modification"; just "security vulnerability" alone would have technically sufficed, but it doesn't project the same ruthless self-criticism and drive for excellence.

As luck would have it, the first few customers had their campaigns go as planned. "I like this stealer, rep++", said one. "The stealer rocks, stealer and support is just great", said the other. Just between these few early adopters, the campaigns they set up racked up thousands of compromised users, hundreds of thousands of compromised passwords and several hundred compromised cryptocurrency wallets. Some prospective customers had trouble getting their operation running, and the author made sure not to leave them behind, posting: "for those who have difficulties in purchasing VPS or servers, we provide turnkey solutions, please contact us if you need".

Interested?


Terms of purchase

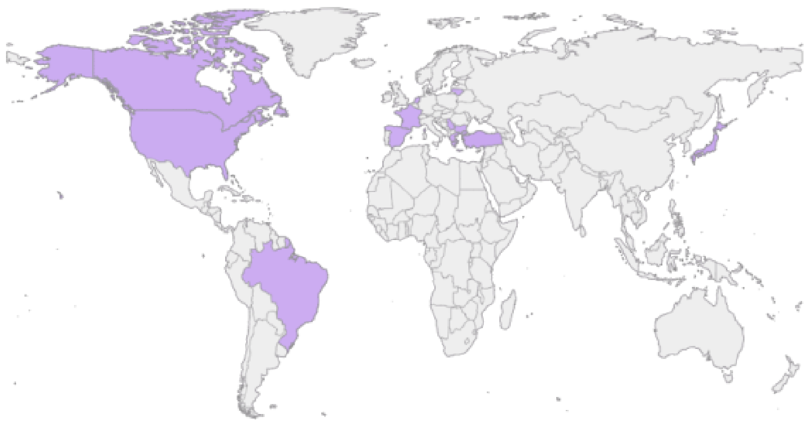
1.Transfer of software to (revocation of license) is prohibited
2. Prohibit personal software cracking (reverse) (revoke license)
3. The software price may vary according to the update
4. Purchase of software license will not be accepted after the license is activated

License purchase.
- US $59 - 1 week
- US $199 - 1 month
- US $499 - 3 months
- US $999 - Lifetime
Accept BTC USDT payments
Guaranteed transaction support
Contact to buy.
TG: https://t.me/[redacted]
Xmpp:[redacted]**@thesecure.biz**
Tox[redacted]**3EEE89E9**

Please let us know if there are any other needs for native information data collection

Victimology

In theory, the author of Rhadamanthys isn't concerned with what you do with the ill-gotten data handed to you by the stealer. Commit fraud, sell the data, start a civil war, it's all the same to him. In practice, the main customers for off-the-shelf malware like this are opportunistic cybercriminals, who aim to infect whomever and whenever ("as long as the target is not located in the commonwealth of independent states", per the author, who certainly did not invent that practice). Campaign victims are therefore spread all around the world, though some spikes are visible where a campaign particularly enjoyed success, and some operators will put their own twist on where and how they aim the infection (one campaign disseminated samples under the guise of video editing software, such as OBS studio, pushed to the crowd of unsuspecting streamers via Google ads).



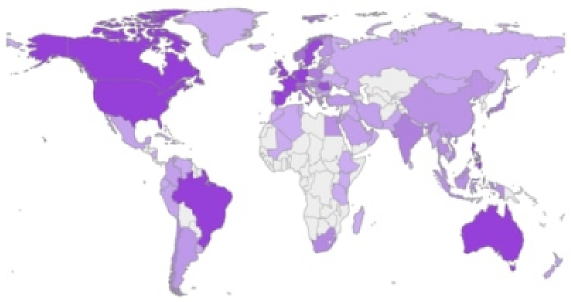


Figure 5 – Two sample campaign heatmaps shared on the forums. The perpetrator of the second campaign, which netted 600,000+ passwords and 900+ cryptocurrency wallet contents, is the one who wrote "I like this stealer, rep++", to which the author responded, "What a great New Year's gift!"

Common wisdom says people who operate this sort of malware are typically not too concerned with "big game hunting" the way the big ransomware gangs are. To them it's a numbers game: rake in many victims and monetize them wholesale. Think of your favorite hack of the past decade, and chances are at no point it came to light that actually the initial breach was due to a trigger-happy cybercriminal spamming Emotet maldocs in every direction, who then suddenly realized one of their twenty thousand victims was a lucrative target. Still, these indiscriminate attacks do end up aimed at major organizations, by sheer statistics; via our telemetry we were able to confirm an attempted Rhadamanthys infection of a government agency in Canada, as well as an energy company in India's infrastructure sector. We like to imagine that even if these attempts had succeeded, they would have led to just two more people having a bad day and canceling their credit cards, and not the more troubling scenario of a hefty sum exchanging hands followed by a champagne bottle being opened someplace geopolitically hostile to India; but there is certainly no guarantee of it.

Functionality Overview

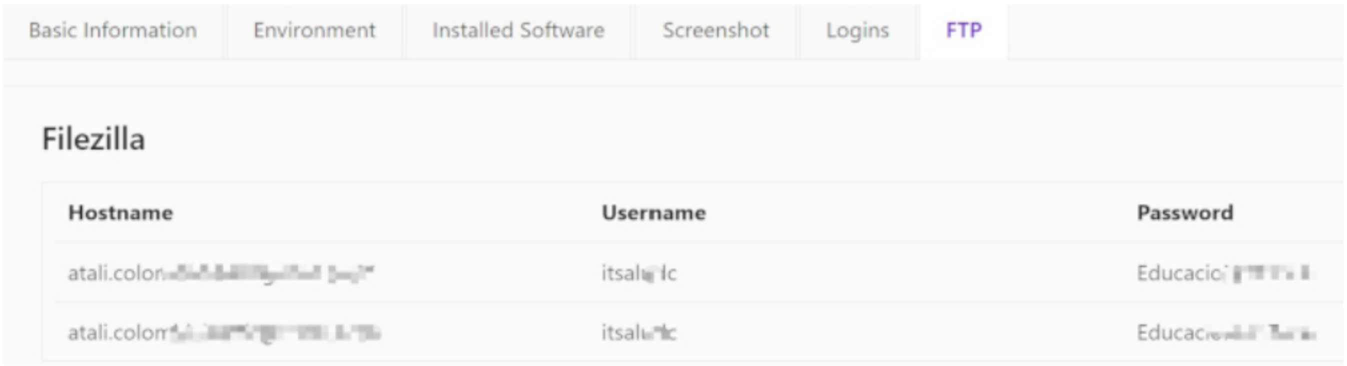
In the award-winning film **Everything Everywhere All At Once**, antagonist Jobu Tupaki delivers the following monologue:

> I got bored one day, and I put everything on a bagel. Everything. All my hopes and dreams, my old report cards, every breed of dog, every last personal ad on craigslist. Sesame Poppy seed. Salt. And it collapsed in on itself. Because, you see, when you really put everything on a bagel, it becomes this. The truth. [..] Nothing matters. Feels nice, doesn't it? If nothing matters, then all the pain and guilt you feel for making nothing of your life, it goes away. Sucked into a bagel.

Having read that, you now understand Rhadamanthys stealer's design philosophy. The mind-numbing list of features included in the initial release speaks plenty for itself already, but we would be remiss not to emphasize the inclusion of – for instance – capabilities for stealing information from **KMeleon** and **Pale Moon** web browsers, which each possess a market share imperceptible to the naked eye; or stealing cryptocurrency from the **Firefox Auvitas** Wallet browser extension, which, as of the writing of these words, has exactly one user. Rhadamanthys' feature list was not hand-picked to maximize return on developer time investment. It resulted from one simple guiding principle: "Add it in! Add it all in!".

Figure 6 – It's too late. You can't look away from it now.

Maybe you are here for the actual list of things that King Crete had put on the bagel. In that pathological case, the list follows below, without bulleted list format so as not to consume too much vertical real estate. Rhadamanthys' feature list includes stealing the victim's system information – **Computer name**, **username**, **ram capacity**, **CPU cores**, **screen resolution**, **timezone**, **geoip**, **environment**, **installed software**, **screenshots**, **cookies**, **history**, **autofill**, **saved credit cards**, **downloads**, **favorites** and **extensions**; it steals credentials from FTP clients – **Cyberduck**, **FTP Navigator**, **FTPRush**, **FlashFXP**, **Smartftp**, **TotalCommander**, **Winscp**, **Ws_ftp** and **Coreftp**; and from Mail clients **CheckMail**, **Clawsmail**, **GmailNotifierPro**, **Mailbird**, **Outlook**, **PostboxApp**, **Thebat!**, **Thunderbird**, **TrulyMail**, **eM** and **Foxmail**; It steals credentials from 2FA applications and password managers **RoboForm**, **RinAuth**, **Authy** and **KeePass**; VPN services including **AzrieVPN**, **NordVPN**, **OpenVPN**, **PrivateVPN_Global_AB**, **ProtonVPN** and **WindscribeVPN**; Note-taking applications including **NoteFly**, **Notezilla**, **Simple Stick Notes** and **Windows Sticky notes**; Message history from messenger applications including **Psi+**, **Pidgin**, **tox**, **Discord** and **Telegram**; also, it steals victim credentials for **Steam**, **TeamViewer** and **SecureCRT**.


Basic Information | Environment | Installed Software | Screenshot | Logins | FTP

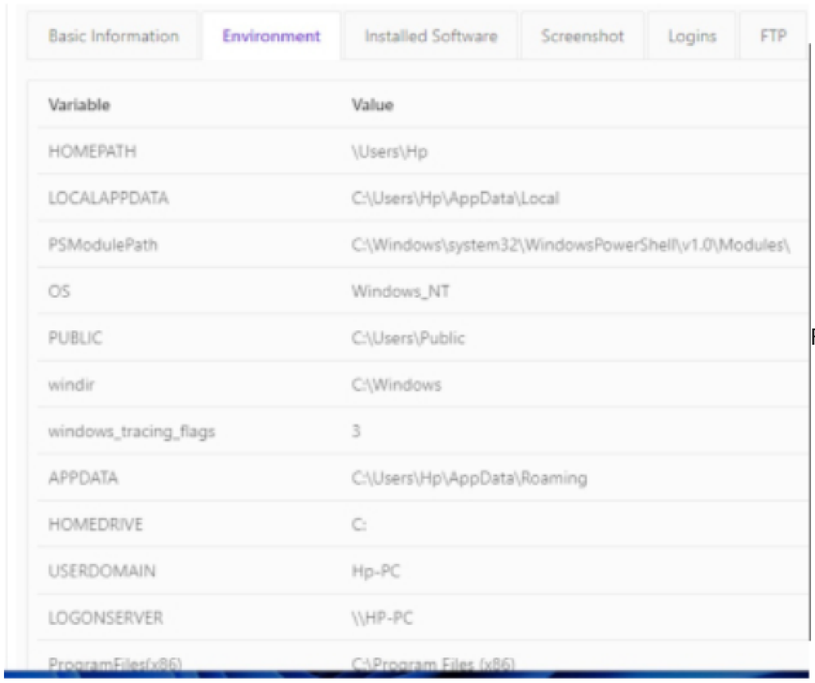
Filezilla

Hostname	Username	Password
atali.colon[redacted]	itsal[redacted]lc	Educacio[redacted]
atali.colon[redacted]	itsal[redacted]lc	Educac[redacted]

Figure 7 – Exfiltrated Filezilla FTP credentials as they appear on the attacker's end.

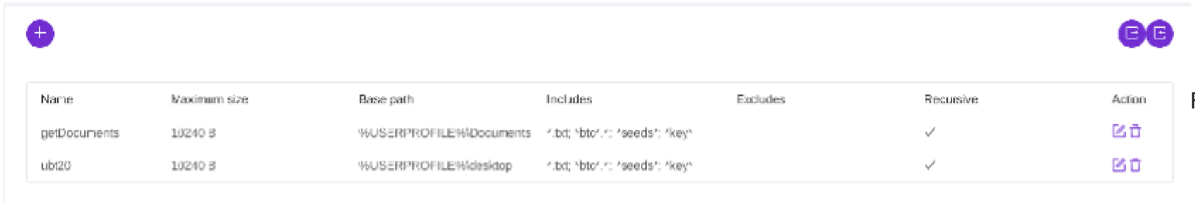
The author put a particular emphasis on features related to stealing cryptocurrency; in one version update, which featured 9 new features, 4 of these were enhancements to exfiltrating and cracking cryptocurrency wallets. The list of supported wallets in the initial release is truly unwieldy, and includes **Auvitas**, **BitApp**, **Crocobit**, **Exodus**, **Finnie**, **GuildWallet**, **ICONex**, **Jaxx**, **Keplr**, **Liquality**, **MTV**, **Metamask**, **Mobox**, **Nifty**, **Oxygen**, **Phantom**, **Rabet**, **Ronin**, **Slope**, **Sollet**, **Starcoin**, **Swash**, **Terra**, **Station**, **Tron**, **XinPay**, **Yoroi**, **ZilPay**, **Coin98**, **Armory**, **AtomicWallet**, **Atomicdex**, **Binance**, **Bisq**, **BitcoinCore**, **BitcoinGold**, **Bytecoin**, **coinomi**, **DashCore**, **DeFi**, **Dogecoin**, **Electron**, **Electrum**, **Ethereum**, **Exodus**, **Frame**, **Guarda**, **Jaxx**, **LitecoinCore**, **Monero**, **MyCrypto**, **MyMonero**, **Safepay**, **Solar**, **Tokenpocket**, **WalletWasabi**, **Zap**, **Zcash** and **Zecwallet**. We wouldn't blame the reader for wondering how many of these are actually Pokemon that we had covertly added to the list as a practical joke.

All of these stealing actions are performed automatically upon infection. If the attacker decides to get more hands-on with the infected machine, they can push a new configuration to the **"file grabbing" module**, which will exfiltrate all files matching a windows search query; or, for the true power user, push **hand-crafted powershell** to be executed on the victim machine.



Variable	Value
HOMEPATH	\Users\Hp
LOCALAPPDATA	C:\Users\Hp\AppData\Local
PSModulePath	C:\Windows\system32\WindowsPowerShell\v1.0\Modules\
OS	Windows_NT
PUBLIC	C:\Users\Public
windir	C:\Windows
windows_tracing_flags	3
APPDATA	C:\Users\Hp\AppData\Roaming
HOMEDRIVE	C:
USERDOMAIN	Hp-PC
LOGONSERVER	\\HP-PC
ProgramFiles(x86)	C:\Program Files (x86)

Figure 8 – Exfiltrated environment variables as they appear on the attacker's end.



Name	Maximum size	Base path	Includes	Excludes	Recursive	Action
getDocuments	10240 B	%USERPROFILE%\Documents	*.txt; *btc*.*; *seeds*; *key*		✓	
ubt20	10240 B	%USERPROFILE%\desktop	*.txt; *btc*.*; *seeds*; *key*		✓	

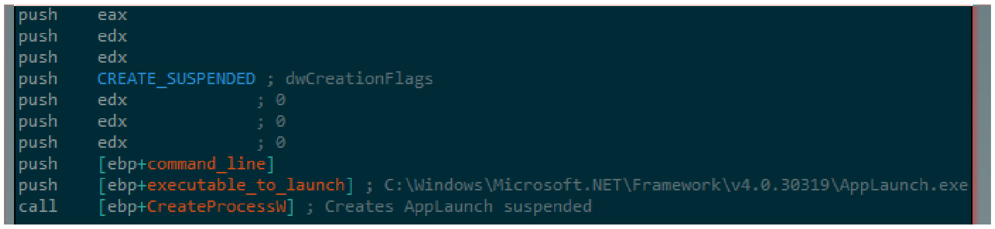
Figure 9 – Files exfiltrated by the "file grab" module as they appear on the attacker's end.

Technical Analysis Highlights

Preliminary execution flow

In this detailed and instructive write-up, Eli Salem punches with determination through each of the half-dozen execution stages (droppers, shellcodes, installers, …) that this malware goes through before it gets to the information-stealing functionality.

When analyzing Rhadamanthys, we have observed differences between the logic of the analyzed sample and the logic detailed in the above write-up, which testify to the malware's constant development and flexible build process. Most notable was the behavior of the NSIS loader DLL, which in the execution flow we analyzed, creates a suspended process from `C:\\Windows\\Microsoft.Net\\Framework\\v4.0.30319\\AppLaunch.exe` then replaces the suspended process' sections one-by-one with injected malicious code.



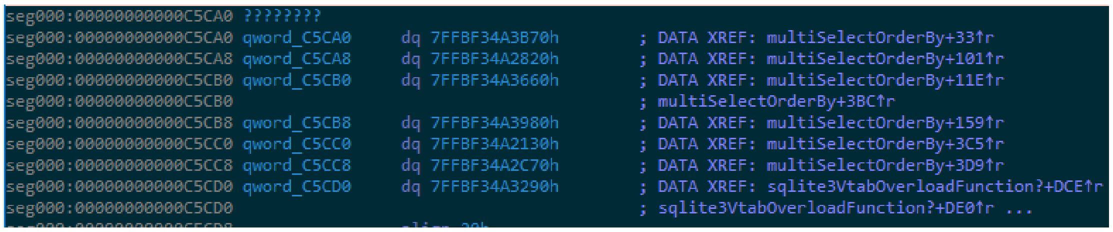
```
push    eax
push    edx
push    edx
push    CREATE_SUSPENDED ; dwCreationFlags
push    edx             ; 0
push    edx             ; 0
push    edx             ; 0
push    [ebp+command_line]
push    [ebp+executable_to_launch] ; C:\Windows\Microsoft.NET\Framework\v4.0.30319\AppLaunch.exe
call    [ebp+CreateProcessW] ; Creates AppLaunch suspended
```

As described in the above-mentioned write-up, the injected code then proceeds to load several execution stages in sequence, one of which attempts many VM evasions taken from the Al-Khaser project and then unhooks functions in `ntdll.dll` in an attempt to avoid detection. Finally, it resolves an internally obfuscated C2 address and, from there, downloads the final stage containing the actual information-stealing functionality.

Analyzing an Orphaned Memory Dump

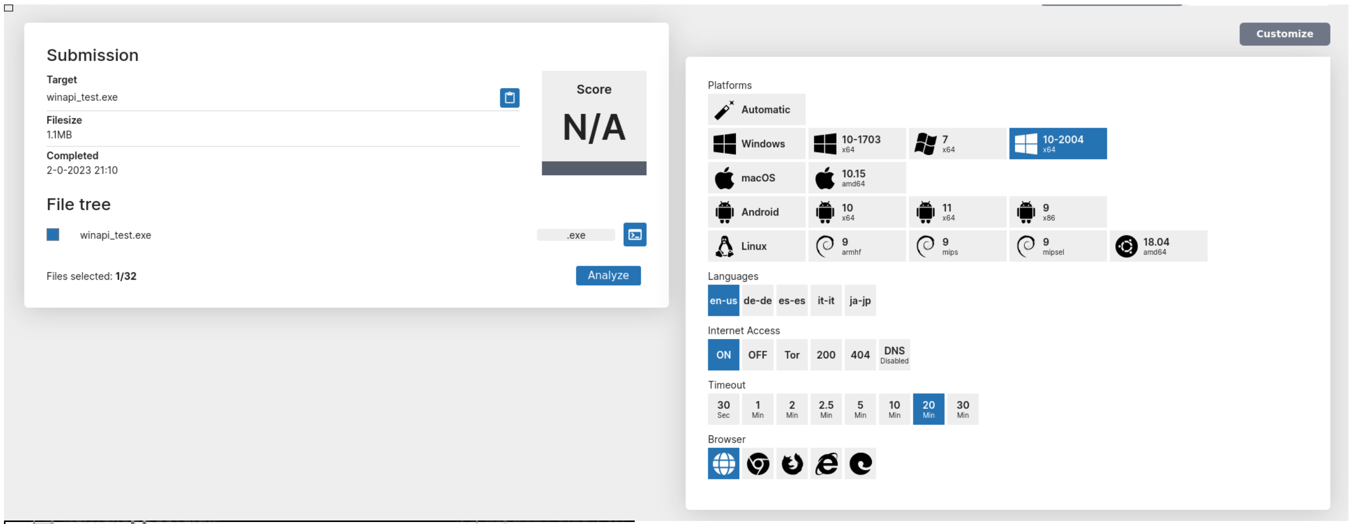
Analyzing the actual stealing logic is not so straightforward. Without access to a live C2 server, at this point an analyst has two options. Either they go chasing a brand new execution chain, doing the hard work debugging all the stages and hoping to catch a live C2 server which won't filter them out using god knows how many heuristics; or else working with a dump in an unreadable state, obtained from a sandbox run that happened when the C2 was still live. In this particular case the memory dump contains many telling strings that telegraph what the malware does in broad strokes, but there are many obstacles before proper interactive disassembly can take place.

The first and most major obstacle is the lack of resolution for API calls. Opening the dump in a disassembler and following the function calls, one very quickly runs across what must be a homebrew import table of dynamically resolved functions. The dump is an artifact of a sandbox run that has long since concluded and these addresses seem to be meaningless now. We were able to resolve nearly every function using a method that will be explained below.



```
seg000:0000000000C5CA0 ????????
seg000:0000000000C5CA0 qword_C5CA0     dq 7FFBF34A3B70h        ; DATA XREF: multiSelectOrderBy+33↑r
seg000:0000000000C5CA8 qword_C5CA8     dq 7FFBF34A2820h        ; DATA XREF: multiSelectOrderBy+101↑r
seg000:0000000000C5CB0 qword_C5CB0     dq 7FFBF34A3660h        ; DATA XREF: multiSelectOrderBy+11E↑r
seg000:0000000000C5CB0                                         ; multiSelectOrderBy+3BC↑r
seg000:0000000000C5CB8 qword_C5CB8     dq 7FFBF34A3980h        ; DATA XREF: multiSelectOrderBy+159↑r
seg000:0000000000C5CC0 qword_C5CC0     dq 7FFBF34A2130h        ; DATA XREF: multiSelectOrderBy+3C5↑r
seg000:0000000000C5CC8 qword_C5CC8     dq 7FFBF34A2C70h        ; DATA XREF: multiSelectOrderBy+3D9↑r
seg000:0000000000C5CD0 qword_C5CD0     dq 7FFBF34A3290h        ; DATA XREF: sqlite3VtabOverloadFunction?+DCE↑r
seg000:0000000000C5CD0                                         ; sqlite3VtabOverloadFunction?+DE0↑r ...
```

First, we know that these addresses, during the sandbox run, pointed to DLLs that were loaded into memory. Second, we know in what environment the execution took place: a tria.ge environment with the code name `Win10v2004-20220812-en`. We upload our own dummy executable to the sandbox, make sure that we choose the same environment used in the original sandbox run, then look at a DLL of our choice and recover the DLL version.



Submission

Target
winapi_test.exe

Filesize
1.1MB

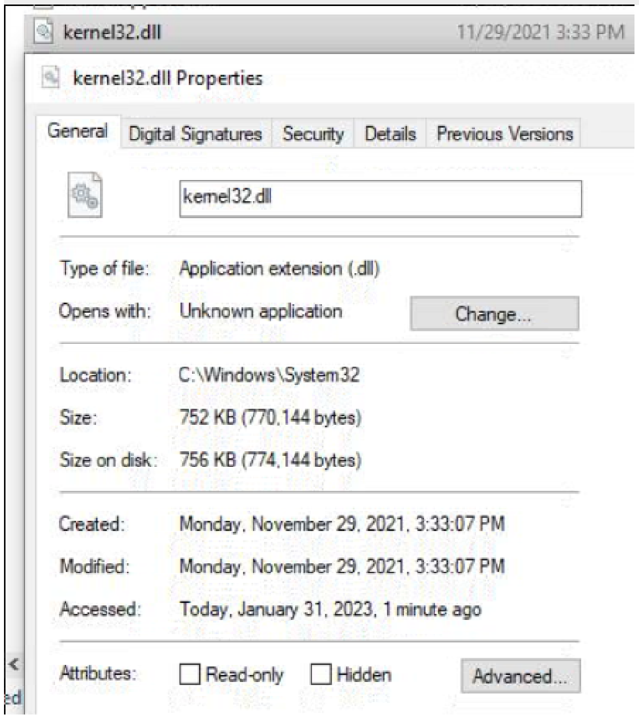
Completed
2-0-2023 21:10

Score
N/A

File tree
winapi_test.exe .exe

Files selected: 1/32

Platforms: 10-2004 x64
Languages: en-us
Internet Access: ON
Timeout: 20 Min



kernel32.dll 11/29/2021 3:33 PM

kernel32.dll Properties

General | Digital Signatures | Security | Details | Previous Versions

kernel32.dll

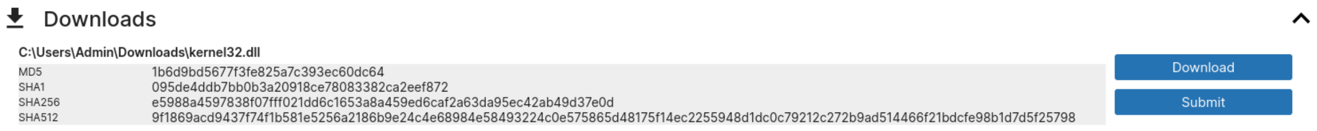
Type of file: Application extension (.dll)
Opens with: Unknown application

Location: C:\Windows\System32
Size: 752 KB (770,144 bytes)
Size on disk: 756 KB (774,144 bytes)

Created: Monday, November 29, 2021, 3:33:07 PM
Modified: Monday, November 29, 2021, 3:33:07 PM
Accessed: Today, January 31, 2023, 1 minute ago

Attributes: ☐ Read-only ☐ Hidden

Unfortunately, even if we do have the DLL version, Microsoft is not so generous with offering historical versions of DLLs for download. There are various workarounds for this sort of issue (e.g. you might want to consult winbindex). We opted to use an esoteric feature of tria.ge sandbox: many users had asked for functionality to manually dump files generated during an execution flow. As a workaround, the sandbox introduced a feature allowing users to dump any file they wish, as long as they open windows File Explorer and delete the file there manually. Well, if we try deleting `kernel32.dll` from its residing place in `C:\windows\system32` the OS won't allow it (and justifiably so), but nothing prevents us from *copying* the file somewhere else, then deleting the copy. The same DLL loaded into memory during the original sandbox run is now available from the "downloads" section of the analysis report once the analysis is terminated.

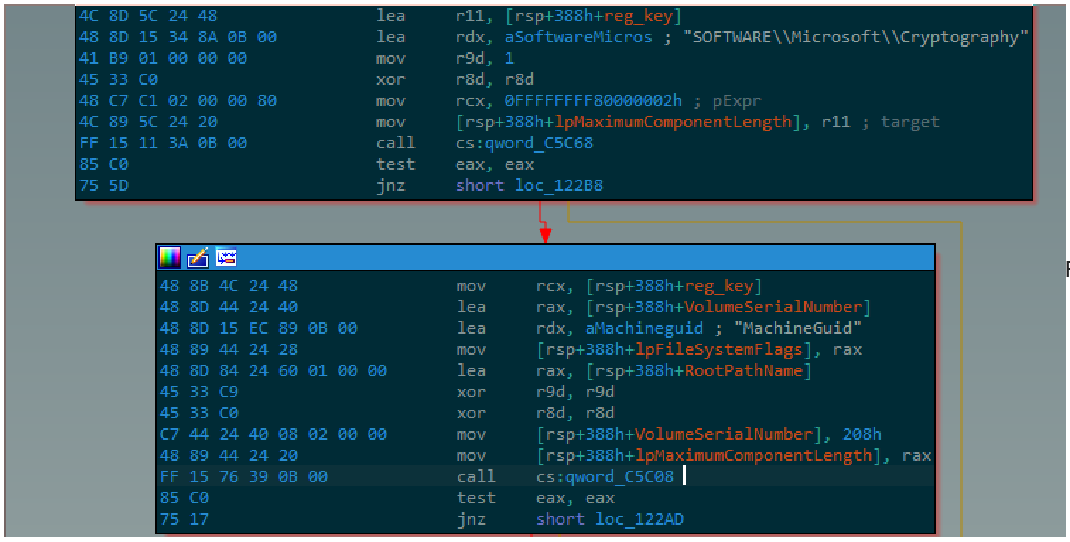

Downloads

C:\Users\Admin\Downloads\kernel32.dll

MD5	1b6d9bd5677f3fe825a7c393ec60dc64
SHA1	095de4ddb7bb0b3a20918ce78083382ca2eef872
SHA256	e5988a4597838f07fff021dd6c1653a8a459ed6caf2a63da95ec42ab49d37e0d
SHA512	9f1869acd9437f74f1b581e5256a2186b9e24c4e68984e58493224c0e575865d48175f14ec2255948d1dc0c79212c272b9ad514466f21bdcfe98b1d7d5f25798

In this way we download many DLLs that are the "usual suspects" that malware, or any software really, would want to resolve APIs from – such as `advapi32.dll`, `user32.dll`, `msvcrt.dll`, `ws2_32.dll` and so on. Now we can open each of these in a disassembler, which manually loads the file and assigns virtual addresses to each of the DLL functions. Sadly we are far from done because we still do not know the base address of the DLL when it was loaded originally, or even what particular DLL contains the function that some memory address refers to.

Not even knowing which is the relevant DLL can be mitigated to some degree by plain observation – for example, in the below image, the function `qword_c5c08` (pointer value `0x7ffbf1bd5f20`) is taking a registry key as an argument, and so is highly likely to have come from `advapi32.dll`. But this won't work for every DLL – we won't always be lucky enough to find a function that the malware feeds such an incriminating hardcoded string as a parameter. More crucially, even if we somehow knew the correct DLL for every function address, this still wouldn't tell us the original address the DLL was loaded at during the original sandbox run, necessary to calculate the rebase delta between the function addresses that were loaded then into memory (which we are trying to resolve) and the labeled function addresses in the loaded, annotated DLL that we have open in the disassembler.


```
4C 8D 5C 24 48                  lea     r11, [rsp+388h+reg_key]
48 8D 15 34 8A 0B 00            lea     rdx, aSoftwareMicros ; "SOFTWARE\\Microsoft\\Cryptography"
41 B9 01 00 00 00               mov     r9d, 1
45 33 C0                        xor     r8d, r8d
48 C7 C1 02 00 00 80            mov     rcx, 0FFFFFFFF80000002h ; pExpr
4C 89 5C 24 20                  mov     [rsp+388h+lpMaximumComponentLength], r11 ; target
FF 15 11 3A 0B 00               call    cs:qword_C5C68
85 C0                           test    eax, eax
75 5D                           jnz     short loc_122B8
```

```
48 8B 4C 24 48                  mov     rcx, [rsp+388h+reg_key]
48 8D 44 24 40                  lea     rax, [rsp+388h+VolumeSerialNumber]
48 8D 15 EC 89 0B 00            lea     rdx, aMachineguid ; "MachineGuid"
48 89 44 24 28                  mov     [rsp+388h+lpFileSystemFlags], rax
48 8D 84 24 60 01 00 00         lea     rax, [rsp+388h+RootPathName]
45 33 C9                        xor     r9d, r9d
45 33 C0                        xor     r8d, r8d
C7 44 24 40 08 02 00 00         mov     [rsp+388h+VolumeSerialNumber], 208h
48 89 44 24 20                  mov     [rsp+388h+lpMaximumComponentLength], rax
FF 15 76 39 0B 00               call    cs:qword_C5C08
85 C0                           test    eax, eax
75 17                           jnz     short loc_122AD
```

Figure 15 – qword_c5c08 is probably a function that interacts with the Windows registry in some way.

To get past this hurdle we note that the function addresses in the sandbox dump are probably divided into contiguous sequences that were each imported from the same DLL. This means if we take 10 qword pointers from the table and get lucky enough that they were all resolved from the same DLL, then in that DLL when loaded into memory, these 10 functions will exist with the same *differences* between their addresses. To show the key insight here we will use a toy example: suppose our list of 10 addresses to resolve begins with some address *AX* then proceeds with *AX+0x300*, *AX+0x500*, *AX+0x930*, and so on six other addresses; suppose further that in one of the loaded and annotated DLLs we find that for some address *AY* it happens that *AY+0x300*, *AY+0x500*, *AY+0x930* and so on and so on are all addresses of functions. This is a very lucky coincidence to have happened on its own, and in all probability, the original address *AX* in the original sandbox run resolved to the function that is in *AY* in our annotated file. It is possible to further sanity-check the match by looking at the 10 function names that match the addresses on the list and verifying that they seem like a reasonable list to have been required by the software that had been run in the sandbox.

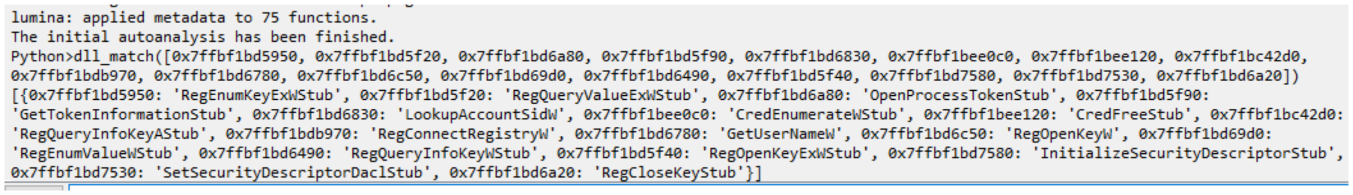
The following IDAPython code, when run in a loaded DLL database, will automate the task of finding matches of function address sequences:

```
exports = list(Functions(0x0000000000000000,0xFFFFFFFFFFFFFFFF))

def dll_match(imports):
    result = []
    import_anchor = imports[0]
    for anchor in exports:
        if all([anchor+(_import-import_anchor) in exports for _import in imports]):
            result.append({_import:get_func_name(anchor+(_import-import_anchor)) for _import in imports})
    return result
```

For example, the address seen in the above image (the one we suspect to have been resolved from `advapi32.dll`) appears in the following sequence of 10 addresses:

```
[0x7ffbf1bd5950, 0x7ffbf1bd5f20, 0x7ffbf1bd6a80, 0x7ffbf1bd5f90, 0x7ffbf1bd6830, 0x7ffbf1bee0c0, 0x7ffbf1bee120,
0x7ffbf1bc42d0, 0x7ffbf1bdb970, 0x7ffbf1bd6780, 0x7ffbf1bd6c50, 0x7ffbf1bd69d0, 0x7ffbf1bd6490, 0x7ffbf1bd5f40,
0x7ffbf1bd7580, 0x7ffbf1bd7530, 0x7ffbf1bd6a20]
```

We open an annotated idb of the `advapi32.dll` file we dumped from the sandbox, load the above IDA script and run the function `dll_match` with this list of addresses as input. As output we receive the correct resolution for each of these function addresses.



```
lumina: applied metadata to 75 functions.
The initial autoanalysis has been finished.
Python>dll_match([0x7ffbf1bd5950, 0x7ffbf1bd5f20, 0x7ffbf1bd6a80, 0x7ffbf1bd5f90, 0x7ffbf1bd6830, 0x7ffbf1bee0c0, 0x7ffbf1bee120, 0x7ffbf1bc42d0,
0x7ffbf1bdb970, 0x7ffbf1bd6780, 0x7ffbf1bd6c50, 0x7ffbf1bd69d0, 0x7ffbf1bd6490, 0x7ffbf1bd5f40, 0x7ffbf1bd7580, 0x7ffbf1bd7530, 0x7ffbf1bd6a20])
[{0x7ffbf1bd5950: 'RegEnumKeyExWStub', 0x7ffbf1bd5f20: 'RegQueryValueExWStub', 0x7ffbf1bd6a80: 'OpenProcessTokenStub', 0x7ffbf1bd5f90:
'GetTokenInformationStub', 0x7ffbf1bd6830: 'LookupAccountSidW', 0x7ffbf1bee0c0: 'CredEnumerateWStub', 0x7ffbf1bee120: 'CredFreeStub', 0x7ffbf1bc42d0:
'RegQueryInfoKeyAStub', 0x7ffbf1bdb970: 'RegConnectRegistryW', 0x7ffbf1bd6780: 'GetUserNameW', 0x7ffbf1bd6c50: 'RegOpenKeyW', 0x7ffbf1bd69d0:
'RegEnumValueWStub', 0x7ffbf1bd6490: 'RegQueryInfoKeyWStub', 0x7ffbf1bd5f40: 'RegOpenKeyExWStub', 0x7ffbf1bd7580: 'InitializeSecurityDescriptorStub',
0x7ffbf1bd7530: 'SetSecurityDescriptorDaclStub', 0x7ffbf1bd6a20: 'RegCloseKeyStub'}]
```

It turns out that the above-mentioned function that had been loaded to the address `0x7ffbf1bd5f20` during the sandbox run is `RegQueryValueExW`. Using this method, it is easy to go “shopping” and try running the same script against various DLLs to see what matches are obtained, and how feasible they are. While that specific workflow doesn’t scale very well, it’s not too difficult to see how the process can be streamlined, if need be (for instance, by keeping a precomputed database of function address differences of many DLL versions, and making all the difference comparisons against it).

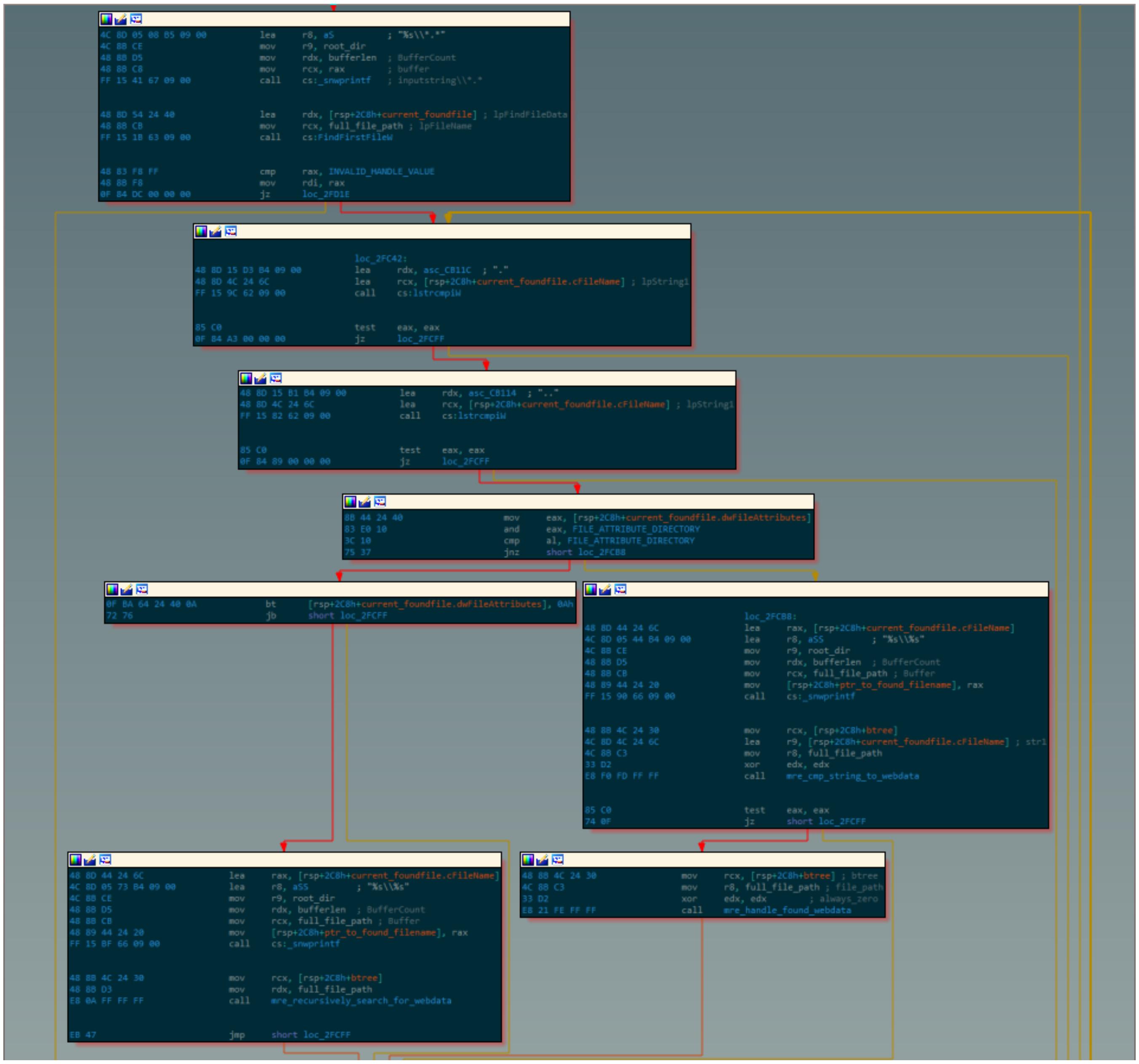
Interactive Disassembly of a Sample Functionality: Stealing Chrome Information

Even with the API calls resolved, the database is still very large and spans over 2500 functions. Many of these are library functions from 3rd party libraries such as sqlite3 and lua_cjson; this introduces a further hassle in that resolving these functions requires us to compile our own annotated version of these libraries and then perform a bindiff (or somesuch) to label the functions used by Rhadamanthys. This is an infamously finicky process, and many of the labels are not of very much use before being verified manually.

With all that said, the state of the database is more palatable now and allows us to analyze the execution flow in a way we couldn’t before. As an example we will focus on the malware’s capability of stealing stored login credentials, cookies and so on from Google Chrome, which includes three stages:

- Searching for the correct directory containing all the data
- Reading raw data out of files of interest containing cookies, login data, and so on
- Based on whether the data is in JSON or SQL database format, using 3rd-party library logic to parse the data

The malware first performs a recursive search of the victim filesystem for a file named “web data” in order to navigate to `%LOCALAPPDATA%\\Google\\Chrome\\User Data\\default`, then traverses the tree to look for other artifacts such as “Cookies” or “Login Data” and collects each match into a binary bitfield; if this bitfield is sufficiently nonzero, the malware is then satisfied that it has located the Chrome directory correctly.



```
4C 8D 05 08 85 09 00          lea     r8, aS          ; "%s\\*.*"
4C 88 CE                      mov     r9, root_dir
48 88 D5                      mov     rdx, bufferlen  ; BufferCount
48 88 C8                      mov     rcx, rax        ; buffer
FF 15 41 67 09 00             call    cs:_snwprintf   ; inputstring\\*.*


48 8D 54 24 40                lea     rdx, [rsp+2C8h+current_foundfile] ; lpFindFileData
48 88 CB                      mov     rcx, full_file_path ; lpFileName
FF 15 1B 63 09 00             call    cs:FindFirstFileW


48 83 F8 FF                   cmp     rax, INVALID_HANDLE_VALUE
48 88 F8                      mov     rdi, rax
0F 84 DC 00 00 00             jz      loc_2FD1E
```

```

                              loc_2FC42:
48 8D 15 D3 B4 09 00          lea     rdx, asc_CB11C  ; "."
48 8D 4C 24 6C                lea     rcx, [rsp+2C8h+current_foundfile.cFileName] ; lpString1
FF 15 9C 62 09 00             call    cs:lstrcmpiW


85 C0                         test    eax, eax
0F 84 A3 00 00 00             jz      loc_2FCFF
```

```
48 8D 15 B1 B4 09 00          lea     rdx, asc_CB114  ; ".."
48 8D 4C 24 6C                lea     rcx, [rsp+2C8h+current_foundfile.cFileName] ; lpString1
FF 15 82 62 09 00             call    cs:lstrcmpiW


85 C0                         test    eax, eax
0F 84 89 00 00 00             jz      loc_2FCFF
```

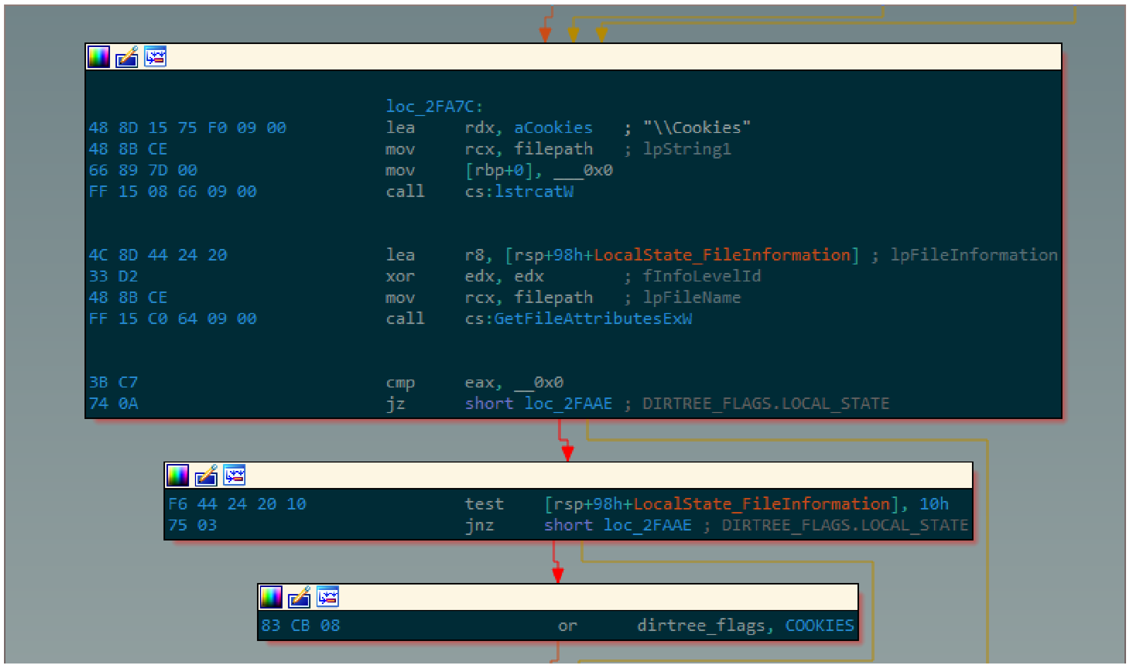
```
8B 44 24 40                   mov     eax, [rsp+2C8h+current_foundfile.dwFileAttributes]
83 E0 10                      and     eax, FILE_ATTRIBUTE_DIRECTORY
3C 10                         cmp     al, FILE_ATTRIBUTE_DIRECTORY
75 37                         jnz     short loc_2FCB8
```

```
0F BA 64 24 40 0A             bt      [rsp+2C8h+current_foundfile.dwFileAttributes], 0Ah
72 76                         jb      short loc_2FCFF
```

```

                              loc_2FCB8:
48 8D 44 24 6C                lea     rax, [rsp+2C8h+current_foundfile.cFileName]
4C 8D 05 44 B4 09 00          lea     r8, aSS         ; "%s\\%s"
4C 88 CE                      mov     r9, root_dir
48 88 D5                      mov     rdx, bufferlen  ; BufferCount
48 88 CB                      mov     rcx, full_file_path ; Buffer
48 89 44 24 20                mov     [rsp+2C8h+ptr_to_found_filename], rax
FF 15 90 66 09 00             call    cs:_snwprintf


48 88 4C 24 30                mov     rcx, [rsp+2C8h+btree]
4C 8D 4C 24 6C                lea     r9, [rsp+2C8h+current_foundfile.cFileName] ; str1
4C 88 C3                      mov     r8, full_file_path
33 D2                         xor     edx, edx
E8 F0 FD FF FF                call    mre_cmp_string_to_webdata


85 C0                         test    eax, eax
74 0F                         jz      short loc_2FCFF
```

```
48 8D 44 24 6C                lea     rax, [rsp+2C8h+current_foundfile.cFileName]
4C 8D 05 73 B4 09 00          lea     r8, aSS         ; "%s\\%s"
4C 88 CE                      mov     r9, root_dir
48 88 D5                      mov     rdx, bufferlen  ; BufferCount
48 88 CB                      mov     rcx, full_file_path ; Buffer
48 89 44 24 20                mov     [rsp+2C8h+ptr_to_found_filename], rax
FF 15 BF 66 09 00             call    cs:_snwprintf


48 88 4C 24 30                mov     rcx, [rsp+2C8h+btree]
48 88 D3                      mov     rdx, full_file_path
E8 0A FF FF FF                call    mre_recursively_search_for_webdata


EB 47                         jmp     short loc_2FCFF
```

```
48 88 4C 24 30                mov     rcx, [rsp+2C8h+btree] ; btree
4C 88 C3                      mov     r8, full_file_path ; file_path
33 D2                         xor     edx, edx        ; always_zero
E8 21 FE FF FF                call    mre_handle_found_webdata
```

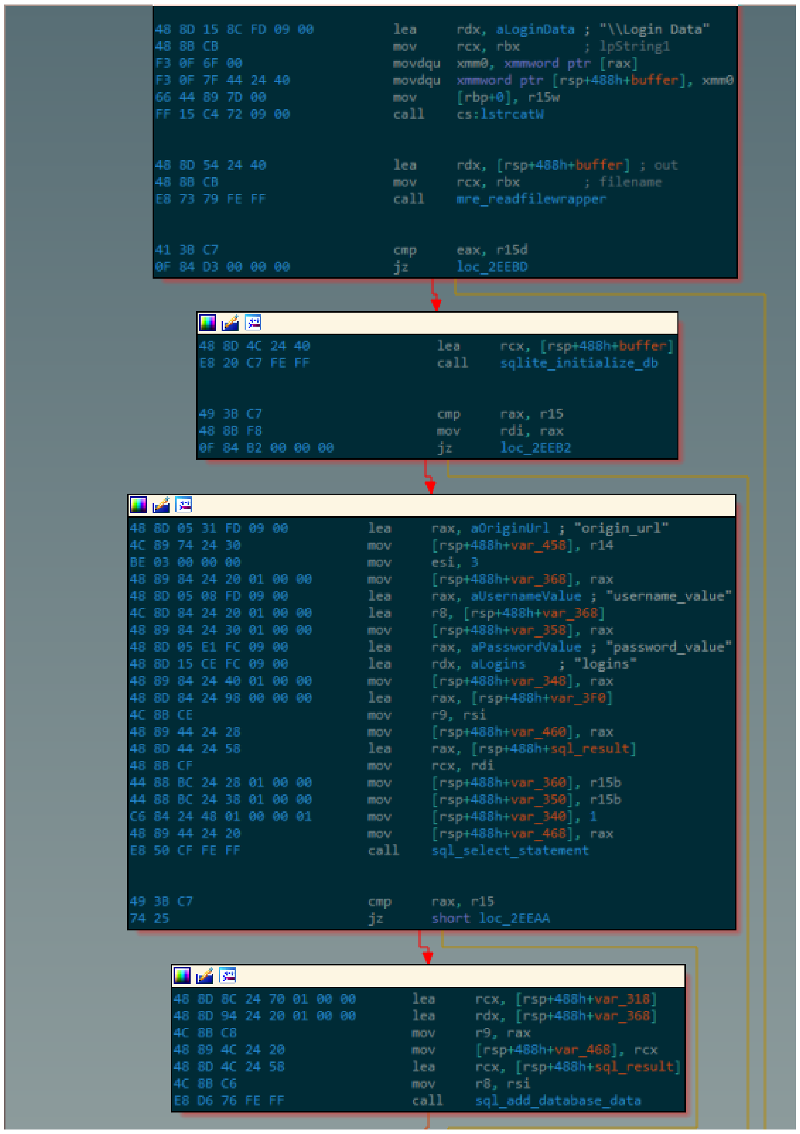


```
                              loc_2FA7C:
48 8D 15 75 F0 09 00          lea     rdx, aCookies   ; "\\Cookies"
48 8B CE                      mov     rcx, filepath   ; lpString1
66 89 7D 00                   mov     [rbp+0], ___0x0
FF 15 08 66 09 00             call    cs:lstrcatW

4C 8D 44 24 20                lea     r8, [rsp+98h+LocalState_FileInformation] ; lpFileInformation
33 D2                         xor     edx, edx        ; fInfoLevelId
48 8B CE                      mov     rcx, filepath   ; lpFileName
FF 15 C0 64 09 00             call    cs:GetFileAttributesExW

3B C7                         cmp     eax, __0x0
74 0A                         jz      short loc_2FAAE ; DIRTREE_FLAGS.LOCAL_STATE
```

```
F6 44 24 20 10                test    [rsp+98h+LocalState_FileInformation], 10h
75 03                         jnz     short loc_2FAAE ; DIRTREE_FLAGS.LOCAL_STATE
```

```
83 CB 08                      or      dirtree_flags, COOKIES
```

The malware then accesses files of interest, such as `login data`. Some of these files are SQL databases, in which case the malware initializes a SQL database from the file contents in-memory, then obtains the desired data by issuing a SELECT statement. In contrast, others are in JSON format, and so the malware instead calls a function to parse the JSON and extract the value associated with a certain key:

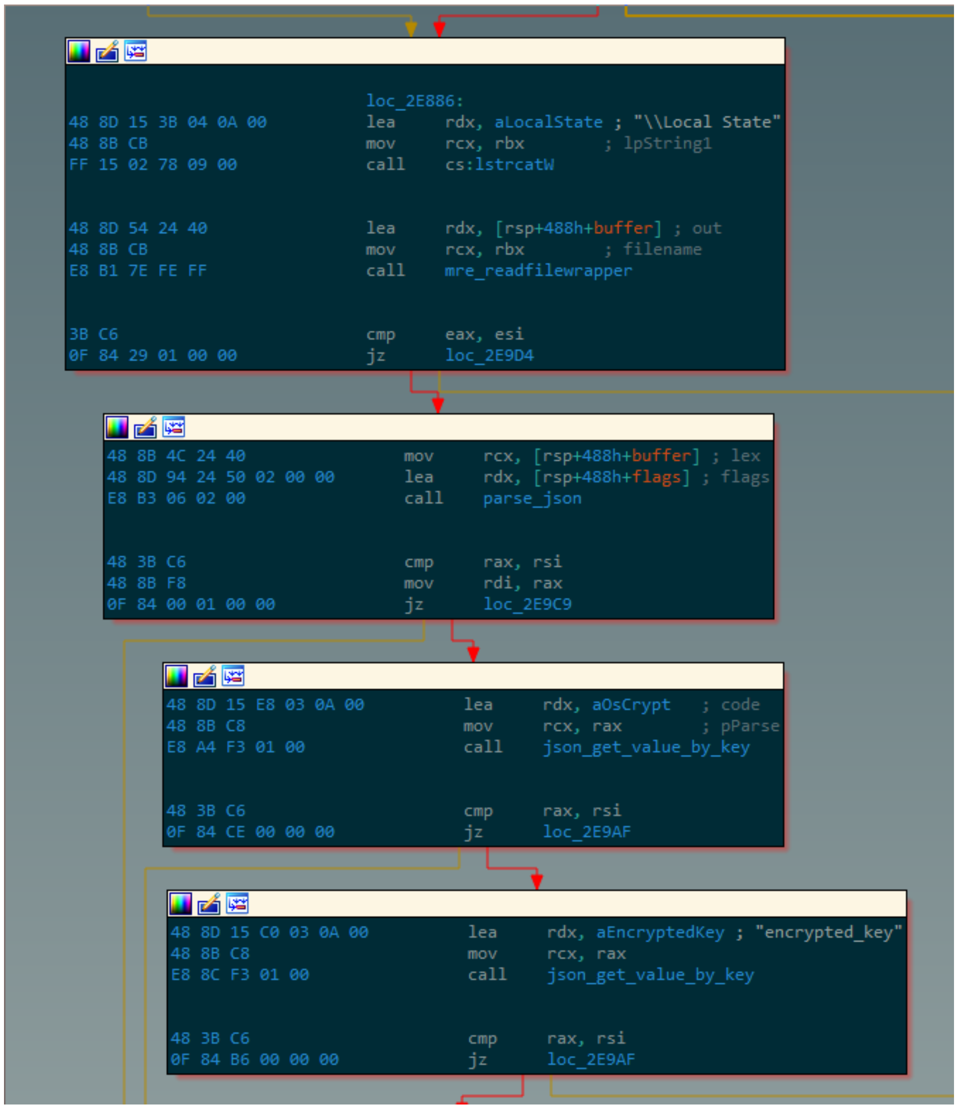


```
48 8D 15 8C FD 09 00          lea     rdx, aLoginData ; "\\Login Data"
48 8B CB                      mov     rcx, rbx        ; lpString1
F3 0F 6F 00                   movdqu  xmm0, xmmword ptr [rax]
F3 0F 7F 44 24 40             movdqu  xmmword ptr [rsp+488h+buffer], xmm0
66 44 89 7D 00                mov     [rbp+0], r15w
FF 15 C4 72 09 00             call    cs:lstrcatW


48 8D 54 24 40                lea     rdx, [rsp+488h+buffer] ; out
48 8B CB                      mov     rcx, rbx        ; filename
E8 73 79 FE FF                call    mre_readfilewrapper


41 3B C7                      cmp     eax, r15d
0F 84 D3 00 00 00             jz      loc_2EEBD
```

```
48 8D 4C 24 40                lea     rcx, [rsp+488h+buffer]
E8 20 C7 FE FF                call    sqlite_initialize_db


49 3B C7                      cmp     rax, r15
48 8B F8                      mov     rdi, rax
0F 84 B2 00 00 00             jz      loc_2EEB2
```

```
48 8D 05 31 FD 09 00          lea     rax, aOriginUrl ; "origin_url"
4C 89 74 24 30                mov     [rsp+488h+var_458], r14
BE 03 00 00 00                mov     esi, 3
48 89 84 24 20 01 00 00       mov     [rsp+488h+var_368], rax
48 8D 05 08 FD 09 00          lea     rax, aUsernameValue ; "username_value"
4C 8D 84 24 20 01 00 00       lea     r8, [rsp+488h+var_368]
48 89 84 24 30 01 00 00       mov     [rsp+488h+var_358], rax
48 8D 05 E1 FC 09 00          lea     rax, aPasswordValue ; "password_value"
48 8D 15 CE FC 09 00          lea     rdx, aLogins    ; "logins"
48 89 84 24 40 01 00 00       mov     [rsp+488h+var_348], rax
48 8D 84 24 98 00 00 00       lea     rax, [rsp+488h+var_3F0]
4C 8B CE                      mov     r9, rsi
48 89 44 24 28                mov     [rsp+488h+var_460], rax
48 8D 44 24 58                lea     rax, [rsp+488h+sql_result]
48 8B CF                      mov     rcx, rdi
44 88 BC 24 28 01 00 00       mov     [rsp+488h+var_360], r15b
44 88 BC 24 38 01 00 00       mov     [rsp+488h+var_350], r15b
C6 84 24 48 01 00 00 01       mov     [rsp+488h+var_340], 1
48 89 44 24 20                mov     [rsp+488h+var_468], rax
E8 50 CF FE FF                call    sql_select_statement


49 3B C7                      cmp     rax, r15
74 25                         jz      short loc_2EEAA
```

```
48 8D 8C 24 70 01 00 00       lea     rcx, [rsp+488h+var_318]
48 8D 94 24 20 01 00 00       lea     rdx, [rsp+488h+var_368]
4C 8B C8                      mov     r9, rax
48 89 4C 24 20                mov     [rsp+488h+var_468], rcx
48 8D 4C 24 58                lea     rcx, [rsp+488h+sql_result]
4C 8B C6                      mov     r8, rsi
E8 D6 76 FE FF                call    sql_add_database_data
```



```
                                loc_2E886:
48 8D 15 3B 04 0A 00            lea     rdx, aLocalState ; "\\Local State"
48 8B CB                        mov     rcx, rbx        ; lpString1
FF 15 02 78 09 00               call    cs:lstrcatW

48 8D 54 24 40                  lea     rdx, [rsp+488h+buffer] ; out
48 8B CB                        mov     rcx, rbx        ; filename
E8 B1 7E FE FF                  call    mre_readfilewrapper

3B C6                           cmp     eax, esi
0F 84 29 01 00 00               jz      loc_2E9D4
```

```
48 8B 4C 24 40                  mov     rcx, [rsp+488h+buffer] ; lex
48 8D 94 24 50 02 00 00         lea     rdx, [rsp+488h+flags] ; flags
E8 B3 06 02 00                  call    parse_json

48 3B C6                        cmp     rax, rsi
48 8B F8                        mov     rdi, rax
0F 84 00 01 00 00               jz      loc_2E9C9
```

```
48 8D 15 E8 03 0A 00            lea     rdx, aOsCrypt   ; code
48 8B C8                        mov     rcx, rax        ; pParse
E8 A4 F3 01 00                  call    json_get_value_by_key

48 3B C6                        cmp     rax, rsi
0F 84 CE 00 00 00               jz      loc_2E9AF
```

```
48 8D 15 C0 03 0A 00            lea     rdx, aEncryptedKey ; "encrypted_key"
48 8B C8                        mov     rcx, rax
E8 8C F3 01 00                  call    json_get_value_by_key

48 3B C6                        cmp     rax, rsi
0F 84 B6 00 00 00               jz      loc_2E9AF
```

With the information parsed into plaintext format, it can now be appended to the database of stolen information that is eventually reported back to the attacker, and the stealing functionality for this specific target (Chrome) is concluded. Most of Rhadamanthys' code base is composed of an entirely too large amount of variations on this same idea, each targeting a different piece of data as laid out in the earlier description of the malware's feature set.

Conclusion

Rhadamanthys represents a step in the emerging tradition of malware that tries to do as much as possible, and also a demonstration that in the malware business, having a strong brand is everything. Some readers may recall the odd tale of Godzilla loader, which tried to undercut Emotet by retailing for a quarter of the price and boasted a set of features so different from its competitor that a proper comparison between the two was impossible. This was a stark demonstration that cybercriminals don't make explicit calculations of which feature-sets will net them a higher amount of victims – they rely on a fuzzy feeling of how well they trust the developer, the brand and the sound of the feature list; and, failing that, on trial and error. Any developer can write a piece of malware, and some developers can even write a decent piece of malware with useful features, but it takes a cunning mind in tune with the market to come out swinging like the author of this malware did, shouting "I have all the features, I have the best features" and fostering a successful relationship with a base of early adopters.

Should we be worried? It is tempting to quietly classify malware like Rhadamanthys in the "nuisance" drawer. An uninvited credit card charge of $2,000 doesn't seem like much compared to ransomware and the existential threat it poses to entire organizations. It's easy to forget the process by which the $2,000 charge happens – the malware doesn't just steal your credit card details, it steals *everything*. It is a relief that employees in government agencies and energy companies who get hit with this kind of malware are typically treated the same by attackers as any other victim, but we would do well to remember that this is not a law of nature.

We would also do well to dwell on the technical demands such malware poses to us on the defensive side. It used to be a quaint sport to sidestep sandboxes and frustrate analysts who are trying to properly disassemble, debug, analyze the API call log of some malware. For how long can the analysis be delayed? An hour? A day? God forbid, a week? But nowadays, there seems to be a sinister shift where malware authors have gotten more ambitious and are saying, let’s see how close I can get to the whole analysis just not happening. While we are still quite far off from that scary scenario, malicious instruments that inch us closer there are being thought of all the time, and the unexplored and under-explored space of such instruments is well and truly frightening. As an industry, it is paramount that we create new tools and protocols to meet these instruments with equal and opposite force when they arrive.

GO UP
BACK TO ALL POSTS