Malware, when to extract configuration data from system memory

Jeroen Kraan
Hogeschool van Amsterdam
Email: Jeroen.Kraan@hva.nl

Ricardo van Zutphen
Hogeschool van Amsterdam
Email: Ricardo.van.Zutphen@hva.nl

I. Introduction

Dynamic malware analysis is an important tool to recognize and help understand new threats in the form of malware. Dynamic malware analysis is studying the behaviour of malware while it is being executed on a controlled host computer. This can be in a virtualized environment or on a physical machine. The purpose of dynamic analysis is to collect behavioural data like arguments used in system API calls, the contents of the system memory, and network communications. Network communication is an important way to recognize who the malware is communicating with. The who is called a command and control server (C2). C2s are part of what is called a malware configuration. This can contain C2s, version numbers, and any other information used by a specific type of malware.

One way to collect parts of the configuration data is to look at the network communication of malware. Another type is trying to retrieve the configurations or part of it from process memory dumps.

Automated Dynamic malware analysis systems like Cuckoo Sandbox try to automate this process by making memory dumps at a point during the execution of the malware. These memory dumps can then be analysed by various tools. The question is: at what point in execution does the system memory contain the malware configuration?

Cuckoo Sandbox is planning to develop a new module to increase the chance of extracting a configuration from the memory. To support this development, this research will be focused on finding the most likely time slot and possibly the required events during execution in which the malware configuration will be located in the system memory. Our hypothesis: Creating process memory dumps during the malware execution increases the chance of finding configuration data compared to creating dumps at the end of execution, will help determine when to create process memory dumps during the malware execution.

This research will focus on ransom and banking malware. By ransomware we mean malware that has a focus on extorting a victim by holding hostage of their computer, by encrypting their personal files and asking for a sum of money in order to restore their files. Examples of names of ransomware are: Cryptolocker [1], TeslaCrypt [2], and Locky [3]. By banking malware, we mean malware that has a focus on stealing financial information, stealing login credentials, keystroke logging, and form manipulation. Examples of names of banking malware are: Dridex [4], Zeus [5], and Vawtrak [6]. The reason of choosing these types of malware is that the binaries for this malware are likely to contain configuration data like C2s, because the malware will need to upload the collected data.

This paper will be organized as follows: section 2 will contain a statement of the problem. Section 3 will contain an overview of the collected dataset used for measurement. Section 4 explains how to recognise the in-memory malware configuration for the collected dataset. Section 5 will contain the result of the measurement. Section 6 will contain the conclusion.

II. Problem statement

Malware usually communicates with command and control (C2) servers. We call the IP addresses, ports, encryption keys, URLs, other information used to communicate, and other information used by the malware: configuration data. This data can be embedded in the malware binary, where it is usually encrypted or obfuscated in some way. If malware wants to use the configuration data while being executed on a system, it will need to decrypt the information and load it into the system memory [7].

Dynamic malware analysis systems like Cuckoo Sandbox [8] can try to take advantage of this. A process memory dump can be made, so that the configuration information can be retrieved from the dump. Cuckoo Sandbox makes these dumps at the end of an analysis. The timing of making this dump is crucial. If the dump is made after a process containing the information has exited, meaning the process has ended and no longer resides in the memory, the information will not be in the dump. This causes Cuckoo analyses to sometimes miss configuration data from malware processes that have already exited.

The problem is that the exact moment in time where the information resides in the memory is not usually known. A possible solution for this is using interval-based memory dumping. This approach creates memory dumps at a set interval. The fewer seconds between each interval, the larger the chance of creating a dump containing the desired information. This comes with the drawback of having to search a large amount of memory dumps and possibly just missing the right moment to dump [9].

This research will focus on finding the best moment to create process memory dumps. We will do this by trying to find the most likely moment during the execution of malware in which the configuration data is present in the memory. This information can then be used to implement a more accurate version of the interval-based memory dumping method in Cuckoo Sandbox.

A. Research question and goal

The question this research will try to answer is: 'What is the most likely moment during execution of malware at which the system memory contains the malware configuration data?'

The hypothesis this research will try to prove is Creating process memory dumps during the malware execution increases the chance of finding configuration data compared to creating dumps at the end of execution. What the end of execution is, is determined in the scope section.

The goal is to find a moment during execution in which the configuration data is most likely located in the system memory. This information will be used to develop a new analysis module for the dynamic malware analysis system Cuckoo Sandbox to automatically try to extract malware configurations from memory dumps. The development of this module is not included in this research.

B. Methods

For this research, ransomware and banking malware will be used. For each type of malware the dataset of samples will consists of two malware families.
The samples will be in the form of executable files of the actual malware. This means no malware that will still need to perform some form of exploit to be able to execute will be used.

To analyze these samples, a modified Cuckoo Sandbox instance configured with virtual machines using Windows 7 Professional 64bit as the operating system will be used as the analysis system.

The modified Cuckoo version will support interval-based creation of process memory dumps. The time of the intervals between dumps will be decreased until the configuration data is found in one of the dumps.

C. Virtual machine configuration

The used hypervisor for the virtual machines for Cuckoo is Virtualbox. The virtual machines will be created by using VMcloak [10], an automated virtual machine generation and cloaking tool for Cuckoo Sandbox.

All virtual machines will have the following specifications:

- 1 CPU core 3.2 Ghz
- 2 GB RAM
- Internet connection

The installed software on all virtual machines is:

- Windows 7 Professional 64bit
  - Without any updates, including Service Pack 1
- Adobe PDF reader 9.0
- Adobe Flashplayer 11.7.700.169
- Visual studio redistributable packages: 2005 - 2013.
- Java JRE 7
- .NET framework 4.0

Using Yara, a tool using signatures to recognize data structures in binary data, we will determine if a malware configuration is present in the memory dump or not [11].

The data collected by measuring the presence of the malware configuration in the memory dumps will be used to create a timeline containing the average time in which the configuration resides in the system memory.

D. Measurement

To prove or disprove our hypothesis, the time in seconds when the configuration data is in the memory during the execution of the malware, is used.

The system time of the host system of the Cuckoo instance will be used to mark each memory dump with a timestamp. These timestamps will be used to create a timeline of each analysis. On this timeline we can mark where the configuration data was in-memory.

Only dumps created after the moment of infection are used, which is the exact moment the first process has started inside the virtual machine.

E. Scope

Our hypothesis is that dumping process memory during the execution of malware increases the chance of finding configuration data in the memory dump compared to doing it at the end of the execution. In this research we use Cuckoo Sandbox to run the executables.

This research will only focus on ransomware and banking malware.

For Cuckoo Sandbox a maximum analysis time of 300 seconds will be used. This will be the end of execution. The default Cuckoo Sandbox analysis timeout is 120 seconds. Most processes have exited after this time. As a margin of error, we add another 180 seconds.

The dataset will consist of two types of malware. For each type, a set of samples of two malware families is used. The banking malware families are Zeus and Vawtrak. The ransomware families are Locky and Teslacrypt.

F. Research design

The research consists out of multiple steps which are explained below.

The first step is the collecting of malware samples of two families for ransomware and banking malware. An external expert malware researcher and Cuckoo Sandbox developer will assist in collecting these.

The current Cuckoo Sandbox version(2.0-RC1) does not support making process memory dumps at intervals (timeslots). This means a modified version of Cuckoo that does support this will need to be created. This modified version will then be used to create a Cuckoo instance.

Yara, with a set of signatures, will be used to automatically recognize configuration data in the memory. These signatures will be created by first analyzing two samples from each malware family from the dataset using the Cuckoo instance.

The goal of each analysis is to find out when the malware starts communicating with its C2 because at this moment the configuration data should be in the memory.
The memory dumps created at this moment will be manually analyzed with the goal of finding patterns usable to create Yara signatures.

When all signatures are ready for usage, all malware samples from the dataset will be executed on the Cuckoo instance. During this execution, all memory dumps are created and sorted per analysis and grouped by family.

For each malware family the Yara signatures will then be used to verify, for each analysis, in which of the memory dumps the configuration data is present. The memory dumps are sorted into timeslots which can later be used in a timeline.

The verification process will be used to conclude in which of the timeslots the configuration data is present. These conclusions will then be used to create a timeline for each analysis. The timelines of each analysis will then be combined to create a new timeline for the malware family the analyses belong to, showing the average time in which the configuration data is present in memory. These per-family timelines will be combined into one timeline.

This final timeline proves or disproves our hypothesis and should show the most likely moment in which the configuration data is in the memory.

Presentation of results

The measured results will be presented in a visual form by the use of charts and graphs in the form of timelines.

III. Data Acquisition

The collected dataset consists of 460 malware samples. The malware families included in this set are Zeus and Vawtrak, which is banking malware, and the families Locky and TesLaCrypt, which are ransomware. The samples were collected by first searching for file hashes, which can be used to uniquely identify a file, and then collecting the corresponding malware samples with the help of an external expert malware researcher. The researcher used his privileges with the online malware analysis platform VirusTotal, to obtain the malware samples. All collected file hashes are listed in appendix A.

A. Attributes

This section contains information about each malware family and the collected samples for this family. It lists the amount of samples to be used, the file types in the dataset, and the entropy score for each file. A malware binary can be packed or encrypted in order to hide its code, any configuration data, and any other strings. These are the samples we want to use because they likely contain some form of configuration data.

Compressed and encrypted binaries have a high entropy, this entropy can be measured in the form of Shannon Entropy [12]. The result of this measurement is a floating point number. Using this number and the table mentioned in [12], which contains statistical measures based on different types of data, we can determine if a sample is most likely a native, packed or an encrypted executable.

Zeus

The hashes for the Zeus samples were collected from 'Zeustracker'[1] and given to us by the external malware researcher.

The set for this family contains 115 samples, each of these files is either a PE32 or an MS-DOS executable. Zeus is a banking malware family. During execution it will still need to download its configuration file. The URL of the location of this configuration file will be in memory during the execution of samples of this set [5]. The Shannon Entropy score for each sample in the set is 6.578 or higher, the average is 6.936. this indicates that the samples are most likely packed executables [12].

[1] https://zeustracker.abuse.ch

Vawtrak

The hashes for the Vawtrak samples were collected from [13] and from 'SSL Blacklist'[2]

The set for this family contains 115 samples, each of these files is a PE32 executable. Vawtrak is a banking malware family. The binaries in this set do not have to download all their configuration data, part of it is already embedded in the file in the form of multiple URLS [6]. The Shannon Entropy score for each sample in the set is 6.413 or higher, the average is 7.039. this score lies in between the score of packed and encrypted executables, meaning it could be either [12].

Locky

The hashes for the Locky samples were collected from 'Ransomwaretracker'[3] and given to us by the external malware researcher.

The set for this family contains 115 samples, each of these files is a PE32 executable. Locky is a ransomware family. The binaries in this set do not have to download all their configuration data, part of it is already embedded in the file in the form of multiple URLS [14]. The Shannon Entropy score for each sample in this set is 6.367 or higher, the average is 6.999. This score lies in between the score of packed and encrypted executables, meaning it could be either [12].

TeslaCrypt

The hashes for the TeslaCrypt were collected from 'Ransomwaretracker' and given to us by the external malware researcher.

The set for this family contains 115 samples, each of these files is a PE32 executable. TeslaCrypt is a ransomware family. The binaries in this set do not have to download all their configuration data, part of it is already embedded in the file in the form of multiple URLS [2]. The Shannon Entropy score for each sample in this set is 6.068 or higher, the average is 6.777. this indicates that the samples are most likely packed executables, but could also be native executables containing compressed or encrypted sections [12].

[2] https://sslbl.abuse.ch/

[3] https://ransomwaretracker.abuse.ch/

REFERENCES

[1] M. tran, "CryptoLocker and the Rise of Cryptographic Ransomware", Tufts University, 2014.

[2] J. Wyke and A. Ajjan , "The Current State of Ransomware", Sophos, 2015, pp. 36-42.

[3] J. Long, "Locky: The New Face of Ransomware", East Carolina University, 2016.

[4] D. O'Brien, "Dridex: Tidal waves of spam pushing dangerous financial Trojan", Symantec, 2016, pp. 21-24.

[5] J. Wyke, "What is Zeus" Sophos, 2011, pp. 10-13.

[6] J. Koustek, "Analysis of Banking Trojan Vawtrak", AVG Technologies, Virus Lab, 2015, pp. 10-11.

[7] J. Wyke, "Breaking the bank(er): Automated configuration data extraction for banking malware", Sophos, 2015, pp. 5,9.

[8] https://cuckoosandbox.org/.

[9] T. Teller and A. Hayon, "Enhancing Automated Malware Analysis Machines with Memory Analysis", 2014, pp. 1-2.

[10] http://vmcloak.org/

[11] C. Roberston, "Indicators of compromise in memory forensics," SANS, February 2013.

[12] J. Hamrock and R. Lyda, "Using Entropy Analysis to Find Encrypted and Packed Malware", IEEE Computer Society, 2007, pp. 1-3.

[13] E. Sahin and J. Wyke, "Vawtrak v2", Sophos, 2016, pp. 34-42.

[14] P. Nelson, "Locky Ransomware Analysis", Stern Security, 2015.

APPENDIX A

Locky Hashes

01e2b6fb23d4a6b5250e95fdf47f0d01
053d6ae27d906e6303dd5604262ccd31
05f96e4199d83caa6f5e189016215e45
09f95bd2323574b6edeac8f8e349e4dd
0aa56c23cff79948f977ebd1a470b4ad
0c8f52995d8303837a3be33246658e0c
0fb871b4b329003dd29ed674228e0206
124b76844281e9067654506429437545
1311d9372c3550300d400c3fe83cd867
137e9311d5807974eabb5fa394de0a15
150ffde680083d6e8d814d93fdc5b5e8
1725b728a5225a47e3e6fc0092281071
17e23f44a0bcf6b27f480439714b4688
17f493da40a77f6bc7940f3166e9d89b
1b9f7d4c8a918cc8fb1cddadab9ee81b
1cd414da2994719c23c85f076efed410
1d0687fe7c7c5591f1049ecb84e8cbd9
1ef99356e1b53b6ccc163364a5418bf4
1fa1023e66b05db04d22ed4f57d37651
252957f37b8bd7a57473eab5f1a65d5c
254c5c13dd02a9fccbe2a40ac0b04355
264f16260d200d0501cf83220ed9a30a
27a6dbbc09d4fb7e0912e9fea078f5db
2809b79768b898adc24eb276e5866ce2
28b86d53228b2f5b042db52c3a6341fe
296b0a81f0925c95e01839690c0934bc
297529814d8d292594a1981fad30daa6
2e96ae4983cdec64af16788300c50e9d
2f6a0dd5a5967e533cd78f1942b95b50
3258e4be68770b76315a5059a0cb3199
3b522c3e3fc6cf29a2c8c65a80f14a08
3d91e5f119093cf1639f8d38b35d1742
3dc8d7cee33a5c2fc39c5386146a1d35
3e733108c36c9d407f7a40868d251911
40849d82a14058cbc91d0ecea473d1d8
47380d71be72bb4ff55b5e51f8bdc963
475bd8f697f7ebf88682be5458e4cdd2
527290686ec5515f248d4d20c3bb29df
5341fabc65b3984bf5e0d1718983020c
5695803a695e1dc6443ea09572e6b14c
579eb4f17d08c5061d8cf71b96436a8a
5a5817583b302c651a71f0ddfbe33a6d
5af520ad3507da22aaa756357e78eb57
5db257bb49dddf90d093d17b43eaa9c9
5dce19699be78fa82e32a96aee436c44
5fbdf068522177ac6392fa8f8689ed43
5fc1ccd8530954f61ceeafb77e72045e
6159c5d8a54ab76dec48a795b4b73318
63b695765260d6d1d2a5e5fb88130dfd
66b17e85d778c8aa51ef635858faa8a3
680a02bdae6724c537053f0590b731fd
68aa9f8fdb7c43ebdb4a7b3a6ceb98d2
710f6476ca3029e2017e6472b751127d
71b8d35385ca32cd413c8f708e802c9b
72dd8bc7871f07e2d6320270c60ac451
768b0a09344df69404d2466c5a45aaf0
77287dec5a92a3163c3c88ddecc8ba50
7822f2ca1be80f98649a30fe5441d0ba
7e409b55d878a463e974b50c92cb172d
810c011911151d3e8a064ad44a600421
81e85dcaf482aba2f8ea047145490493
82f32982439cf4fa320a0f9a8e4adc98
85af825a34e5b0c000c6c4b4fa065d82
86b735f30639165462f709e689daffcb
885c4cfbf0b9b7956adcbd5b93688836
89b2bae66f6a8e24396fba2dfa062227
8adbcffe2cb52628afe8d6412c1e3a06
8ed052b3c5c92727f385761786e16eb6
901ef89350b60a992e1a6c67a61dcdab
90eb8948513e21a8c87f8295ac7e81f5
9e3588c7245e6ddec33473e5b4975670
9fd4d9c87668844d3f645b6877d64d89
a01d60682ad5fadc9018908185e8cde3
a1e5fbcfee3aa4a025954774493edab1
a2236e65f3d0849ca2b85775ade093ff
a2c8be7f272bdd1191bdf112ba1aa9ac
a5116b31ab81b6c8fc0c9251dbb6f315
a56722d826d5f222a8385cbc5666b63b
a6bf89594d36f2f5c499efde3c584bd0
a8344ecd79b1c1371526ee06a0a0fcd6
ac4e4c3cb5cc6c068466e937f48adcc8
ad2022689e5de22e2e706b065e148c25
afd40dca335530ec993d9cf91be96b4c
b1c156ff3c59f19e30f96545bea247cf
b1c957ab802f39839f2b92d7d55e7f83
b97ed89e814ad91338a6bdd5f7853566
b986b4396f001e508898baa4ba71367f
ba5a6500cfba0e674f8bd2a62a308ae5
ba7505460a9c758c25bb3de1208cf23f
baaaefc5706911cd0a797808e23544ae
bf1890c2109ac0d6eb6183a98353c2ce
bf664e4a8f36d2de1614984e6982bbb2
c49177a1553b3240dfea69ee09d58b70
c6b7f5336ae4b985b0b523f3db76adbc
c6ff697b6c1b2164ede3fa5fac0e127a
c9be9e7751b8f164d04a31a71d0199c6
c9cbbb6364f51f790585aec9f5fa98cc
cb3425d0e436e358a07c3f38110135a1
cdd120508a1f0ff0b5b18497d67ca349
d35d938cccbccb5b84a19d2271c97ae7
d6e56a430c2c53104ca5b0cd092875b1
d72d9aa76ab313d50f059774d78875de
d98e82be5222b3686d58a625c77ab488
dce1cd7955c0352f86ecd7364f4a3fdb
e4ad8906a152085e7564252b47f6c10d
e52cc2b7136c572838b8a9e2b021bd5b
e57c0d32918eabeb319d1ee52d11df14
e81a50d312fe396641fb781a63667f3e
f08c3b7ace25f1ce76bebd3429762ef8
f0d96cecf681e1f5f1b7dbf9f6a518b6
f5fdc2a9d330a7e607003445edd9dfc2
f7bc8f3b73313b238944e4812a3e4975
fb469897a4536876306ae78e18409be6
fe4985beae55b054259bf14d3a3e50a7
ff06ce7adf8f6cf1973a6845859ff0b5

TeslaCrypt Hashes

005a6963097536bb687192aa0247ef3a
055e612b2818622f50967cf098427c17
071313017f6e276ba2a800ccf0362014
08a01b8e22656b17e7effcd8ee171e5c
0ea102aa6196a11eb84cb5a4ecb60f33
0f3f0d90419cc8c91ddcb5c430867707
118064f032f310d3f4af292600f7c7b6
12562a42cd391b27ddafcf8a0b0cfde3
156aca02760e0d7ed1934b666794c45b
1571350e00ba56b2bf23ac751053639b
1938b6e247a686839564ddf99a1c27f8
1b46211a12fc2e419bc77feaddc68053
1cedac81bec37becff994f3bd16b1a92
1dc7fb65a936731dbcaed723419a12f8
2180152ae725f04088dfc9eae68066a6
230c2660e50c7fd753f7a57393d1b327
25c1d151e7377c0cac6572edd3df4a90
285f9f45e19b7933b39d23b9a3ed0563
2c5adf1b60ac6a0ed1c8872012ce0a82
355441d76229368f22e0d903558f1c24
3ad4b974967a3da9767ca80e3227ae4d
4742ed29ab69e60405a75178c6685c9d
48cc26a4fcc6a3a09bd81bf8bf773531
48ffaf220f167c19ad9888dba639c0d9
4d20cc6b4021d176e806e96952ce3019
4d8339c4ac3225102e60e4885b443268
50aab7f0c52691e1e66d544e43e16987
517732067df17a66f8dc2fbee8ac97c2
518e931a049d7d64c381dbebc96cf19d
555ea3963b253263ad89ffcc155d67a3
55b7a77b7c43cfa416a679d8bfa6dfbb
5aac7e400fefb6864971a8262a9cad06
5d5b6c2ef3a7b934ed657681f09abe5e
5f1db9e4cb450d750124e244b520c514
5ff6f37db3d56c3abe8dbfb938b11262
6981409964483b4b7624c1a85f575e21
6e72482c91f40143f2014c3fc4bfca0c
6f9532c7b7d3e508732ded880e6cf831
715bf2f0f539b49f5eabcd784ac4175c
725009eaa8a92d1d0cc46d70154939a9
729aedbd826f86b213cfcde2fc1e1743
7444ace98a3e2e588a39fe2d54180530
74bdc7b3c8b04d65de527ebaa5e98c6f
776e822a7d178b76143ce058c87fede6
778d9b3bafbd4a0f74cc32f2483fe0f5
77d66df823582d46b5b41e6837b89ac3
78d8fadc8ddd5f17dcac4411f145c92b
794d2e712252bc8bc682f25cc66c3f3c
795352e779a0105e4c644df3085848a5
79f170cd385b217275e7c7db63899eae
7a0e726d058c1d95c8e09b4394d4e5ea
7df5f6d611eb67f481a95804c5ce0630
7f70f850b327f27e10623d7e13058c3f
8691b47066dcb4605cf7e82b0889c3f0
8737dac549e33c49facd8d000995bf97
880130a4c1a01e4c611f85576cfd9661
89dba5be2371073843a4cb8fba81184a
8c789cb4e1de85176d8883314e0be9f1
948b2d66e8db1fb7f4c017ce4e381098
951c86cd98b1e181503167299d410857
96469329287b7e2cd7fdfd916addc1a2
9659d8de6bd1a3648c7a67ee538683e4
99333789293e2198b5c75f69cf978e8f
9f82d05168c593b297f222e35f75d1f2
a0f02c3c9bfa3845099f100931b4242b
a100eff9dae36b7b3bae3ed072639d99
a1559181f4d306118f589dd86a8a3c30
a56ad8c98d10a1dc1287983bcc6e11fa
a5bcb278884567194108fedcb797e7dc
a601cbb0a15bf9cb4cea80be4b6dcfe8
a8aea9e1d840359af9d7ce6155ff244b
aa0ac61c36b7cff9cd85a9a6d792c45b
aa7092e36d6ac9f885b8caaf70b57ed4
ac4d46e5c8b04bd99b454237bbf5f6fc
b0dfa978aaba1f0ae0c79d05b8f804e1
b163e8fe594af8ba6a429b32940df067
b96fd404534f155f7566947308fd220a

b9c1e09b996165cc19150c46ca42f789
b9f41073806c83a9d44c0c66ce8b55eb
bb75e0ae806040be55b40d452686d770
bc2ed64cb7d0243c7b3c131263077293
bf7118e2fe12f38fa2c41ad3cc843344
c3a4b8692c6bdb0bce16067f150419da
c3cb934d93db52b22c2ce7d03abe8417
c3f42c89da95d0a85c788eccdb9b2e6a
c4b14d1478cb8f2f03be946880c8eb96
c589dca14b4333fce03b1aeb57a32b85
c6b73f8ce2f66b0a7c063c2347d732b0
c7eef7fb8f346f150cc23db56eb6ebce
ca03464f0ed7235e14485c8e753ea572
cab329ec48c90638556a99e1654d0509
cb9cfe955cfe5fa1f0c7bb0d6624e41b
d15eebbfd21479858c044faca86b7756
d5289115d669750f4f8f448842eee8a0
d58e0347714ba9dfa46bc33691a02d87
d5975f208ac5783f4b58e8632cb17c47
d6609f0e82f65102b7d33417f2e8c599
d7a72a833bbca58196537a6345b9f4ed
d907a9616fc14126653d03772fc7de7a
da28166f9ff6bbea1c23b63dcba842f1
db9a3ebf29d902e3177a26aa8dbc2af2
dc0416c30fc4bd13d3527c6119cf9ace
e058ecc0e3853de1f8b727f3cf34dce4
e6a9e9e261b7bfa13b9f51b7cc466360
ea7a382a75db24dad6014a4e9c5e0ceb
ed07cf9bb0776107b2c3eedd2fdb8064
f27f3bd810aece963a520583c6481a88
f417f72aa747d2b41a6c9467a027786e
f50c3962458aa054a20d65b061f17bff
f50f693ba762bfb674004a49560a23af
f623aa2147c701b68f8db4e3bb36701c
fa67597147f38665cf7470f9936b95cd
fb6ac8d602af2363ab5c5e48525838b1
fc00d65b2fc6d46826a4deaebac54896
fd7c988278891a9e0cf174ec6c2ea182

Zeus Hashes

bd07b2a48e75fa30f66be2e1b0b38301
8c1265d8904a7a58e1346442cae5e170
065e6b516c4fab893826103db6aeb5dc
09eb0efbb48e7efe2e19e71edd655f3e
08105db8efec4b2515c00ba92ee8bb47
0d8f9c5898596251233c3fd1dcb34161
122d32cf91a5f6a545496e0c7c64355f
14a18b30c40f5a4fafe08e0c21cc5844
177e77d48bdf6424eaf0bbbff2905236
199d04e319f6f8c1e88ab3ddc7ed28d5
c3b4dec5b7de567c7e27fc74e9604a61
1e0058d2e69f3bc4b961451710e2fa06
1fa4764c0c1eae57af50d4a5277886aa
237031fc5cf6b5ed59d8e750b860341f
5e6724fc9399fa65783def8da17c9c9d
6ddcb9b8cd612a955f1e4a97038776b9
29c0e993e5ff6106d93be0b54282831f
2c0244c28036f9cb5f9a703c8b329f2f
2cb6faf81fc9cb701e71d0497a25f1d1
6ac99572c9c7e50f9a26585d29c78b28
1e1b539de3866ab173661a3425e18932
39e80644d53c68c84f21b98f6e556afa
322c39b56988f5a35e64c54633c196ee
3356ebab1bc8aa9e6212f584bdfc7566
338caa225b3906be3bf5399d8cb74df9
33fa98383d855527c1a166ac61f92ec2
ed2f75c41a4db77b4efc83745b2bca82
353f3b54de9ecfd82c63a2aeaf1c3b9c
63f854d00e2b61f5fcaf42f5d0048272
392b86e7d4c9f28e98862e39cff6e49b
06d99e5c2ab8a0669aa1a14ab78c43b5
3a2c75fef3be79b5d8662f121f85b4bb
80ff54e79c59ecc802e96f45c7a2a3ee
f7f79d8821abd3035a3c77b4d1319334
70455144019254c1e663305c2f429893
461f8cb7c8f1dd63b062fe726ea764e2
46ab2d15b560b7a07d39862907290220
c2c30e59e9df7d8212f882873ee2e46b
edd4b084fedf2854d02e35a502ba6f3b
493b3700a1ac3b5b872bf2a516bcb701
8e0f37f2d36aada0ba87f5e66a3c27ff
803bcf844b545a52422deb299b37e034
4e9a5f7e45043f7ce0e2822b947fc117
f8aa5e18c64caa327f9ba3619b5786f8
50220851ac85a9422c35966b433c203b
5054c0c2dad7207eb1aa69f5c48c978f
6abe6beb6fbfff5d36665bacc52761ed
50ea80fd625cfbb549d4cfd60056268a
d2b4f845048382441091f13f2fe14a37
52be0408084f536e42feb7c57f521592
429ebf3b919d8959e39f5c90b22e81dd
ec2d2fe95511881d7b159d19cea3ad23
0074eeee2333c762aca2e61506b605f9
5746dd569623431ba41a247fa64847d7
63992249e966ff33d7555e887ce28595
65b16d40f024b5c1cf8676dc1c252d56
6da10c3719c7bfc5617f6095d08854cc
70a0f9cef4d7a4952eb659b049e98fc7
7331895ea0d778ffef3ab95d3e1c355b
10fb8a43c5ec22df96ee00c535a9d91e
78c00cd8930229a1d34b334f983a633b
b3014782ac4754655f4e4cdc6912dd1e
7a6bbc32868a9f776452355f909f95d6
7b2c587c79dfbdb60d71c0144a3e2ece
7cd6c4a6103f23858c7ed047391f1d3b
7e3b8c6062f7f11fef7cd66d068539c7
805df9572b345cc8691198ed1caba924
82f34481e82f289f89ef69e4eb2abb3a
0981cdee9730fdba83d355088b9253ce
86255ec982e822f6b57855d3866618ae
8802d13595da8294c84821e5e3086442
cb9cc726fc2e79877ac9d6d79ceb2ef3
314ebe53dccaf8e8ef20041b6b2d200e
8fad5dfac3671d90dd12792ec81fe595
96383909da192e14760de588761e38ac
9b2162f6148f7ba9a15e2b2424952973
9f890fe67151372e2dcb34d4329eacc6
a79089b5e6744c622d61befa40af77d3
e0d1c49f07bc85f29b744c8eaecd623d
ac2d2fe0bd0c40db038c88cca4a3296f
eb8cea5697eae596cbcdb9fe66bef95b
2d4cdd88b7735780678f1d64c6644cba
8eba22e4dc4211d81e4d88424275b8b9
e305c642a23e5508cff32fac40bd7fba
a99644b1df2f81f1c4a99d65c26a8a79
b73aa307e8c2328f6a7dfde1a1f024fc
ba57db487bb18b15217cbb08c923da50
bd6466701c9e93ab24d77c34d44106a7
be39759c2e6f2685097deae282692851
8febcd2383f7cde2381b9fd969ef6080
c30825c55ddd1b3d93ee6141d44c78ef
6f70a39e1efe106cc1c9720f56b5234a
d2d969b5db07a1dcffab0831907d31b5
d725561817d04a1dd0c889781613b577
f55574038afb0121b29354b58c6fa26c
d86ec2bf5962d1254e08458b17ff9594
d87a03973f8cb42b90a573f831d29bd3
c7b4b8b541f3184e9a797570e0db948d
4fc4a7c030217d415f61c24c3d988175
dcdf3aaed2047d2bc746ab3200667261
d63b0008dfe8d156605c576269ee8bda
e2190f61b532bd51e585449baae31bc1
e015577954d006f81726e2e1ebb9fd99
b5e1caea46402cbbfcb8ca059820ce9f
2055b4979a44542e31f79bd799bc8190
e7c054ea8bc2f66e914ef82841d329fc
e89261b86d55db32261ea9117fefb1aa
915eac13cd5560377ee8e311ab1f4141
ebffa1e446ac21950941ae3463aa2df2
eccfe46a97deef63172aac0ae8771d9a
e9757a42b52258823bbb54948b285acb
a7d168823ab98e958176e4b7b54d95cb
71b2be350e0564459cfc978e7a393cd9
f76a509fee28c5f65046d6dc072658b2
f7d66e3f5e8bae163b388eb2234162eb

Vawtrak Hashes

8dcc59d80187e7092cf7a42af25aef49
628e487c639a60f018857b509dbd2fda
8987150ad8cfa3573c48be37749afa7f
9fdc12aa9f7e7ec1f9cbbda842d29cfc
dcca66d2f152ed0f30689bf957a51996
1f7a111d89308a76acd71fb540296300
4dff9c669528db097676d5103f940038
d3938c66b07729195fd98370fed5a163
d19fae78c3f0c118727fe4e3a36371e2
0999aac89b5512f4741d66d0fb7736ab
d0ca560b91e7d8661cc6982f1770bfe0
96ee1fdf55edd1814eb101394d697b2b
0518a55101fb8d32a2a0a87476e11c58
2d3eca4cca62ba80b90845d84798f445
03cef21c061b7e2bbacd3565e4cb6ce2
dbedb7a1d28374944d810f79a35aec4e
f22ebe5ff3430b3f76525b237d1666ab
0d218cd48a8f047919a3580a6c0bf18e
af87caba6485c1f0a973d3e1b375bf85
aa78a033b2a12a09e1abd46f14ac1094
b6002927e80fb0b2b6dc47ae1812e593
a4b639f0e006f26a49e060cb7f1fe1b5
af7fd98c9da738526e1dda0e1097c64e
de34014de4233e19b9b2f79f1aed660e
bb1048e1b097a6f8e859f64f4c476d96
77dbd2add7fad4917b51ad23eaa6759c
13d268397be051fa4d4cc77e9c993693
114a7715e6a392e55163fcdfc1e732ca
3167a0e34496018beeea371d565df7b3
ec51d6c059dc383b83c82ce4e29f3a31
b7d12a011c8dcfe7e951a1ffed6460a2
986a6172364d731a05a473d38a70c7fb
b1a525127b5f07f733c12c4b7cb58f85
222407dac5bebfc9dd4df5a30ad24b92
932482c62e72f05614d717b0b9684a16
a817ae1dba7120aaf9395618ff63d94b
c6c9660deddf8bd5c4b240ca8f1a877b
38df231463bf7a9daab863a701f7c5ee
05124b57116e4d934a50e2ba88a3eb7e
be1f9b5e50b4da5830b3f149bd0efc2f

11c4ea4c105ef6353f5b49674283fbad
93c9ae1339ff9574260ed816aae49232
b59e65e018a2e39004a1befb2a06f608
6280dbb8b2b9c5f312a69d864e8d15a8
2cb257c1ec6c4140f1a24b84310430cf
d1a2a48088fbceaf56cc8968bc5ffcc9
6315cc598f02690b0e1d5047c77bc0f2
7caf1331f54f81da0b4b6ebd0d611aa3
ceaddbd3682069488c857a80b55a6aa8
5a332593e734839b7136c30446a7cd1d
a4205a51335a9456b15937f3b04a9c28
eb90b5648469171d689a042be0118946
9bed000ece49f946e0dfc2a7aad8e743
486dc8c54dd42b13a9aa6cee9e68c01d
5d8d97e173bb17d4f551dc1bbbced7a2
7bff6c797530192b634e2f120e649671
20d09918e20cf8f9fd302cc3c767ff64
af614d0e25a70f598568c6f251bc3187
85e98a0325b750eb2a5400d3de373e72
78bb12d54259582ec8dd4979d5163a8c
ac03783e31b5fc2e0ebe2e77a6d49c48
0ca834d0eb27e091ca66a461969c24be
d56f28283054f4d2de073de22f5600d4
50752cadce59980232fddb70352d9a93
de82d9b5f27daaee2fceb09448a02255
eb05400ce2aebbda2168e10ac8c2b2fd
ef4cddbe5bfa035077168835816a6751
14061a4c43074f2a1ef5cec0577cef9f
f6ca365f77b05e6cb2e3a5b387d52995
6040df33d12de69a819812b7799d8bf6
439fe9f1777e285c7ae5dd989fba63c1
77eb0b78f82573099fde676e225ffaee
338496ffced59059ddb0deb4600d0599
b4cb2af3d9ca46dc2f3444cb1e568c44
853983fb7f6adaf029a3d2e56818affe
07a06e52f7da1804096cd7d8dd788b38
9803e48f685a521a19ec169cb8d0ed41
cd81ba783d6dfa812388a66fb0d88e91
848e99db1bdc96fdbc88d53693f29714
c2d1d5b08243a202f328bcc05a1b2ae4
04741d2fe07a12af4f67359b06323128
87161faa7c7edc37bcf3b5005390d32b
e9da6328f72f9023edb45c6d236f6c1c
211e9145dd724e538362face5740e6e2
6f43b0710c2b25f8eea9598e0955c297
af06284c0fe4c99d4d3f6b5b562f50a8
5ec10cfc4b29356d1bac2391e596f15f
554fe4d3403b8c9d14d5379368c9ab2e
864019d29a63b86bdc13ab7b05f8a8a5
048d559df99a7fee82fe5fd4dfee900a
110458278211d7f6f29180a78fa125c7
142ad2c753f1929a3407952fc8ac147d
16b691b9c41227fb9aaf592d7f49c722
199d642f5c50780045085cb5992a52fd
1d59afe9be5899c51c4f62aaa6536e5c
1dd40a9223e0c43e4e5890aa84da1844
209480561bbd613503a0950211588f4b
25f813e97409bf7808756f1913b11102
28b577ce059b5c3851b469911ca637aa
31b1e70519d0a8ac1303ea89a0d817dc
34080a4e7c2bb069e525e22e555f60dd
34976648b44273c0b336d2ef89e672db
3c8f1e08e774dd503d7528a1d6d49951
4172363f9ef187f0b26e04f9e331501e
4826e1b51599e3eeaa792e9621170324
6359dffafaf53f1d4b7e2d548a9556cf
6b87d33b169986cb34f913c14a547f75
6edaef466a97955a842f54e53f205991
786c43da9212a35dcd3364d9a09fe1b3
7ccc57f92ee1132e30141f22bbb385db
7d9b38fad4992247cab2663a1e6ed137
842e2cbdea3abc786332e1eeff20a59a
845ced65ee8d3ada63fb940f4dfd4e51
86af5b1b003fa2a570dc45ca247a6274
8927105aaf53fbb0495be81835474d74