

Threatray – Getting started

Session 1 – Intro and malware classification

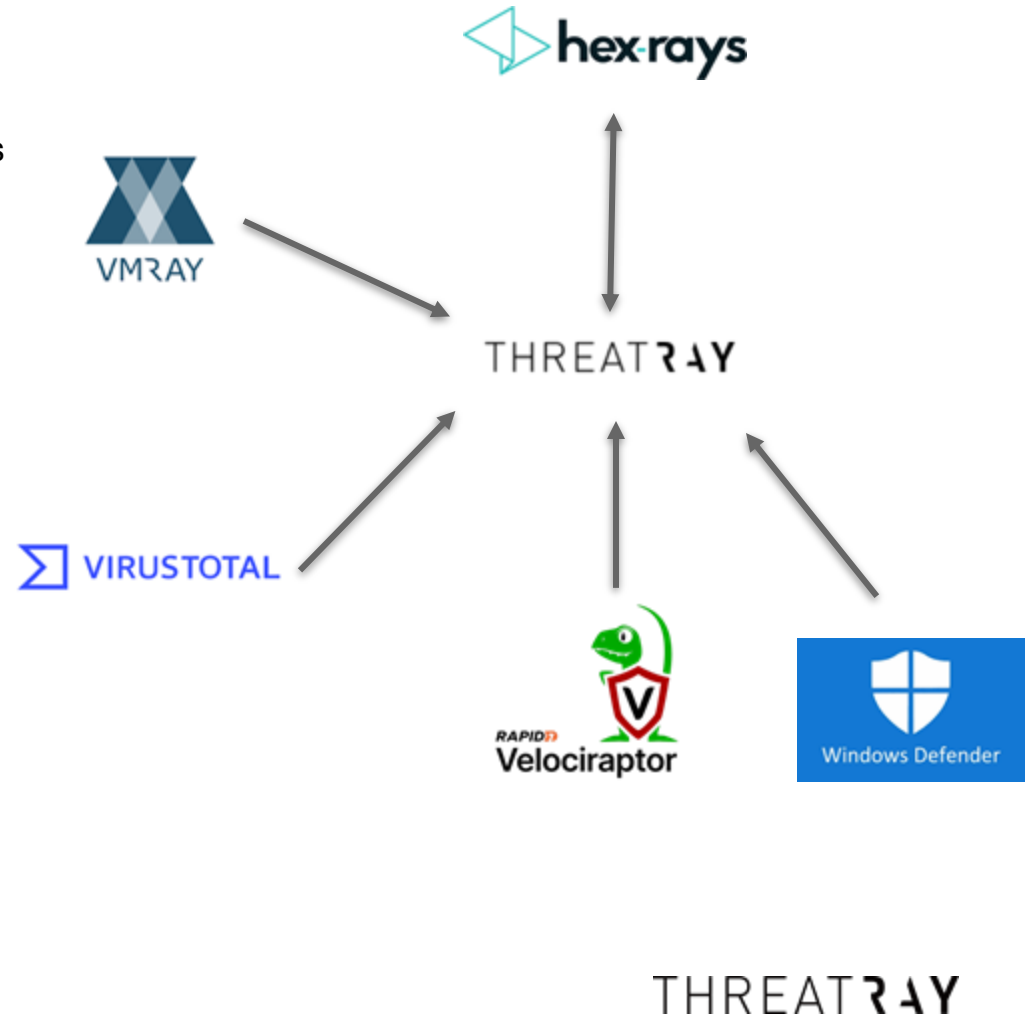
Content

- Threatray background
- File submission and the private organization repository
- Sandboxing view
- Binary intelligence view
- Threatray classification and detection verdict details
- Capability analysis

Threatray background

Threatray overview

- **Analysts Lack Deep-Dive Tooling.** Malware analysts defending against APTs and advanced threats are stuck with “surface-level” tools that remain slow, manual, and incomplete.
- **Code-First Analysis.** Threatray unlock the value of binary code deliver next gen malware defence capabilities beyond traditional hashes, metadata, and dynamic behaviour.
- **A novel type of product** dedicated to code similarity analysis that integrates with existing product stack
- **Truly novel capabilities.** Resilient classification, rapid variant discovery, and precise attribution — accelerating investigations that were previously impossible with existing tooling



The Core of Threatray's Technology

- **Code similarity analysis at scale and precision:** First platform to deliver on the promise of code similarity analysis (sometimes also called “code DNA analysis”).
- **Code search engine:** A first-of-its-kind engine that makes binary code instantly searchable at scale. That is, we can match unknown / suspicious code against our database of 100's of millions of malware and benign code fragments.
- **Proven ML Models:** Highly tuned, battle-tested machine learning models are at the core of our engine.



Unique industry-leading capabilities



Classify and understand unknown code fast

Leading malware family classification
— Accurately categorizes thousands of families across cybercrime, APTs, and C2 frameworks

Identifies new malware variants that bypass traditional solutions like YARA, AV, and VirusTotal

Industry-leading benign code classification

Code DNA for attribution



Hunting and intelligence

Code Pivoting – Instantly identify malware variants and clusters with point-and-click simplicity—no YARA rule writing needed

Traditional Pivoting – Seamlessly pivot on IPs, URLs, filenames, and other standard indicators

Accelerated IOC Discovery – Surface new indicators of compromise faster with intelligent correlation capabilities

Unified Intelligence Feeds – Integrate OSINT and raw malware feeds with your proprietary data and active cases in a single platform

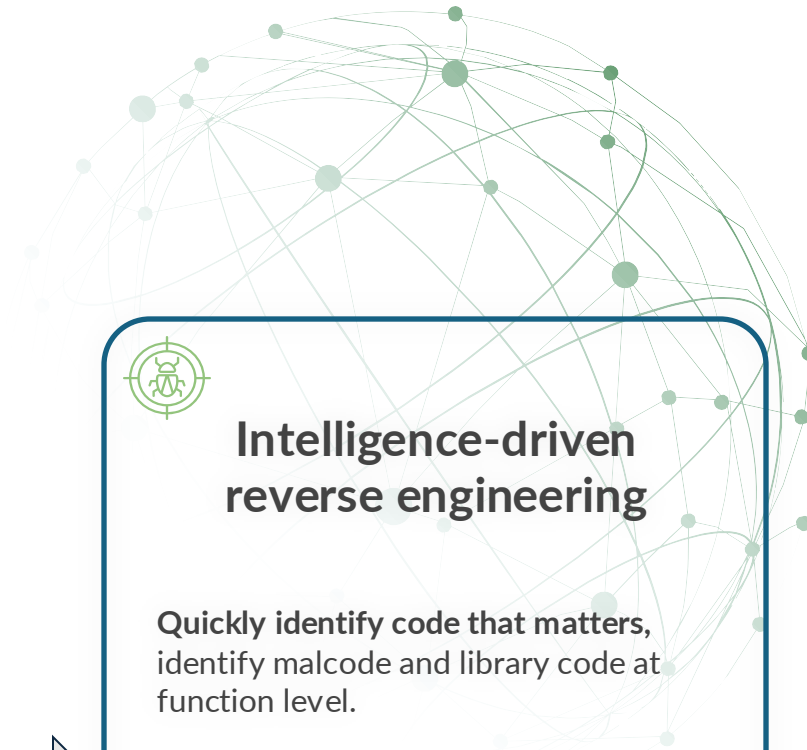


Intelligence-driven reverse engineering

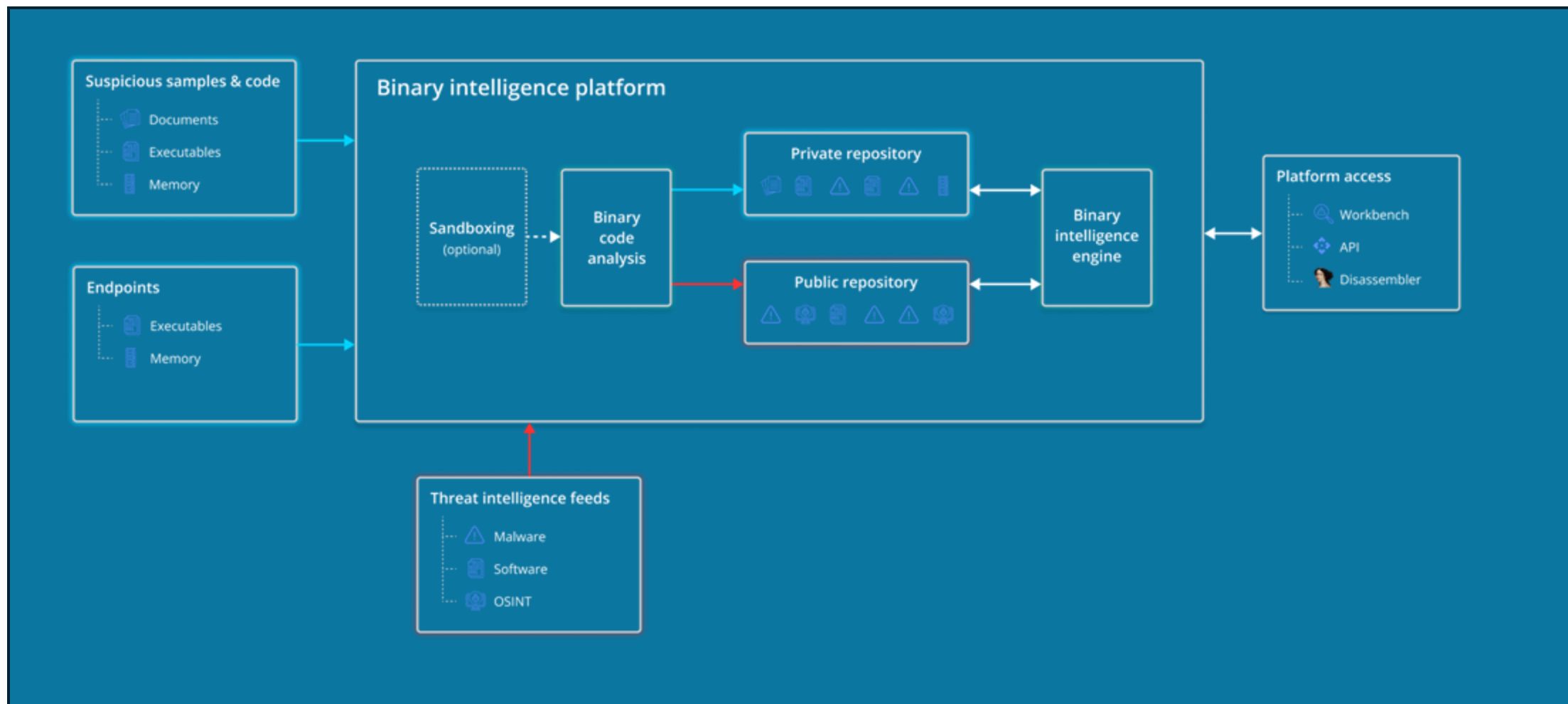
Quickly identify code that matters, identify malcode and library code at function level.

Streamline YARA rule development - Accelerate the creation and testing of precise YARA rules, saving significant analysis time.

Advanced code analysis and hunting — Use clustering and function-level analysis to identify shared code, track variants, and discover novel threats



Threatray architecture



File Submission and Organization Repository (Private per Customer)

File submission

Hash, Filename, IP address, ...

1 Submit File

File

8cb79686725831395879227658c0dd5f

Analysis mode

2 Dynamic **3** Static Minidump Endpoint Scan MANS file

Execute sample in an sandbox. Recommended option. Required for documents and scripts.

Analysis options

Environment

<input checked="" type="checkbox"/>	Windows 10	H2 1909, 64-bit
<input type="checkbox"/>	Windows 7	SP1, 64-bit
<input type="checkbox"/>	Windows 7	SP1, 32-bit

Selection of VM (custom images are available on demand)

Timeout in seconds*
180

☒ Enable network

☐ Submit as compound sample ⓘ

☐ Manually select DLL exports to call

Command line arguments

Label
POC-demd

Files can be labeled and then searched for Using label (e.g., case labeling)

Cancel Submit

Submission (1) is very similar to sandboxes

Normally use (2) dynamic analysis

- Especially for packed code (most of cyber crime malware)
- Scripts, documents etc.
- When you don't know

Unlike sandboxes Threatray can analyze any piece of code (not just executable code), like memory dumps, broken PE files etc. statically.

Use (3) static analysis for

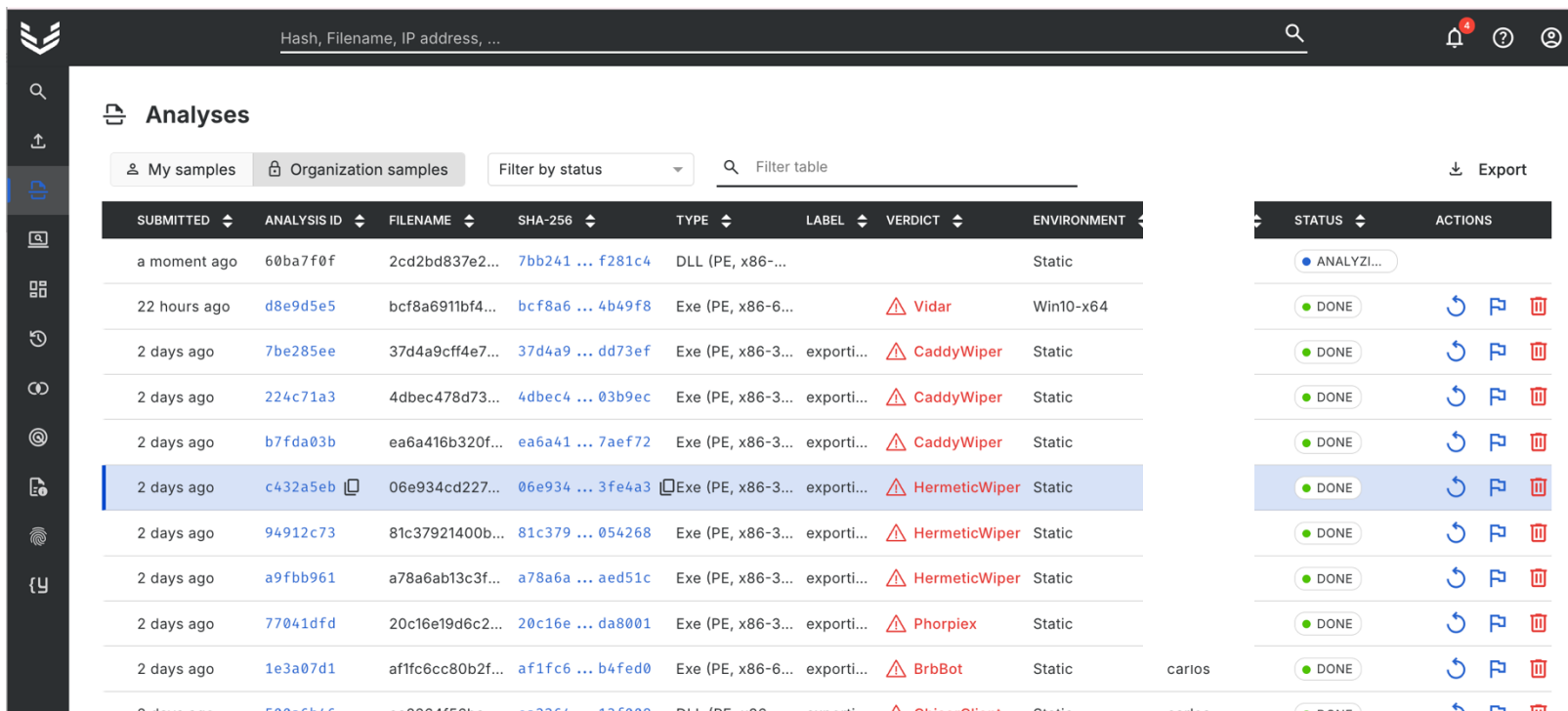
- Shellcode
- Raw memory dumps
- Non-packed code (often APTs)

Memory analysis: Threatray also supports the analysis of process mini-dumps and Trellix HX MANS files

For supported file types and more details check <https://docs.threatray.com/docs/submitting-files>

THREATRAY

Analyses – Your org’s private repo



Hash, Filename, IP address, ...

Analyses

My samples Organization samples Filter by status Filter table Export

SUBMITTED	ANALYSIS ID	FILENAME	SHA-256	TYPE	LABEL	VERDICT	ENVIRONMENT
a moment ago	60ba7f0f	2cd2bd837e2...	7bb241 ... f281c4	DLL (PE, x86-...			Static
22 hours ago	d8e9d5e5	bcf8a6911bf4...	bcf8a6 ... 4b49f8	Exe (PE, x86-6...		Vidar	Win10-x64
2 days ago	7be285ee	37d4a9c47e7...	37d4a9 ... dd73ef	Exe (PE, x86-3...	exporti...	CaddyWiper	Static
2 days ago	224c71a3	4dbec478d73...	4dbec4 ... 03b9ec	Exe (PE, x86-3...	exporti...	CaddyWiper	Static
2 days ago	b7fda03b	ea6a416b320f...	ea6a41 ... 7aef72	Exe (PE, x86-3...	exporti...	CaddyWiper	Static
2 days ago	c432a5eb	06e934cd227...	06e934 ... 3fe4a3	Exe (PE, x86-3...	exporti...	HermeticWiper	Static
2 days ago	94912c73	81c37921400b...	81c379 ... 054268	Exe (PE, x86-3...	exporti...	HermeticWiper	Static
2 days ago	a9fbb961	a78a6ab13c3f...	a78a6a ... aed51c	Exe (PE, x86-3...	exporti...	HermeticWiper	Static
2 days ago	77041dfd	20c16e19d6c2...	20c16e ... da8001	Exe (PE, x86-3...	exporti...	Phorpiex	Static
2 days ago	1e3a07d1	af1fc6cc80b2f...	af1fc6 ... b4fed0	Exe (PE, x86-6...	exporti...	BrbBot	Static
2 days ago	5005b4f6	ea2264f5e8a...	ea2264 ... 13f000	DLL (PE, x86-...	exporti...	ChiselClient	Static

STATUS	ACTIONS
ANALYZI...	
DONE	↺ 🚩 🗑
DONE	↺ 🚩 🗑
DONE	↺ 🚩 🗑
DONE	↺ 🚩 🗑
DONE	↺ 🚩 🗑
DONE	↺ 🚩 🗑
DONE	↺ 🚩 🗑
DONE	↺ 🚩 🗑
DONE	↺ 🚩 🗑
DONE	↺ 🚩 🗑

- The **Analyses view** (1) shows all the samples of an organization that have been submitted for analysis
- **“It is the view of the organization's strictly private malware repository”**
- The view contains samples that have been analyzed (status = DONE), samples that have submitted recently and are being analyzed (status = ANALYZING), etc.
- **We distinguish samples from analyses.** The same samples may be analyzed multiple times over time, resulting in multiple analyses of the same sample.
- We use SHA-256 hashes to uniquely identify samples, and we use the analysis-id to uniquely identify an analysis of a sample.

Sandboxing in Threatray

Sandboxing in Threatray

- **Sandboxing** = dynamic / behavior
- **Threatray** = code / static + intelligence
- **Threatray sandbox**
 - State of the art
 - Excellent detonation rate and anti-anti-analysis
 - No-frills, basic sandboxing information (processes, command line arguments, behavior artifacts, e.g., domains, IPs, URLs, mutexes etc.)
- **Threatray contains a sandboxing component** for two reasons
 - maximize code collection (unpacking, downloading stages) for Threatray deep code analysis, “the more code the better”
 - collection of behavior artifacts for our intelligence capabilities
- **We don’t use any behavior heuristics for detection.** (all classification and detection is coming from Threatray’s unique code detection engine)

Sandboxing

The screenshot displays the Threatray sandboxing interface. At the top, a header bar shows the file hash `eeee4d6d6f4982cc730304c1c42506dcde8555898526e5ea254e789723e678d2` and a search icon. Below the header, a sidebar on the left contains navigation icons. The main content area is divided into two tabs: "BINARY INTELLIGENCE" and "BEHAVIOR", with the latter being active and marked with a red "1".

The "BEHAVIOR" tab displays a "Process graph" showing the execution flow of the malware. The graph starts with a process labeled `a7fd08981ce2130a93ede12f8eb76ae8.exe` (PID #1 [5400]), which is terminated. This process spawns `cvtres.exe` (PID #2 [5964]), which is also terminated. `cvtres.exe` then spawns `svchost.exe` (PID #3 [988]), which is terminated. `svchost.exe` further spawns `rundll32.exe` (PID #7 [5904]), which is terminated. `rundll32.exe` then spawns `wmiprvse.exe` (PID #8 [2900]), which is terminated. `wmiprvse.exe` then spawns `tr` (PID #1 [5400]), which is terminated. `tr` then spawns `tiworker.exe` (PID #12 [1624]), which is terminated. The graph also shows `svchost.exe` (PID #4 [948]) spawning `wmiprvse.exe` (PID #5 [4540]) and `rundll32.exe` (PID #10 [3688]), both of which are active. The `rundll32.exe` (PID #10) is also associated with the `Latrodectus` malware family.

Below the process graph, the "Dynamic artifacts" section is visible, showing a list of artifacts categorized by type: IPS (4), DOMAINS (10), URLs (12), FILES (5), REGISTRY (6), and MUTEXES (3). A search bar is provided to filter entries.

- Go to “behavior view” (1) to see sandboxing information
- The behavior reports contains the following elements.
- **Header** summarizes verdict (classification, detection), 3rd party YARA classification, and meta information
- **Process graph** showing which processes running, Threatray classification / detection per process
- **Dynamic artifacts** =IPs, domains, URLs, files, registry, mutexes
- **Dynamic capabilities** shows malware capabilities like obtained from behaviour logs.
- **Process tree with command line arguments.**

Sandboxing – Header

Meta information

Warning: unc_loader_078, unc_loader_062, LummaStealer, Latrodectus

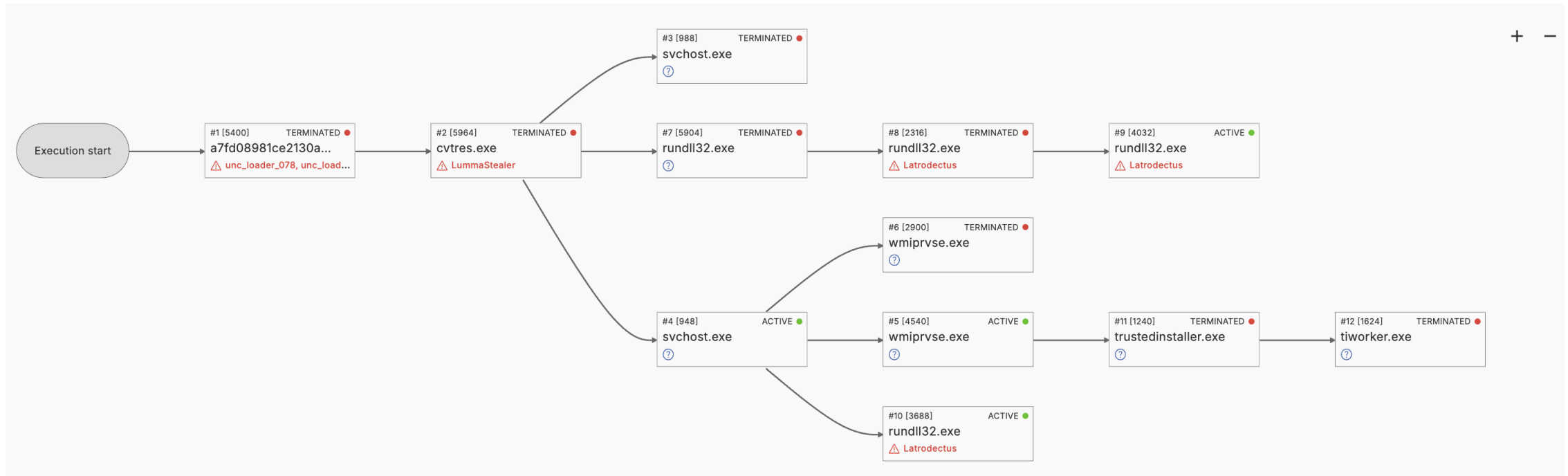
File Details: eeea4d6d6f4982cc730304c1c42506dcde8555898526e5ea254e789723e678d2 | a7fd08981ce2130a93ede12f8eb76ae8.exe | Exe (PE, x86-32) | 9.92 MB | 2025-03-15

Detection		Intelligence		Analysis	
CODE DETECTION	unc_loader_062, unc_loader_0...	COMMUNITY YARA	CAPE_Icedsleep, CAPE_Latroductus_AES, CAPE_Lumma_1, ELASTIC_Windows_Trojan_Donutloader_F40E3759, ELASTIC_Windows_Trojan_Latroductus_841Ff697, ELASTIC_Windows_Trojan_Lumma_4Ad749B0, SEKOIA_Latroductus_Exports, win_latrodectus_g0, win_lumma_a0	Analysis ID	1a3ee1f5
Av	malicious			Analysis created	2025-03-15 09:23:12
				Analysis type	Dynamic on Win10-x64
				Analysis timeout	300s, terminated after 300s

Classification and detection information

3rd party YARA rules from community and Nextron

Sandboxing – Process Graph



- Threatray tries to attribute the code found within each process to malware (or "goodware" / benign families).
- Question mark (?) means, we could not find any code in that process which is attributable by Threatray
- We see that Threatray's is tracking all the components of an infection chain (loader, different malware stages)
 - Loader > Latrodectus > LummaStealer

Sandboxing – Dynamic Artifacts

Dynamic artifacts ⓘ

IPS 4 DOMAINS 10 URLs 12 FILES 5 REGISTRY 6 MUTEXES 3

🔍 Filter entries

104.21.21.170
162.125.66.18
172.67.199.164
45.93.20.42

Dynamic artifacts ⓘ

IPS 4 DOMAINS 10 URLs 12 FILES 5 REGISTRY 6 MUTEXES 3

🔍 Filter entries

assets.dropbox.com	status.dropbox.com
cfl.dropboxstatic.com	www-env.dropbox-dns.com
elaviltabs.run 🔍	www.dropbox.com
forums.dropbox.com	
hingehjan.shop	
horetimodual.com	
remustarofilac.com	

Dynamic artifacts ⓘ

IPS 4 DOMAINS 10 URLs 12 FILES 5 REGISTRY 6 MUTEXES 3

🔍 Filter table

URL ⌵

https://assets.dropbox.com/www/en-us/illustrations/spot/target-miss.svg
https://cfl.dropboxstatic.com/static/images/favicon.ico
https://cfl.dropboxstatic.com/static/metaserver/static/css/error.css
https://elaviltabs.run/utcgiv.dll
https://forums.dropbox.com

Network artifacts

Sandboxing – Dynamic Artifacts

Dynamic artifacts ⓘ



IPS 4 DOMAINS 10 URLs 12 FILES 5 REGISTRY 6 MUTEXES 3

Filter table

Export

FILENAME	OPERATIONS	OPERATIONS DETAILS
:wtfbqq	ACDW	Data written (SHA256): 5e9b53207cb53c38217fb443e9a84c0fa745fa7fc62ace3673a2c49c6e873749
\\?\c:\users\<USERNAME>\appdata\roaming\custom_update\update_80e4a823.dll	ACDW	Data written (SHA256): 5e9b53207cb53c38217fb443e9a84c0fa745fa7fc62ace3673a2c49c6e873749
c:\users\<USERNAME>\appdata\local\temp\5e66odcr5byn6xdaz5zbbcvgb1712r.dll	ACDW	Data written (SHA256): 5e9b53207cb53c38217fb443e9a84c0fa745fa7fc62ace3673a2c49c6e873749
c:\users\<USERNAME>\appdata\roaming\custom_update\update_80e4a823.dll	ACDW	Data written (SHA256): 5e9b53207cb53c38217fb443e9a84c0fa745fa7fc62ace3673a2c49c6e873749
c:\users\<USERNAME>\desktop\af7d08981ce2130a93ede12f8eb76ae8.exe	A	Data written (SHA256): eeea4d6d6f4982cc730304c1c42506dcde8555898526e5ea254e789723e678d2

Items per page: 10

1 – 5 of 5

A: Access C: Create W: Write D: Delete

Host artifacts

Dynamic artifacts ⓘ

IPS 4 DOMAINS 10 URLs 12 FILES 5 REGISTRY 6 MUTEXES 3

Filter table

REGISTRY KEY	OPERATIONS	OPERATIONS DETAILS
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\NETFramework\AppContext	A	
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion	A	
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\DisplayVersion	A	
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\ReleaseId	A	
HKEY_USERS\S-1-5-21-1950431387-398766261-2715397724-1000\Software\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders	A	
HKEY_USERS\S-1-5-21-1950431387-398766261-2715397724-1000\Software\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders\AppData	A	

Items per page: 10

A: Access C: Create W: Write D: Delete

Dynamic artifacts ⓘ

IPS 4 DOMAINS 10 URLs 12 FILES 5 REGISTRY 6 MUTEXES 3

Filter entries

Gfjxuuzoskr
\\?\App1FEF7AA4D7E79638
running

Sandboxing – Process tree and command lines

Process tree ⓘ

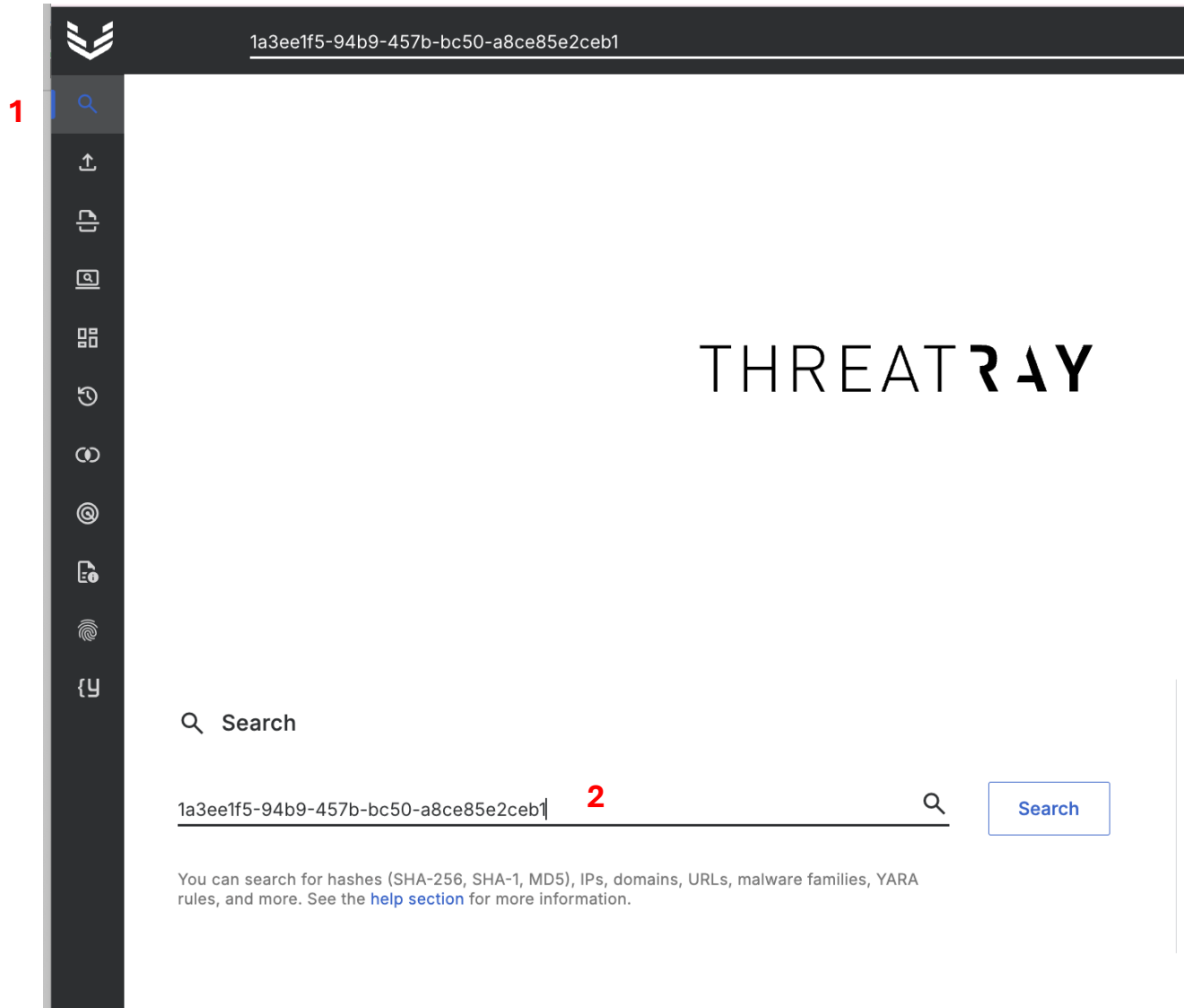
#1	a7fd08981ce2130a...	[5400]	●	"c:\users\<USERNAME>\desktop\a7fd08981ce2130a93ede12f8eb76ae8.exe"	⚠ unc_loader_078, unc_loader_0...	<div></div>
#2	cvtres.exe	[5964]	●	"c:\windows\microsoft.net\framework\v4.0.30319\cvtres.exe"	⚠ LummaStealer	<div></div>
#3	svchost.exe	[988]	●	c:\windows\system32\svchost.exe -k localservicenetworkrestricted -p	?	<div></div>
#4	svchost.exe	[948]	●	c:\windows\system32\svchost.exe -k netsvcs -p	?	<div></div>
#5	wmiprvse.exe	[4540]	●	c:\windows\system32\wbem\wmiprvse.exe -secured -embedding	?	<div></div>
#11	trustedinstaller.exe	[1240]	●	c:\windows\servicing\trustedinstaller.exe	?	<div></div>
#12	tiworker.exe	[1624]	●	c:\windows\winsxs\amd64_microsoft-windows-servicingstack_31bf3856ad364e35_10.0.18362.411_none...	?	<div></div>
#6	wmiprvse.exe	[2900]	●	c:\windows\system32\wbem\wmiprvse.exe -embedding	?	<div></div>
#10	rundll32.exe	[3688]	●	rundll32.exe "c:\users\<USERNAME>\appdata\roaming\custom_update\update_80e4a823.dll", editor	⚠ Latrodectus	<div></div>

Dynamic capabilities

Dynamic Capabilities ⓘ

Filter by category ▾		x cvtres.exe [5964] x ▾		Created: 2025-10-12 18:19:25 CAPA Version: v9.2.1	
reference WMI statements cvtres.exe [5964]	COLLECTION	capture screenshot cvtres.exe [5964]	COLLECTION	send data cvtres.exe [5964]	COMMUNICATION
initialize WinHTTP library cvtres.exe [5964]	COMMUNICATION	reference HTTP User-Agent string cvtres.exe [5964]	COMMUNICATION	prepare HTTP request cvtres.exe [5964]	COMMUNICATION
send HTTP request cvtres.exe [5964]	COMMUNICATION	initialize Winsock library cvtres.exe [5964]	COMMUNICATION	encrypt data using DPAPI cvtres.exe [5964]	DATA-MANIPULATION
read clipboard data cvtres.exe [5964]	HOST-INTERACTION	get common file path rundll32.exe [2316], rundll32.exe [3688], rundll32.exe [4032], a7fd08981ce2130a93ede12f8eb76ae8.exe [5400], rundll32.exe [5904], cvtres.exe [5964]	HOST-INTERACTION	get file attributes rundll32.exe [2316], rundll32.exe [3688], rundll32.exe [4032], rundll32.exe [5904], cvtres.exe [5964]	HOST-INTERACTION
get disk information rundll32.exe [2316], rundll32.exe [4032], cvtres.exe [5964]	HOST-INTERACTION	create or open mutex on Windows rundll32.exe [2316], rundll32.exe [3688], rundll32.exe [4032], a7fd08981ce2130a93ede12f8eb76ae8.exe [5400], cvtres.exe [5964]	HOST-INTERACTION	get hostname cvtres.exe [5964]	HOST-INTERACTION
					write and execute a file cvtres.exe [5964]
					receive HTTP response cvtres.exe [5964]
					open clipboard cvtres.exe [5964]
					write file on Windows rundll32.exe [2316], cvtres.exe [5964]
					get system information on Windows rundll32.exe [2316], rundll32.exe [3688], rundll32.exe [4032], a7fd08981ce2130a93ede12f8eb76ae8.exe [5400], rundll32.exe [5904], cvtres.exe [5964]

How to replay this analysis in your Threatray instance



- Go to the search view (1) and paste the unique analysis ID (2) into the search bar

The unique analysis ID for the Latrodectus / LummaStealer infection chain is

1a3ee1f5-94b9-457b-bc50-a8ce85e2ceb1

- The analysis is in the global Threatray repository and accessible to all our customers (see next session).

THREATRAY

The binary intelligence view

The binary intelligence view

- **The binary intelligence view contains many of Threatray's unique and advanced code analysis capabilities about unknown / suspicious code.**


Binary intelligence view is complementary and not replacing the behavior / sandbox view. Think of behavior view and binary intelligence view as “ying & yang” of malware analysis.

- **The focus is on unknown / suspicious “code blocks” found during an analysis.**


A code blocks can be **binary files on disk** or **code that we find in memory** (e.g., from unpacking, malware stages that are downloaded from the Internet, dropped components etc.)

- **For each code block Threatray shows the same type of information and provides the same type of analysis capabilities.**



Binary intelligence view



unc_loader_078
unc_loader_062
LummaStealer
Latrodectus

eeee4d6d6f4982cc730304c1c42506dcde8555898526e5ea254e789723e678d2  Global

a7fd08981ce2130a93ede12f8eb76ae8.exe | Exe (PE, x86-32) | 9.92 MB | 2025-03-15


Detection ⓘ

CODE DETECTION unc_loader_062, unc_loader_0...

AV malicious

Intelligence ⓘ

COMMUNITY YARA CAPE_Icidsleep [+8 more](#)

Analysis ID 1a3ee1f5 

Analysis created 2025-03-15 09:23:12


Analysis type Dynamic on Win10-x64

Analysis timeout 300s, terminated after 300s

BINARY INTELLIGENCE 1 BEHAVIOR

Submitted file


>
a7fd08981ce2130a93ede12f8eb76ae8.exe

Exe (PE, x86-32) 16226 functions 


Process and memory details ⓘ

a7fd08981ce2130a... [5400] "c:\users\<USERNAME>\desktop\a7fd08981ce2130a93ede12f8eb76ae8.exe"


>
c:\users\...\a7fd08981ce2130a93ede12f8eb76ae8.exe

0x290000
Size: 0x9fa000
Exe (PE, x86-32)
16226 functions



>
Memory region

0x2d80000
Size: 0x14f000
data
No functions



>
Memory region



0x2d824d6
Size: 0x14cb2a
Exe (PE, x86-32)
418 functions
 unc_loader_062



>
Memory region



0x3460000
Size: 0x1000
data
No functions




▼
Memory region



0x37164c0
Size: 0x7
data
No functions


  22%

VERDICT DETAILS
CODE INTELLIGENCE
OSINT HUNT ⓘ
SIMILAR SAMPLES ⓘ
CAPABILITIES ⓘ
FILE INFORMATION
STRINGS

Detection ⓘ

CODE DETECTION —

YARA —

PRIVATE YARA —



AV clean

Intelligence ⓘ

NEXTRON THOR YARA —

COMMUNITY YARA —

PRIVATE YARA —

- **Go to “binary intelligence view” (1)**
- **The header** is the same as in the behavior / sandbox view – after all it is the same analysis, just a different view.
- **The submitted file (2)** is analyzed statically (i.e., without sandboxing).
- **The process and memory detail view (3)** shows all processes and their virtual memory contents, whenever there is unknown / suspicious code in process memory.
- **The static file and each memory of file element in a process are “code blocks”** (green dashed boxes)

Same analysis capabilities for every code block

rundll32.exe [3688] rundll32.exe "c:\users\<USERNAME>\appdata\roaming\custom_update\update_80e4a823.dll", editor

> c:\windows\system32\rundll32.exe 0x7ff61f5b0000 Size: 0x17000 Exe (PE, x86-64) 113 functions ?

▼ Memory region | 0x1dc1bd4 Size: 0x1caf42c DLL (PE, x86-64) 162 functions ⚠ Latrodectus 29%

VERDICT DETAILS CODE INTELLIGENCE FUNCTIONS OSINT HUNT 163 SIMILAR SAMPLES 10+ CAPABILITIES 8 FILE INFORMATION STRINGS

Detection ⓘ

CODE DETECTION	⚠ Latrodectus	42 functions	29%
YARA	—		
PRIVATE YARA	—		
Av	clean		

Intelligence ⓘ

NEXTRON THOR YARA	—
COMMUNITY YARA	CAPE_Icedsleep CAPE_Latroductus_AES ELASTIC_Windows_Trojan_Latroductus_841Ff697 SEKIOIA_Latroductus_Exports win_latrodectus_g0
PRIVATE YARA	—

- Code block is green dashed box (in this example it is code found in process memory)
- Analysis capabilities for this code block are show with light blue background

Threatray classification & detection verdicts, and 3rd party YARA

Detection / classification verdict

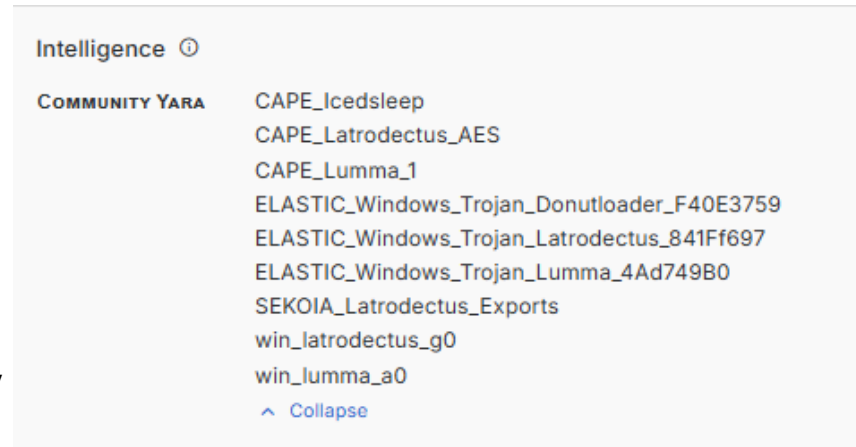


unc_loader_078
unc_loader_062
LummaStealer
Latrodectus

- **A Threatray detection and classification verdict is computed from multiple sources:**
 - Threatray code detection technology (based on code similarity)
 - An OEM antivirus engine
 - YARA rules curated by Threatray
 - YARA rules curated by users (optional)
- **Almost all family identification / classification is coming from Threatray's code detection technology.**
- **For family naming we follow the Malpedia naming convention** (<https://malpedia.caad.fkie.fraunhofer.de/families>) wherever possible. Yet, we track more and different families than Malpedia, and therefore not all our family names can be tracked back to Malpedia.

3rd party YARA matches

- We use 3rd party YARA rules and scan every code block (i.e., memory and files) with those rules
- The 3rd party rules consist of open source YARA rules, which we curate.
- We are also using NEXTRON YARA rule set, through our technology partnership with NEXTRON



3rd party YARA matches

- The **Threatray verdict** (classification, detection) and **YARA matches** are shown in **multiple locations of our analysis reports.**
- **Where are verdicts and YARA shown?**
 - In the header, where we aggregate verdicts and YARA
 - For every code block, i.e., static file and in process memory
- **Verdict and YARA matches have always have the same structure and contain the same type of information (see next slide)**

Verdict and YARA – Example in process memory

Go to “verdict details” to understand why is Latrodectus

We classify the code block as Latrodectus

Memory region 0x1dc1bd4 Size: 0x1caf42c DLL (PE, x86-64) 162 functions Latrodectus 29%

VERDICT DETAILS CODE INTELLIGENCE FUNCTIONS OSINT HUNT 163 SIMILAR SAMPLES 10+ CAPABILITIES 8 FILE INFORMATION STRINGS

Detection ⓘ

1	CODE DETECTION	Latrodectus	42 functions	29%
2	YARA	—		
3	PRIVATE YARA	—		
4	Av	clean		

5 Intelligence ⓘ

NEXTRON THOR YARA —

COMMUNITY YARA

- CAPE_Icedsleep
- CAPE_Latrodectus_AES
- ELASTIC_Windows_Trojan_Latrodectus_841Ff697
- SEKOIA_Latrodectus_Exports
- win_latrodectus_g0

PRIVATE YARA —

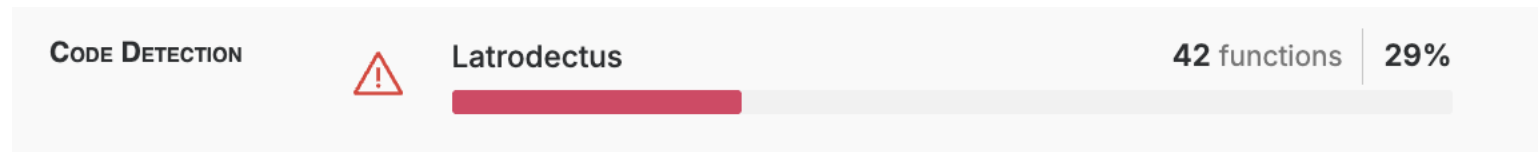
- (1) Threatray code detection identifies Latrodectus
- (2) No detection from Threatray curated YARA rules
- (3) No detection from customer supplies private YARA rules
- (4) We use an OEM antivirus (AV) which has no detection in this case

- (5) We are using curated open-source community YARA rules and are also using the industry leading NEXTRON YARA rules (“3rd party YARA rules”)

In this example we see some community rule matches.

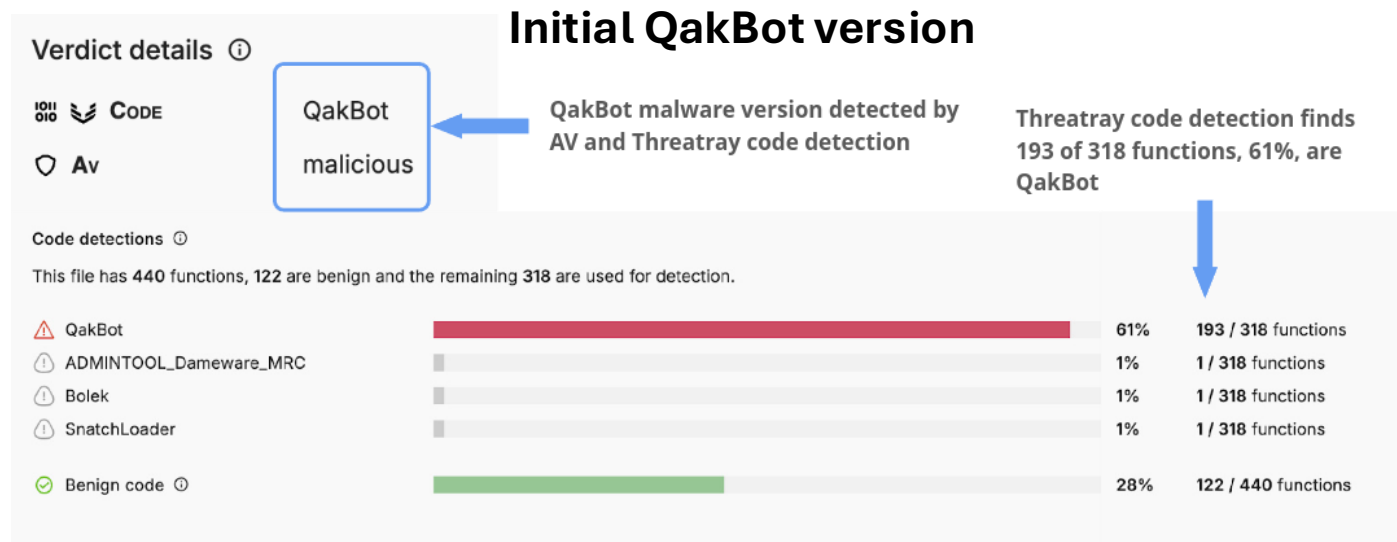
Threatray code detection – how does it work?

- **Code detection is our industry wide unique malware detection and classification technology.** It is based on measuring code similarity of a piece of unknown code block against our huge database of malware code.
- Here we detect Latrodectus because 29% of the unknown code is very similar to Latrodectus code from our malware code database

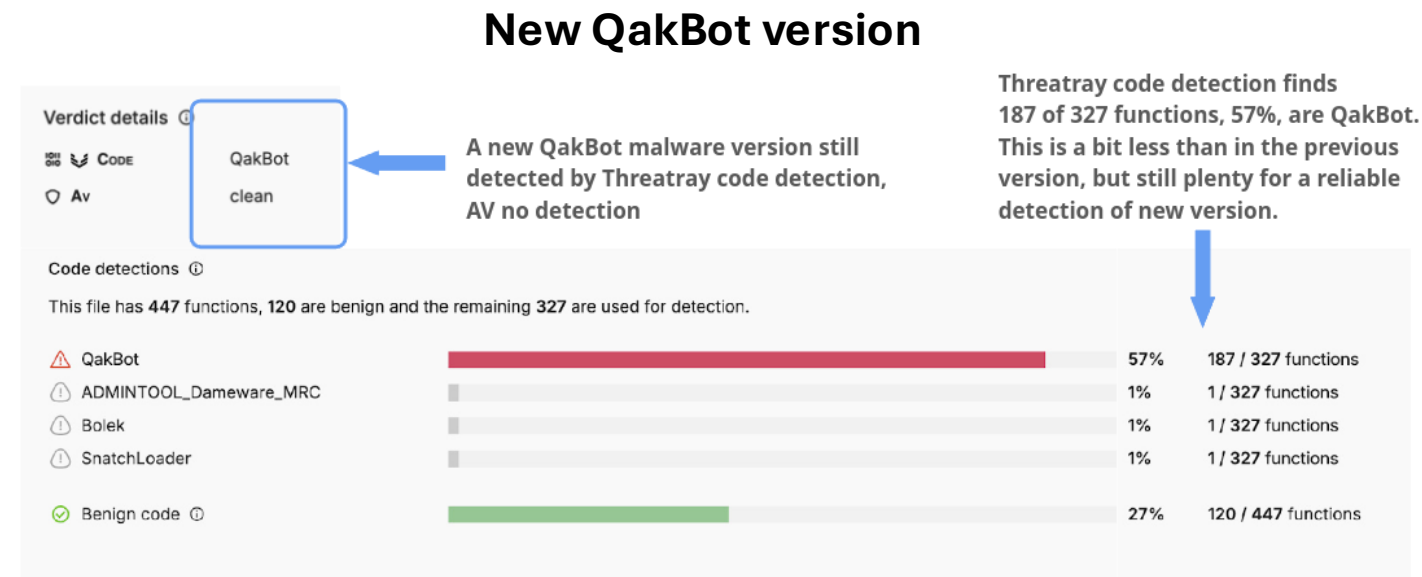


- **Code detections have key advantages for Threatray users:**
 - Very resilient to new malware variants => Threatray detects new variants, YARA, AV, sandboxes miss
 - We can track more malware families than others, since it is very easy for us to track new malware families
 - We can easily and very reliably identify non-malicious codes and tooling like LOLBins etc.

Threatray code detection resilience against new malware variants



- New version of QakBot detected only by Threatray, missed by AV and YARA
- Threatray code detection is resilient against new malware variants
- The reason is that in newer malware variants, the amount of known malicious code is decreasing — yet we can still measure it.



NK APT example only caught by Threatray

1

Appleseed

83d157f5d66a87666749b7795678e3b078518228580f56ce2519fdcf160542b7 | DLL (PE, x86-64) | 404 KB | 2025-03-24

Detection ⓘ	Intelligence ⓘ	Analysis ID	7f795fc6 ⓘ
CODE DETECTION	No intelligence found	Analysis created	2025-03-24 15:05:04
Av		Analysis type	Dynamic on Win10-x64
		Analysis timeout	180s, terminated after 180s

BINARY INTELLIGENCE | BEHAVIOR

Submitted File

Show all | ⚠ Malicious only | ⓘ Intelligence only

83d157f5d66a87666749b7795678e3b078518228580f56ce2519fdcf160542b7 | DLL (PE, x86-64) | 704 functions | ⚠ Appleseed | 25% ⓘ

VERDICT DETAILS | CODE INTELLIGENCE | FUNCTIONS | OSINT HUNT ⓘ | SIMILAR SAMPLES ⓘ | CAPABILITIES ⓘ | FILE INFORMATION | STRINGS | ⬇ | ⓘ

Detection ⓘ	Intelligence ⓘ
CODE DETECTION	NEXTRON THOR YARA -
YARA	COMMUNITY YARA -
PRIVATE YARA	PRIVATE YARA -
Av	

2

24 functions | 25%

clean

- This is an example of North Korean APT sample from the Appleseed family.
- It is a new version of Appleseed, we have received the sample in spring 2025 and were the only technology able to classify the sample (1)
- The reason is our code detection technology, which has found 25% code reuse (2) between this new unknown sample and previous Appleseed sample which is 4 years old.

NK APT example only caught by Threatray

- **Traditional sandboxes fail** to classify new malware variants because of (1) brittle signature tech (2) lack of dedicated APT classification capabilities, leaving critical threats unidentified.
- We only see OEM AV detections which are part of most sandboxes, but no classification.
- See <https://bazaar.abuse.ch/sample/83d157f5d66a87666749b7795678e3b078518228580f56ce2519fdcf160542b7/>

The screenshot shows the MALWARE bazaar interface. At the top, there's a navigation bar with links like 'Browse', 'Upload', 'Hunting Alerts', 'Access Data', 'FAQ', 'About', and 'Logout'. Below this, the main content area displays analysis results for a sample. The sample name is 'ANY.RUN'. The analysis details include: Malware family: n/a, ID: 1, File name: 83d157f5d66a87666749b7795678e3b078518228580f56ce2519fdcf160542b7, Verdict: No threats detected (in a green box), Analysis date: 2025-04-23 13:57:27 UTC, Tags: n/a, and Link: https://app.any.run/tasks/6d4e9c90-09c1-4f13-9016-b96bbac7c407. A note states: 'ANY.RUN is an interactive sandbox that analyzes all user actions rather than an uploaded sample'. Below this, a list of sandboxes and their results is shown: CAPE Sandbox, CyberFortress (Malicious), Dr. Web vsCube (Clean), FileScan.IO (Likely Malicious), Hybrid Analysis (TropenAgent), Intezer (Malicious), Joe Sandbox (malicious), Nucleon Malprob (Malware), CERT.PL MWDB, Spamhaus Hash Blocklist (Suspicious file), Threatray (Timouky), Hatching Triage (Suspicious), and Threat.Zone (Malicious).

Result

Threat name: n/a
Detection: malicious
Classification: n/a
Score: 48 / 100
Link: https://www.joesandbox.com/analysis/1672126

Signature

Multi AV Scanner detection for submitted file

Tracking LOLBins and admin tools

The screenshot displays the Threatray interface for a file analysis. The file name is **LokiPasswordStealer(P... ADMINTOOL_powerrun**. The analysis ID is **994eb71f**. The file is identified as **Exe (PE, x86-32)** with a size of **687.61 KB** and a date of **2025-10-07**. The detection is **ADMINTOOL_powerrun, LokiBot**. The behavior is **Loki** and the Av signature is **malicious**. The intelligence section shows **NEXTRON THOR YARA** and **SUSP_Autolt_Opcodes_Nov20**. The analysis was created on **2025-10-08** and is a **Dynamic on V** analysis with a timeout of **180s**.

The **Process Graph** shows the execution flow:

```
graph LR; Start([Execution start]) --> P1["#1 [3988] TERMINATED  
loki_3d873193b3f6..."]; P1 --> P2["#2 [1476] TERMINATED  
svchost.exe"]; P1 --> P3["#3 [2564] TERMINATED  
explorer.exe"]; P1 --> P4["#4 [2120] TERMINATED  
kfb.exe  
ADMINTOOL_powerrun"]; P4 --> P5["#5 [5384] TERMINATED  
kfb.exe  
ADMINTOOL_powerrun"]; P5 --> P7["#7 [3936] ACTIVE  
regsvcs.exe  
LokiPasswordStealer(PWS)"]; P3 --> P6["#6 [3664] TERMINATED  
securityhealthservi..."];
```

- This is an example where an LOLBin / admin tool is used as part of the infection chain
- We see that the attacker first deploys the Powerrun admin tool (1), and then LokiPasswordStealer malware (2).
- As you can see, we prefix LOLBins / admin tools with “ADMINTOOL_...” in the verdict.

Our 3rd party YARA rules are NOT used for the classification / detection verdict

Hash, Filename, IP address, ...

ad0dfd71defd7fd5f1dd7aa5a421f10ec1533cc4c7e8ee13c952027426cc80b8

7b3404219a39fea0af903c69ffb55e6b.exe | Exe (PE, x86-64) | 120 KB | 2025-10-10

1

Detection ⓘ	Intelligence ⓘ	Analysis ID	c8e3b1a0
Av	NEXTRON THOR YARA SUSP_PowerShell_Agent_Char...	Analysis created	2025-10-10 10:46:59
clean	SUSP_Stage1_Indicators_Jul20_1	Analysis type	Dynamic on Win10-x64
	^ Collapse	Analysis timeout	300s, terminated after 300s
		Label	mb_unknown

- **Our 3rd party YARA is treated as intelligence and NOT used for classification and detection**
- **We strongly advise to look at our 3rd party YARA matches** however, we just don't control those rules and therefore do not use them for detection and classification.
- This example shows a sample that is neither detected nor classified by Threatray, hence the blue question mark (1).

However, we see matches by NEXTRON Thor (2), which are very likely correct, since high-quality rules set.

How to replay your Threatray instance

Here are the analysis IDs for the examples shown:

- For the Latrodectus example, binary intelligence view etc. we have used 1a3ee1f5-94b9-457b-bc50-a8ce85e2ceb1
- For the sample with THOR YARA match but no Threatray detection: c8e3b1a0-2680-4492-a74d-374c27bb5d3f
- For the Appleseed APT example, we have used analysis 7f795fc6-0892-40d5-a5c2-5e1e38528b0e
- For the LOLbin / admin tool example we have used 994eb71f-c4c6-4c9f-b726-be6b9ab47332

Capability analysis

Code capabilities

- **For every code block we extract capabilities of that code block using static analysis.**
- We use the Mandiant CAPA framework for doing so.
- This (static) code-based approach reveals capabilities that are present in code but not present in behavior capabilities.
- Static capabilities are especially useful when analysing samples without any classification information, as they greatly assist in the triage process.

Code capabilities

- This is an example showing a TunnelSpectre sample, a Chinese APT
- It is a DLL that we have analyzed statically only (without sandboxing)
- The capabilities show that this sample features anti-analysis, and host reconnaissance and command execution capabilities.

22d556db39bde212e6dbaa154e9bcf57527e7f51fa2f8f7a60f6d7109b94048e

22d556db39bde212e6dbaa154e9bcf57527... | DLL (PE, x86-32) | 76 KB | 2024-05-27

TunnelSpectre

Detection: CODE DETECTION: TunnelSpectre_Payload, Av: malicious, Intelligence: NEXTRON THOR YARA, MAL_TunnelSpectre_May24

Analysis ID: 38245fc0, Analysis created: 2025-10-12 21:07:57, Analysis type: Static

BINARY INTELLIGENCE

Submitted File [Show all] [Malicious only] [Intelligence only]

22d556db39bde212e6dbaa154e9bcf57527e7f51fa2f8f7a60f6d7109b94048e | DLL (PE, x86-32) | 251 functions | TunnelSpectre_Payload | 53%

VERDICT DETAILS | CODE INTELLIGENCE | FUNCTIONS | OSINT HUNT | SIMILAR SAMPLES | **CAPABILITIES** | FILE INFORMATION | STRINGS

Filter by category

Capability	Category
execute anti-debugging instructions	ANTI-ANALYSIS
check for time delay via QueryPerformance...	ANTI-ANALYSIS
clear Windows event logs	ANTI-ANALYSIS
send data	COMMUNICATION
receive data	COMMUNICATION
execute shell command and capture output	COMMUNICATION
resolve DNS	COMMUNICATION
create pipe	COMMUNICATION
create two anonymous pipes	COMMUNICATION
read pipe	COMMUNICATION
initialize Winsock library	COMMUNICATION
receive data on socket	COMMUNICATION
send data on socket	COMMUNICATION
create UDP socket	COMMUNICATION
hash data with CRC32	DATA-MANIPULATION
encode data using XOR	DATA-MANIPULATION
add user account	HOST-INTERACTION
add user account to group	HOST-INTERACTION
accept command line arguments	HOST-INTERACTION
query environment variable	HOST-INTERACTION
set environment variable	HOST-INTERACTION
get common file path	HOST-INTERACTION
read file on Windows	HOST-INTERACTION
write file on Windows	HOST-INTERACTION
get memory capacity	HOST-INTERACTION
access the Windows event log	HOST-INTERACTION
create or open mutex on Windows	HOST-INTERACTION
check mutex on Windows	HOST-INTERACTION
get networking parameters	HOST-INTERACTION
get hostname	HOST-INTERACTION
get system information on Windows	HOST-INTERACTION
check OS version	HOST-INTERACTION

How to replay your Threatray instance

Here are the analysis IDs for the examples shown:

- For the Code Capabilities examples using the TunnelSpectre APT we have used 38245fc0-9cef-43cd-9c5c-1a32b28dac0b