BT Cyber Threat Intelligence

by Yanis Alim - Emir Hamdoun

LOKI IOC SCANNER

What is LOKI ?

LOKI is a free and simple IOC scanner, a complete rewrite of main analysis modules of our full featured APT Scanner THOR. IOC stands for 'Indicators of Compromise'. These indicators can be derived from published incident reports, forensic analyses or malware sample collections in your Lab.

LOKI offers a simple way to scan your systems for known IOCs.

It supports these different types of indicators:

- MD5 / SHA1 / SHA256 hashes
- Yara Rules (applied to file data and process memory)
- Hard Indicator Filenames based on Regular Expression (e.g. \\pwdump\.exe)
- Soft Indicator Filenames based on Regular Expressions (e.g. Windows\\[\w]\.exe)

Rule Sets

Loki Features some of the most effective rules borrowed from the rule sets of our famous THOR APT Scanner. We decided to integrate **a lot of webshell rules** as even the best Antivirus engines fail to detect most of them. We put **almost half of our hacktool rule set** into the rule base as well.

The IOC signature database is not encrypted or stored in a proprietary format.You can edit the signature database yourself and add your own IOCs. Be advised that attackers may also get access to these rules on the target systems if you use the scanner and leave the package on a compromised system.

Overview

You can easily add your own sample hashes, filename characteristics and Yara rules to the rulesets we bundled with it.

The most common use case is a so-called ‘Triage’ or ‘APT Scan’ scenario in which you scan all your machines to identify threats that haven’t been detected by common Antivirus solutions. You can roll out LOKI like any other software using your preferred method or offer it on a network share. LOKI can then be started via Scheduled Task (GPO). You can simply run it using the UNC path ‘\\system\share\loki.exe’.

Another scenario is the use in a forensic lab. Scan mounted images with LOKI to identify known threats using the provided IOC definitions.

We quickly add IOCs derived from important threat reports to our rule sets (e.g. Regin, Skeleton Key). Use LOKI to check the integrity of your systems fast and target-oriented.

How does it do the job ?

Detection is based on four detection methods:

1. File Name IOC
   Regex match on full file path/name
2. Yara Rule Check
   Yara signature match on file data and process memory
3. Hash check
   Compares known malicious hashes (MD5, SHA1, SHA256) with scanned files
4. C2 Back Connect Check
   Compares process connection endpoints with C2 IOCs (new since version v.10)

Additional Checks:

1. Regin filesystem check (via -reginfs)
2. Process anomaly check (based on Sysforensics)
3. SWF decompressed scan (new since version v0.8)
4. SAM dump check
5. DoublePulsar check – tries to detect DoublePulsar backdoor on port 445/tcp and 3389/tcp
6. PE-Sieve process check

The Windows binary is compiled with PyInstaller 2.1 and should run as an x86 application on both x86 and x64 based systems.

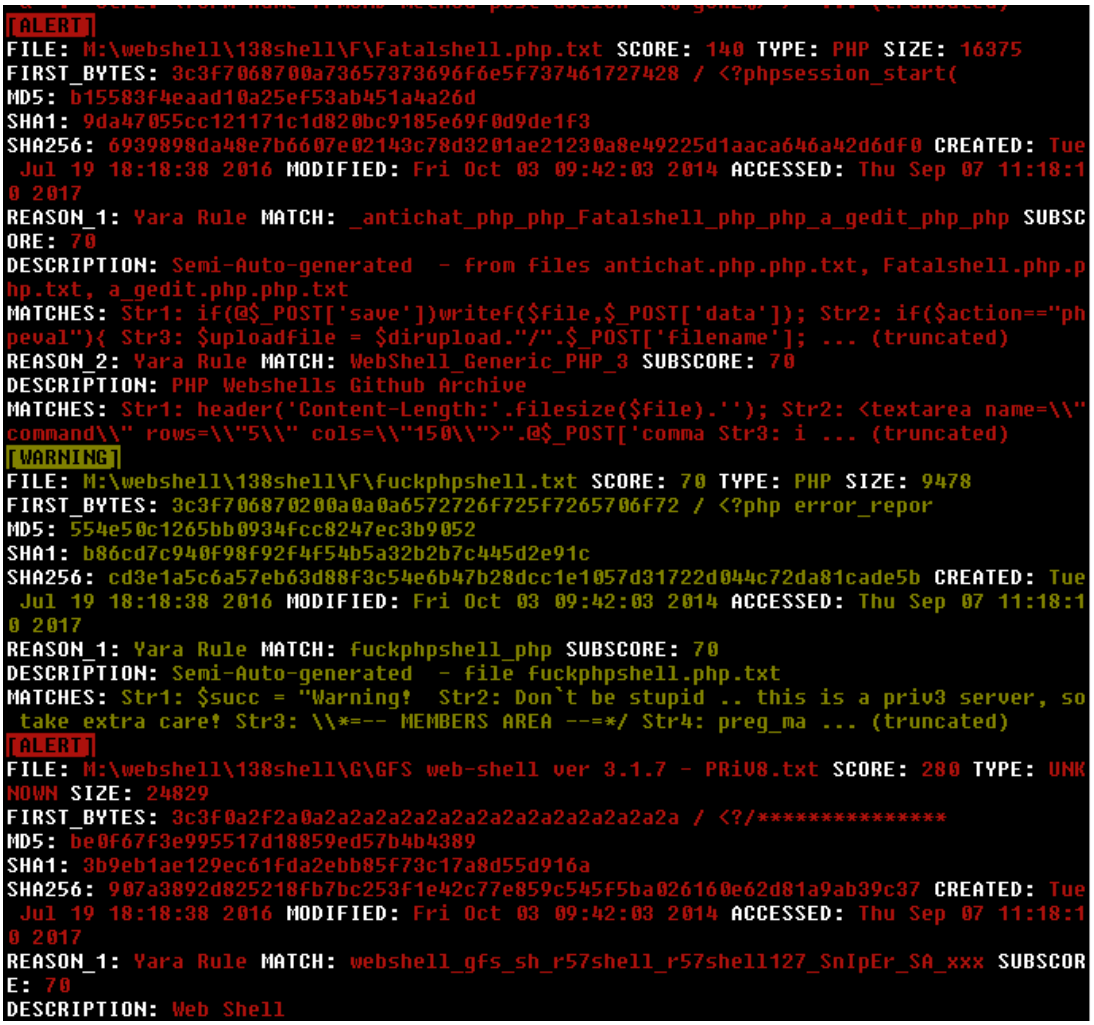
How-To run LOKI ?

https://github.com/Neo23x0/Loki#run

- Download the latest LOKI version from the releases section
- Run it once to retrieve the latest signature base repository
- Provide the folder to a target system that should be scanned: removable media, network share, folder on target system
- Right-click on loki.exe and select “Run as Administrator” or open a command line “cmd.exe” as Administrator and run it from there (you can also run LOKI without administrative privileges but some checks will be disabled and relevant objects on disk will not be accessible)

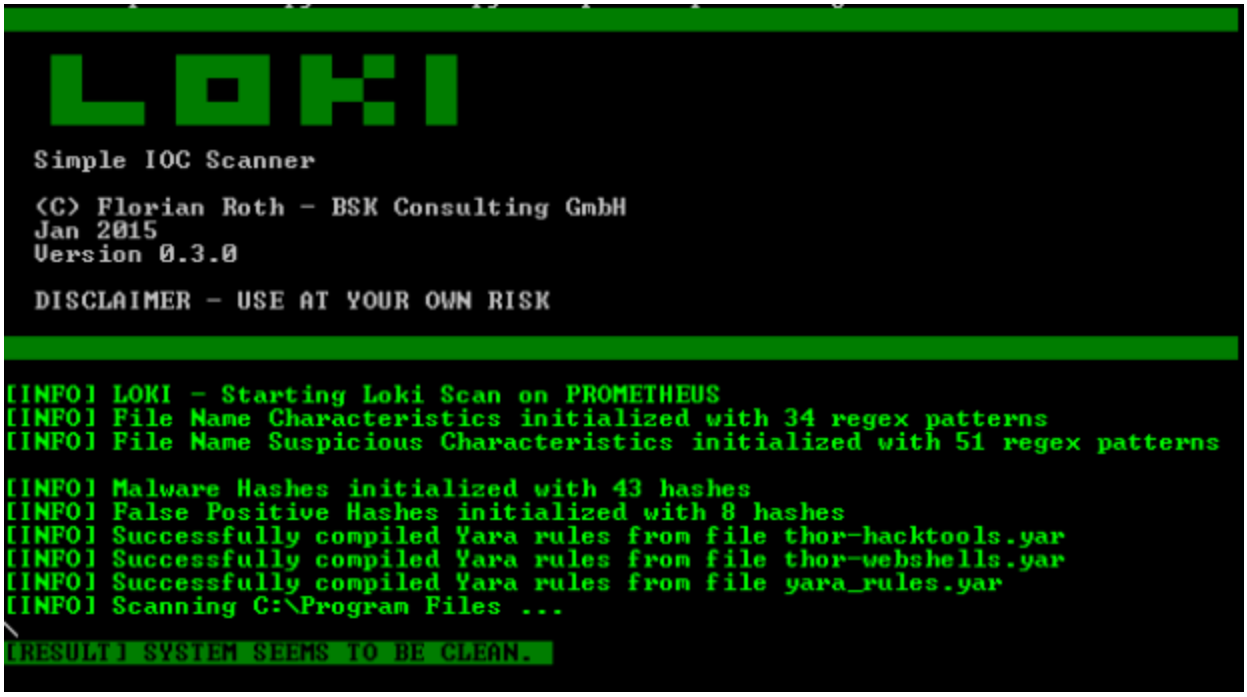
How-To Analyse Reports ?

- The resulting report will show a GREEN, YELLOW or RED result line.
- Please analyze the findings yourself by:
    1. uploading non-confidential samples to Virustotal.com
    2. Search the web for the filename
    3. Search the web for keywords from the rule name (e.g. EQUATIONGroupMalware_1 > search for “Equation Group”)
    4. Search the web for the MD5 hash of the sample
    5. Search in my customer APT search engine for file names or identifiers
- Please report back false positives via the Issues section (mention the false positive indicator like a hash and/or filename and the rule name that triggered)

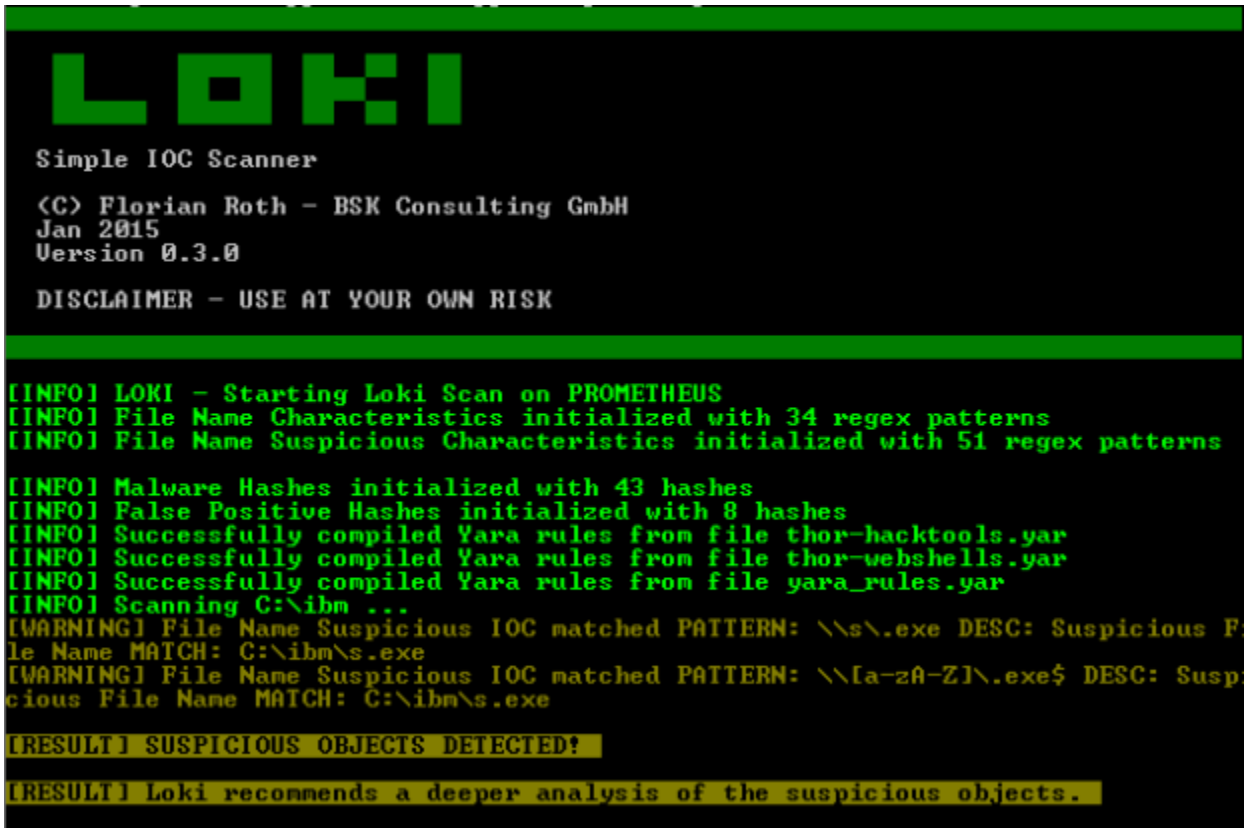


At the end of the scan LOKI generates a scan result. This result can be:

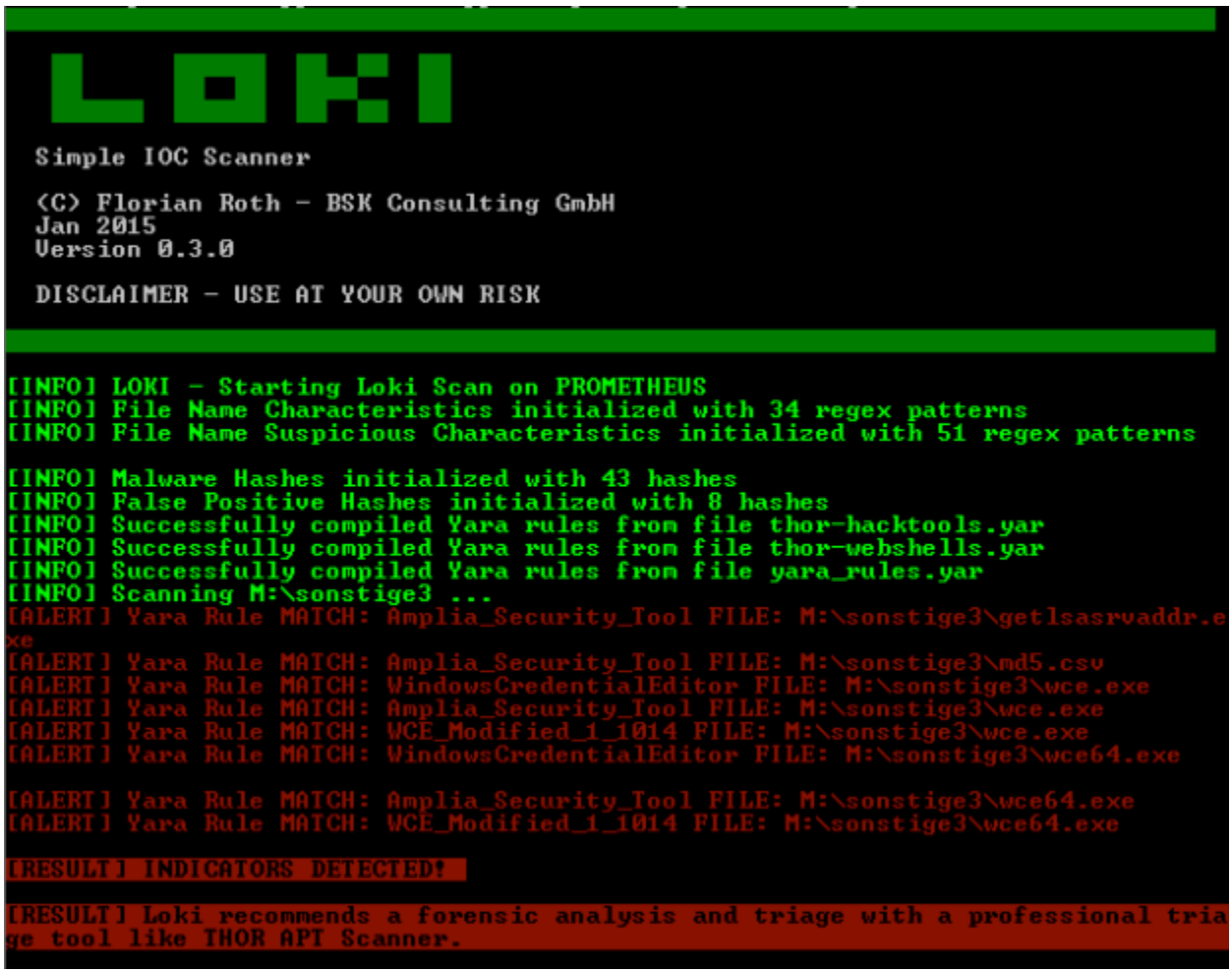
- System seems to be clean :



- Suspicious Objects detected :



- Indicators detected :



How-To Update LOKI ?

LOKI includes a separate updater tool named *loki-upgrader.exe* or *loki-upgrader.py* :

```
usage: loki-upgrader.py [-h] [-l log-file] [--sigsonly] [--progonly] [--nolog]
                        [--debug]

Loki - Upgrader

optional arguments:
  -h, --help   show this help message and exit
  -l log-file  Log file
  --sigsonly   Update the signatures only
  --progonly   Update the program files only
  --nolog      Don't write a local log file
  --debug      Debug output
```

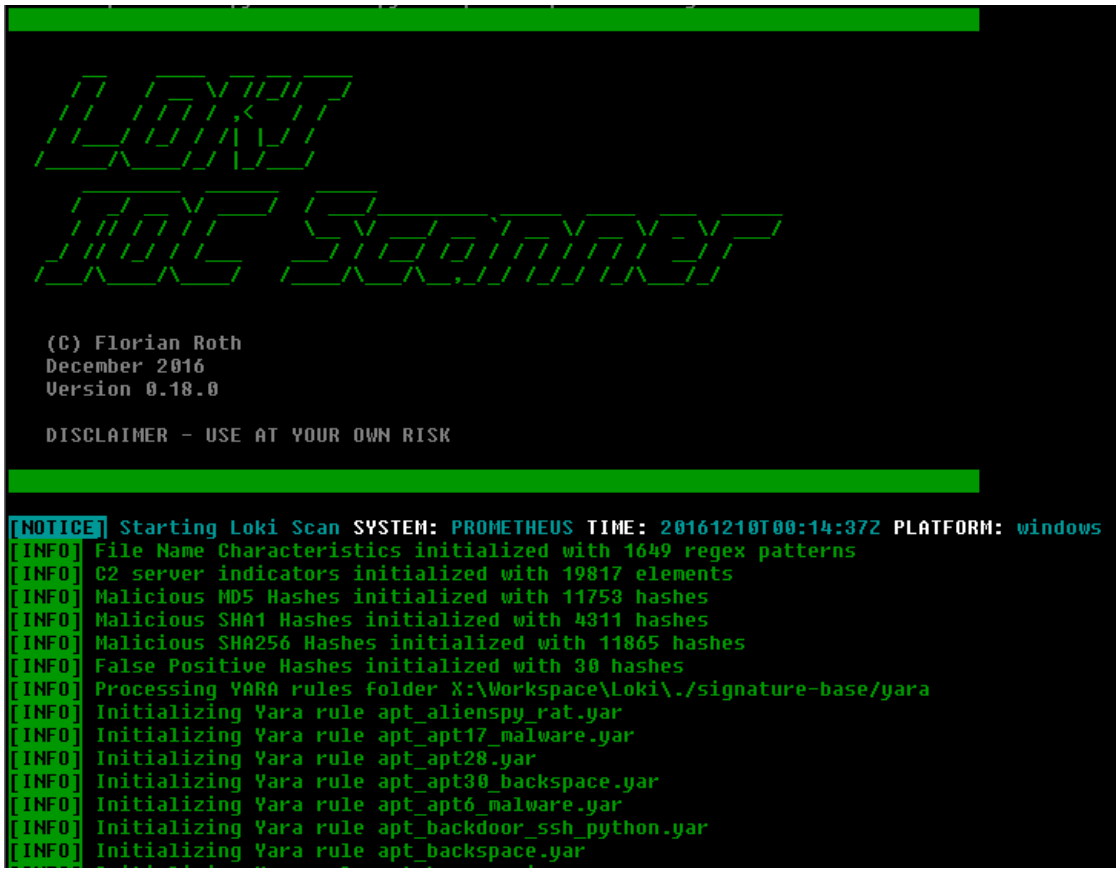
It allows updating the compiled *loki.exe* for Windows and the signature-based sources.

When running *loki.exe -update* it will create a new upgrade process and exits LOKI in order to replace the loki.exe with the newer one, which would be locked otherwise.

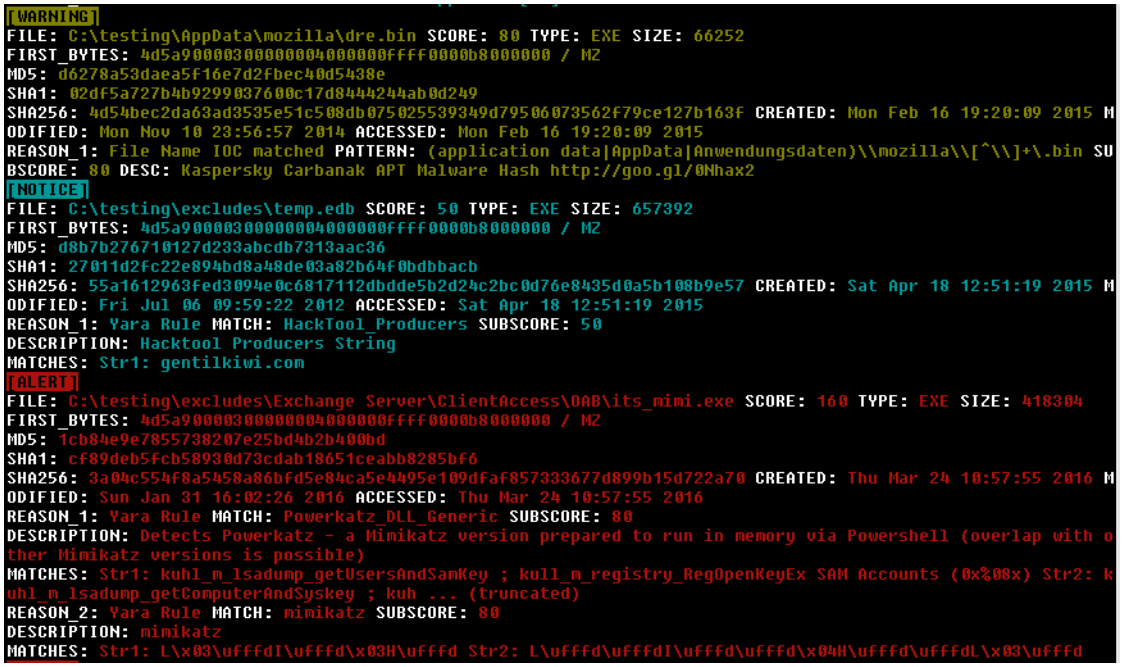
How-To Use LOKI ?

```
usage: loki.exe [-h] [-p path] [-s kilobyte] [-l log-file] [-r remote-loghost]
                [-a alert-level] [-w warning-level] [-n notice-level]
                [--printAll] [--allreasons] [--noprocscan] [--nofilescan]
                [--scriptanalysis] [--rootkit] [--noindicator] [--reginfs]
                [--dontwait] [--intense] [--csv] [--onlyrelevant] [--nolog]
                [--update] [--debug]

Loki - Simple IOC Scanner

optional arguments:
  -h, --help          show this help message and exit
  -p path             Path to scan
  -s kilobyte         Maximum file size to check in KB (default 5000 KB)
  -l log-file         Log file
  -r remote-loghost   Remote syslog system
  -a alert-level      Alert score
  -w warning-level    Warning score
  -n notice-level     Notice score
  --printAll          Print all files that are scanned
  --allreasons        Print all reasons that caused the score
  --noprocscan        Skip the process scan
  --nofilescan        Skip the file scan
  --scriptanalysis    Activate script analysis (beta)
  --rootkit           Skip the rootkit check
  --noindicator       Do not show a progress indicator
  --reginfs           Do check for Regin virtual file system
  --dontwait          Do not wait on exit
  --intense           Intense scan mode (also scan unknown file types and all
                      extensions)
  --csv               Write CSV log format to STDOUT (machine prcoessing)
  --onlyrelevant      Only print warnings or alerts
  --nolog             Don't write a local log file
  --update            Update the signatures from the "signature-base" sub
                      repository
  --debug             Debug output
```

Scan: loki.[exe|py] -p path_to_scan



Scan output:



Hash-based IOCs

```
#
# LOKI CUSTOM EVIL HASHES
# This file contains MD5, SHA1 and SHA256 hashes and a short info like file name
# or hash origin
#
# FORMAT ------------------------------------------------------------------------
#
# MD5;COMMENT
# SHA1;COMMENT
# SHA256;COMMENT
#
# EXAMPLES ----------------------------------------------------------------------
#
# 0c2674c3a97c53082187d930efb645c2;DEEP PANDA Sakula Malware - http://goo.gl/R3e6eG
# 000c907d39924de62b5891f8d0e03116;The Darkhotel APT http://goo.gl/DuS7WS
# c03318cb12b827c03d556c8747b1e323225df97bdc4258c2756b0d6a4fd52b47;Operation SMN Hashes http://goo.g

# 563d1512178cec1f6a73c98d565c98fa;Cygwin nc.exe example

5d853a8de18d844a9ab269f3d51e5072;Five Eyes QUERTY Malware20120.dll.bin
cc8b737edb3f11c9c5dba57035c63103;Five Eyes QUERTY Malware20120.xml
67ac8dc6589a07d950bd12f534dc9789;Five Eyes QUERTY Malware20120_cmdDef.xml
40451f20371329b992fb1b85c754d062;Five Eyes QUERTY Malware20121.dll.bin
ff0afae5c68c5177ed0a3d6339810cae;Five Eyes QUERTY Malware20121.xml
1bc8f4df4551c6efbbb1fe9f965dca49;Five Eyes QUERTY Malware20121_cmdDef.xml
0ed11a73694999bc45d18b4189f41ac2;Five Eyes QUERTY Malware20123.sys.bin
066b6253afc3ad0efe9a15cead4ef7d8;Five Eyes QUERTY Malware20123.xml
790d1b448e97985deb710a94eb927c27;Five Eyes QUERTY Malware20123_cmdDef.xml

ad61e8daeeba43e442514b177a1b41ad4b7c6727;Skeleton Key Malware
5083b17ccc50dd0557dfc544f84e2ab55d6acd92;Skeleton Key Malware

20831e820af5f41353b5afab659f2ad42ec6df5d9692448872f3ed8bbb40ab92;Regin Malware Sample
225e9596de85ca7b1025d6e444f6a01aa6507feef213f4d2e20da9e7d5d8e430;Regin Malware Sample
392f32241cd3448c7a435935f2ff0d2cdc609dda81dd4946b1c977d25134e96e;Regin Malware Sample
40c46bcab9acc0d6d235491c01a66d4c6f35d884c19c6f410901af6d1e33513b;Regin Malware Sample
4139149552b0322f2c5c993abccc0f0d1b38db4476189a9f9901ac0d57a656be;Regin Malware Sample
4e39bc95e35323ab586d740725a1c8cbcde01fe453f7c4cac7cced9a26e42cc9;Regin Malware Sample
500179379093900935 5ba841610412e0f8d60ef5461f2ea272ccf4fd4c83b823;Regin Malware Sample
5c81cf8262f9a8b0e100d2a220f7119e54edfc10c4fb906ab7848a015cd12d90;Regin Malware Sample
7553d4a5914af58b23a9e0ce6a262cd230ed8bb2c30da3d42d26b295f9144ab7;Regin Malware Sample
7d38eb24cf5644e090e45d5efa923aff0e69a600fb0ab627e8929bb485243926;Regin Malware Sample
8098938987e2f29e3ee416b71b932651f6430d15d885f2e1056d41163ae57c13;Regin Malware Sample
```

Filename-based IOCs

```
#
# LOKI File Name Characteristics
# This file contains regex definitions and a description
#
# APPLICATION -------------------------------------------------------------------
#
# Every line is treated as REGEX case sensitive.
# Every line includes a description that gives information about the file name
# based IOC
#
# FORMAT ------------------------------------------------------------------------
#
# # COMMENT
# REGEX;DESCRIPTION
#
# EXAMPLES ----------------------------------------------------------------------
#
# # Various examples from APT case X
# \\svcsstat\.exe;Case 1 - infector
# \\(server|servisces|smrr|srrm|svchost|svhost|svshost|taskmrg)\.exe$;Common IOC
# ProgramData\\Mail\\MailAg\\;Case 2
# (Anwendungsdaten|Application Data|APPDATA)\\sydmain\.dll;Malware X Case 3
# (TEMP|Temp)\\[^\\]+\.(xmd|yls)$;Case 2
# (LOCAL SETTINGS\\Temp|Local Settings\\Temp|Local\\Temp)\\(word\.exe|winword\.exe);Case 2
#

# Ncat Example
# bin\\nc\.exe;Ncat Demo

# Regin
\\usbclass\.sys;File name known from REGIN malware
\\adpu160\.sys;File name known from REGIN malware
\\msrdc64\.dat;File name known from REGIN malware
\\msdcsvc\.dat;File name known from REGIN malware
\\config\\SystemAudit\.Evt;File name known from REGIN malware
\\config\\SecurityAudit\.Evt;File name known from REGIN malware
\\config\\SystemLog\.evt;File name known from REGIN malware
\\config\\ApplicationLog\.evt;File name known from REGIN malware
\\ime\\imesc5\\dicts\\pintlgbs\.imd;File name known from REGIN malware
\\ime\\imesc5\\dicts\\pintlgbp\.imd;File name known from REGIN malware
ystem32\\winhttpc\.dll;File name known from REGIN malware
ystem32\\wshnetc\.dll;File name known from REGIN malware
\\SysWow64\\wshnetc\.dll;File name known from REGIN malware
ystem32\\svcstat\.exe;File name known from REGIN malware
ystem32\\svcsstat\.exe;File name known from REGIN malware
IME\\IMESC5\\DICTS\\PINTLGBP\.IMD;File name known from REGIN malware
ystem32\\wsharp\.dll;File name known from REGIN malware
ystem32\\wshnetc\.dll;File name known from REGIN malware
pchealth\\helpctr\\Database\\cdata\.dat;File name known from REGIN malware
pchealth\\helpctr\\Database\\cdata\.edb;File name known from REGIN malware
Windows\\Panther\\setup\.etl\.000;File name known from REGIN malware
ystem32\\wbem\\repository\\INDEX2\.DATA;File name known from REGIN malware
ystem32\\wbem\\repository\\OBJECTS2\.DATA;File name known from REGIN malware
ystem32\\dnscache\.dat;File name known from REGIN malware
ystem32\\mregnx\.dat;File name known from REGIN malware
ystem32\\displn32\.dat;File name known from REGIN malware
ystem32\\dmdskwk\.dat;File name known from REGIN malware
ystem32\\nvwrsnu\.dat;File name known from REGIN malware
ystem32\\tapiscfg\.dat;File name known from REGIN malware
ystem32\\pciclass\.sys;File name known from REGIN malware
```

Package LOKI with a private Rule Set

LOKI can be packaged with a custom encrypted rule set, which is embedded in the pyinstaller package. In order to include your own rules, place them in a directory named private-signatures in the LOKI directory and execute *build.bat*

```
loki/
├── private-signatures/  <-- YARA rules places in here will by added to loki.exe
├── signature-base/      <-- clear text and still required (retrieved by loki-upgrader.exe)
|   ├── iocs/
|   ├── yara/
```

In order to successfully run the build script, you need to install PyInstaller. We use PyInstaller 2.1 due the problem that Packages built with PyInstaller 3 don't run on Windows 2003 and XP based systems. (yes, we need that in incident response - there are even productive systems out there running Windows 2000 or Windows NT).

The easiest way to do install PyInstaller is:

pip install pyinstaller==2.1

After that, you can just run the build script :

build.bat

You can verify whether the signature set is valid by calling loki-package-builder.py manually :

C:\Python27[-x64]\python.exe loki-package-builder.py --ruledir signatures --target rules

The usage of this tool is:

```
usage: loki-package-builder.py [-h] --ruledir RULEDIR --target TARGET

Package builder for Loki

optional arguments:
  -h, --help         show this help message and exit
  --ruledir RULEDIR  directory containing the rules to build into Loki
  --target TARGET    target where to store the compiled ruleset
```

Add Signature and IOCs

The IOC files for hashes and filenames are stored in the './signature-base/iocs' folder. All '.yar' files placed in the './signature-base/yara' folder will be initialized together with the rule set that is already included. Use the 'score' value to define the level of the message upon a signature match.

You can add hash, c2 and filename IOCs by adding files to the './signature-base/iocs' subfolder. All hash IOCs and filename IOC files must be in the format used by LOKI (see the default files). The files must have the strings "hash", "filename" or "c2" in their name to get pulled during initialization.

For Hash IOCs (divided by newline; hash type is detected automatically) :

Hash;Description [Reference]

For Filename IOCs (divided by newline) :

 # Description [Reference]

Regex;Score;False Positive Regex

You can user the following external variables in the YARA rules that your provide LOKI :

- filename - e.g. condition: $s1 and not filename == 'nmap.exe'
- filepatch - e.g. condition: filepath == 'C:\Windows\cmd.exe'
- extension - e.g. condition: uint32(0) == 0x5a4d and extension == ".txt"
- filetype - eg. condition: extension == ".txt" and filetype == "EXE"
- (see file-type-signatures.cfg in signature-base repo for all detected file types)
- md5 - legacy value

Threat Intel Receivers

Since version v0.10 LOKI includes various threat intel receivers using the public APIs of these services to retrieve and store the IOCs in a format that LOKI understands. It is no problem if these indicators overlap with the ones already included. Loki uses a filename regex or hash only once. (no performance impact).

The threat intel receivers have also been moved to the signature-base sub repository with version 0.15 and can be found in "./signature-base/threatintel".

Provide your API key via -k APIKEY or set it in the script header.

Open Threat Exchange (OTX) Receiver

It's a simple script that downloads your subscribed events/iocs from Alienvault OTX and stores them in the correct format in the './iocs' subfolder. The script is located in the "./threatintel" folder and is named "get-otx-iocs.py" :

```
usage: get-otx-iocs.py [-h] [-k APIKEY] [-o dir] [--verifycert] [--debug]

OTX IOC Receiver

optional arguments:
  -h, --help    show this help message and exit
  -k APIKEY     OTX API key
  -o dir        Output directory
  --verifycert  Verify the server certificate
  --debug       Debug output
```

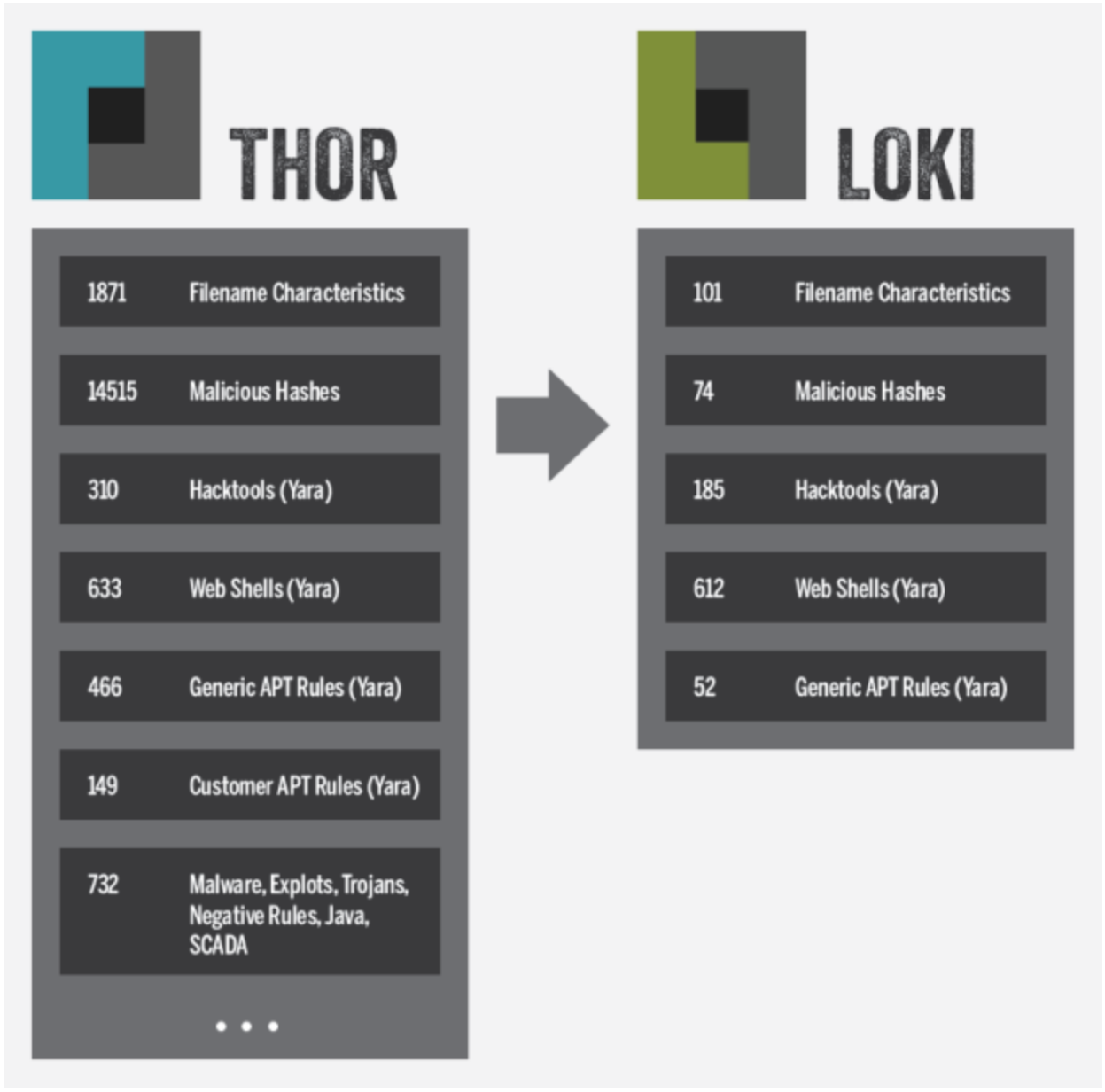
MISP Receiver

A simple script that downloads your subscribed events/iocs from a custom MISP instance and stores them in the correct format in the './iocs' subfolder. YARA rules stored in MISP will be written to the './iocs/yara' subfolder and automatically initialized during startup. The script is located in the "./threatintel" folder and is named "get-misp-iocs.py" :

```
usage: get-misp-iocs.py [-h] [-u URL] [-k APIKEY] [-l tframe] [-o dir]
                        [-y yara-dir] [--verifycert] [--debug]

MISP IOC Receiver

optional arguments:
  -h, --help    show this help message and exit
  -u URL        MISP URL
  -k APIKEY     MISP API key
  -l tframe     Time frame (e.g. 2d, 12h - default=30d)
  -o dir        Output directory
  -y yara-dir   YARA rule output directory
  --verifycert  Verify the server certificate
  --debug       Debug output
```

LOKI vs THOR

	LOKI	THOR
Type	Free / Open Source	Enterprise Product
Main Use Case	Preventive Scanning / Triage	Incident Response / Live Forensics
Platform	Windows (precompiled), Linux / macOS (source with dependencies)	Windows
Size (Binaries)	8 MB	16 MB
Language	Python	Python
Modules	3	26
Bundled Signatures	Open Source (~3000 YARA rules)	THOR's Signature Set (~9000 YARA rules)
Support and Testing	Github README & Issues, Travis-CI	Manual & Support Portal, Internal CI
Special Extras	Levenshtein check PESieve check Double Pulsar check	... a lot, see comparison


THOR
1871 Filename Characteristics
14515 Malicious Hashes
310 Hacktools (Yara)
633 Web Shells (Yara)
466 Generic APT Rules (Yara)
149 Customer APT Rules (Yara)
732 Malware, Explots, Trojans, Negative Rules, Java, SCADA
• • •
LOKI
101 Filename Characteristics
74 Malicious Hashes
185 Hacktools (Yara)
612 Web Shells (Yara)
52 Generic APT Rules (Yara)