Whitepaper

Behind the Bulwark:

Anatomy of a EDR/AV Evasion Toolkit

SOCRadar®
Your Eyes Beyond

socradar.io

Contents:

Bulwark

What is Bulwark

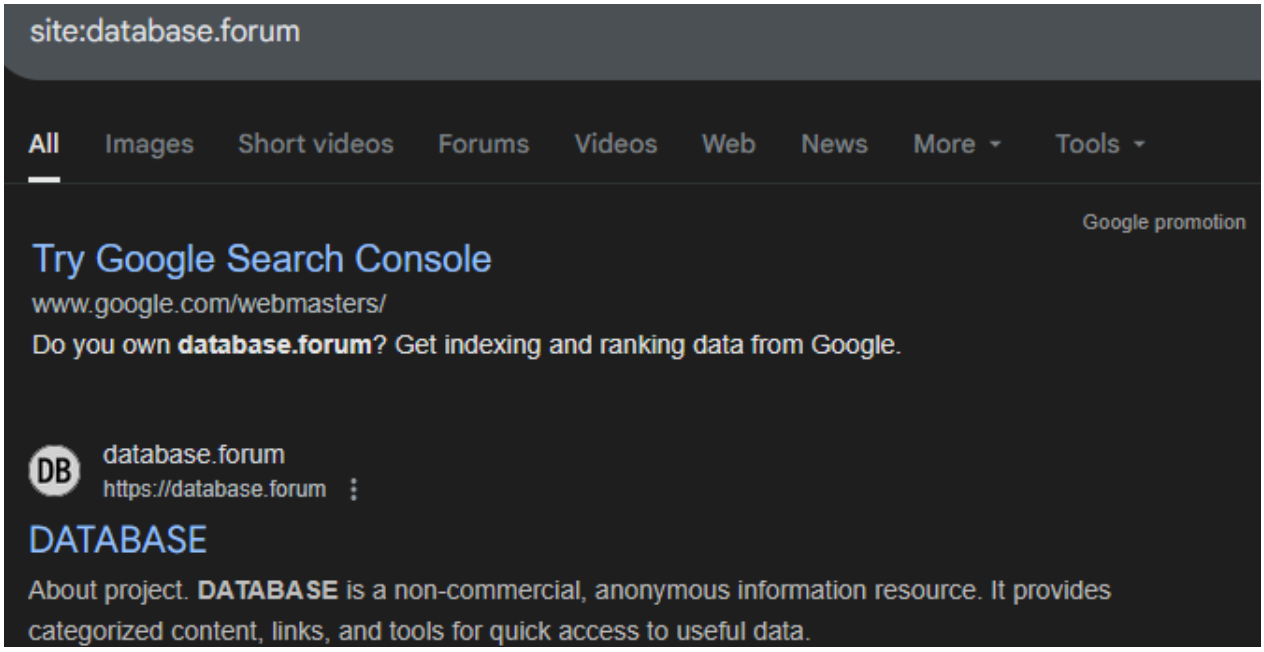
Bulwark is a new tool marketed on Deep Web sites and the Dark Web. It is designed to bypass modern antivirus and EDR solutions, which constitute one of the main lines of defense for most organizations.

The tool allows Windows executables to be protected in a way that enables them to run other binaries in protected environments, effectively circumventing those protections. Although it is presented as a capability intended for "security research purposes only", its publication has clearly resonated across channels used by various threat actors.

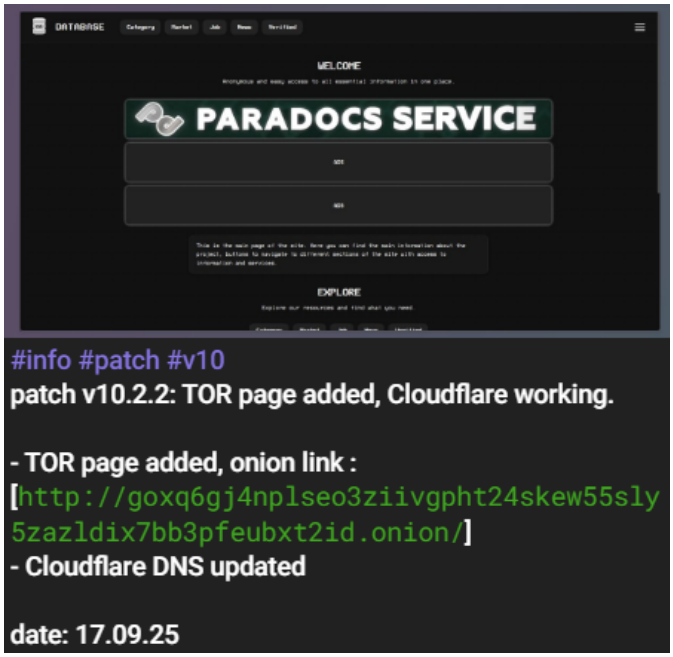
Bulwark landing page

The previous road

Bulwark began appearing in Telegram channels in July, showcasing its capabilities and promising an effective bypass for any EDR or antivirus solution. During continuous hunting activities, SOCRadar's research team detected an announcement referencing a platform called Database.forum, where this tool was listed. At the time, that database was not indexed by mainstream search engines and formed part of the Deep Web, and has recently been added to the Dark Web as well; over the following days, its popularity grew, and it later became discoverable via traditional search engines.



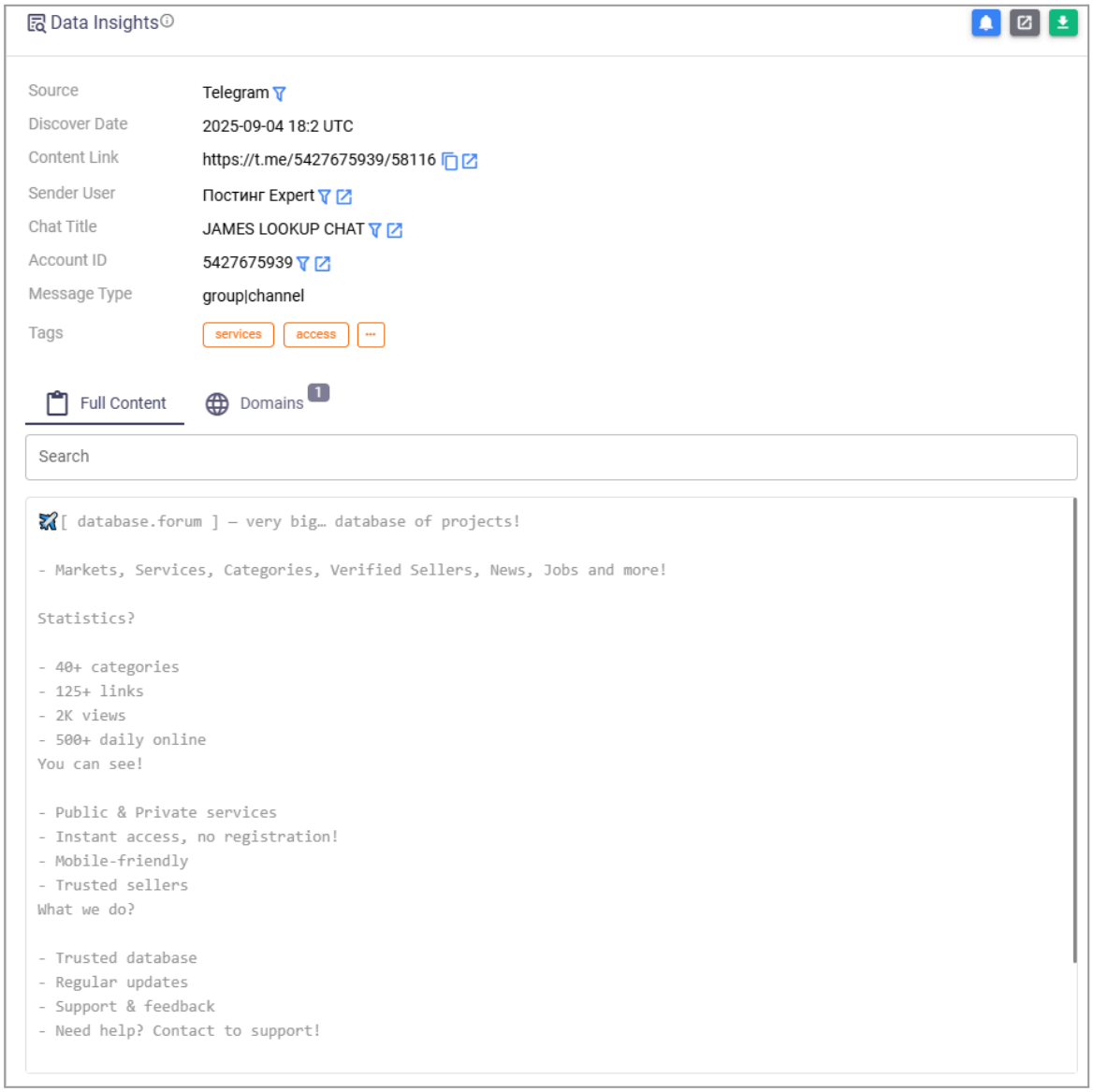
Google results of database[.]forum



Telegram message with the Tor link advertising the database

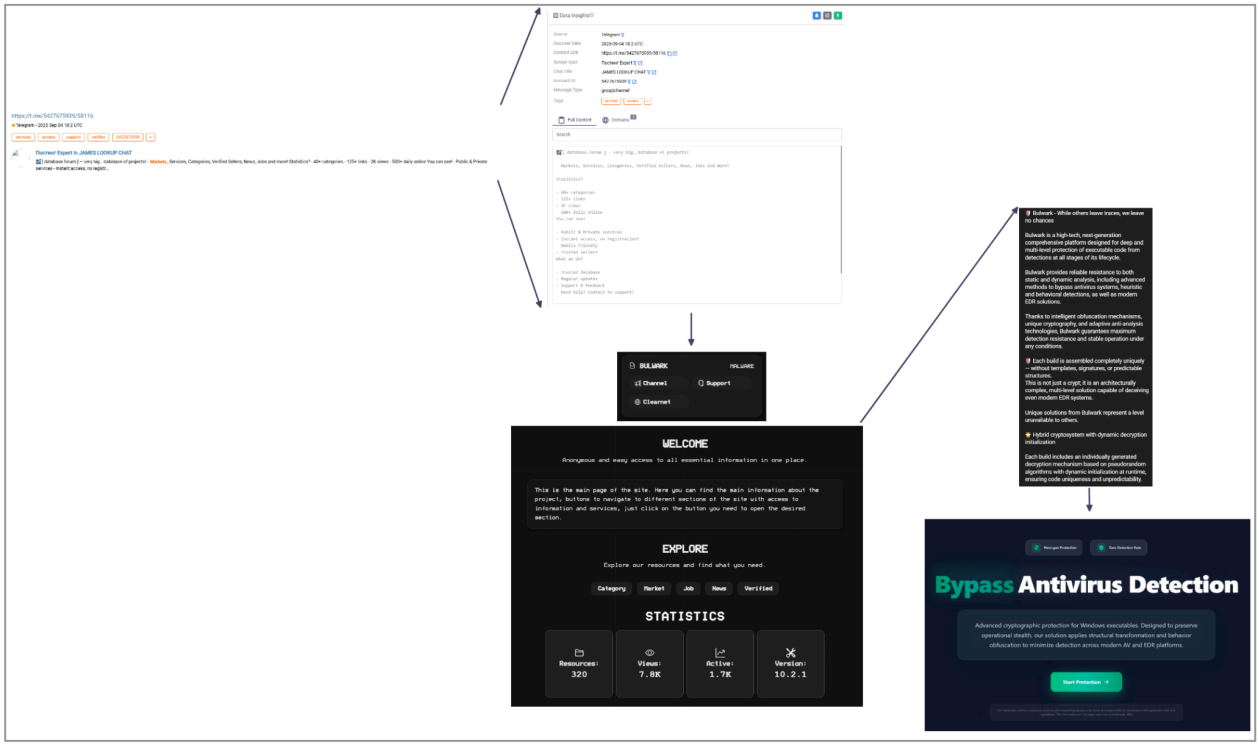
During threat hunting tasks on the SOCRadar platform, we discovered a database forum, where tools such as Bulwark, Aura Stealer, etc. were found.

SOCRadar entry displaying database[.]forum



Detailed SOCRadar entry displaying database[.]forum

Thanks to the capabilities of our platform, information was obtained not only about Bulwark but also about other related tools, as well as their communication channels, which may be of interest to various malicious threat actors.



A diagram showing connections between related malicious sites

Database platform

To understand how Bulwark came to be, it is necessary to go through Database.forum which is a portal run by affiliates and developers where various tools of different kinds are advertised and indexed. Many of these tools are related to threat actors or capabilities that can be used by them.

The Database lets us find both cybersecurity-related tools or channels (tutorials, blacklist services, etc.) and other resources closely tied to threat actors or criminal activity (malware, bypass tools, bots, logs, etc.). Everything is tagged for easier discovery, and the majority of the tools advertised are in Russian.

According to its internal record, the Database was created in early August 2025, so it has evolved very quickly and attracted an unusually large number of tools and affiliates.

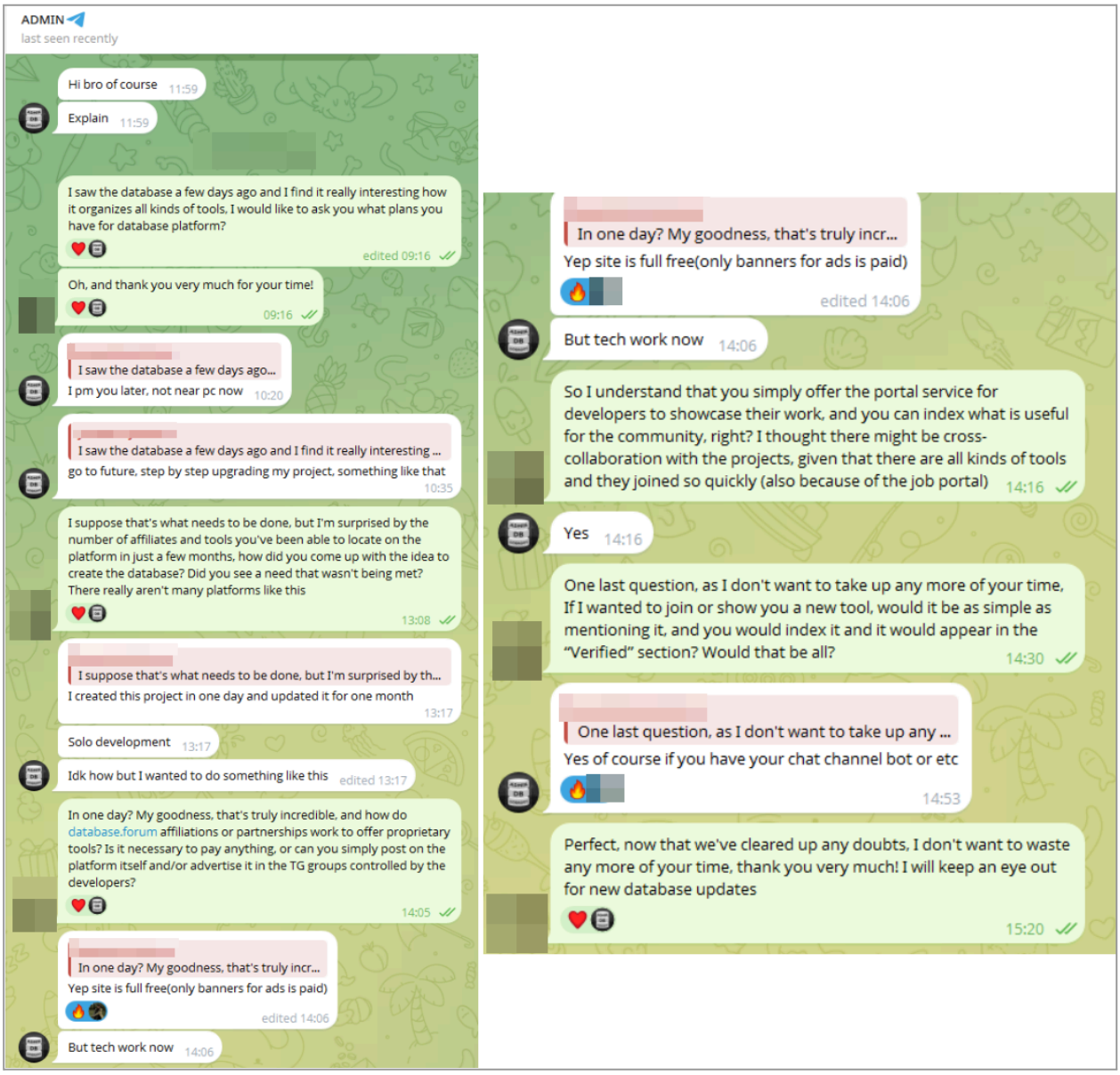
Interview

!! Disclaimer !!

"To better understand the creator's perspective, questions were conducted strictly for research purposes. Throughout the conversations, a technical and respectful approach was maintained, without presuming the interviewees' participation in illegal activities, with the purpose at all times being to understand the techniques used, inform the community, or contribute to academic knowledge."

During the interview with the creator of the Database.forum, key points were verified:

- It was created by a small team.
- They have alliances with affiliates and partners, but do not get involved in developing other tools.
- If we affiliate with Database.forum, no payment of any kind will be necessary, but the database can be promoted through the developers' internal channels, and they have enabled ads within the website for those who want to advertise.
- Their short and medium-term objective is to continue improving the platform and adding more affiliates that will include tools of all types.

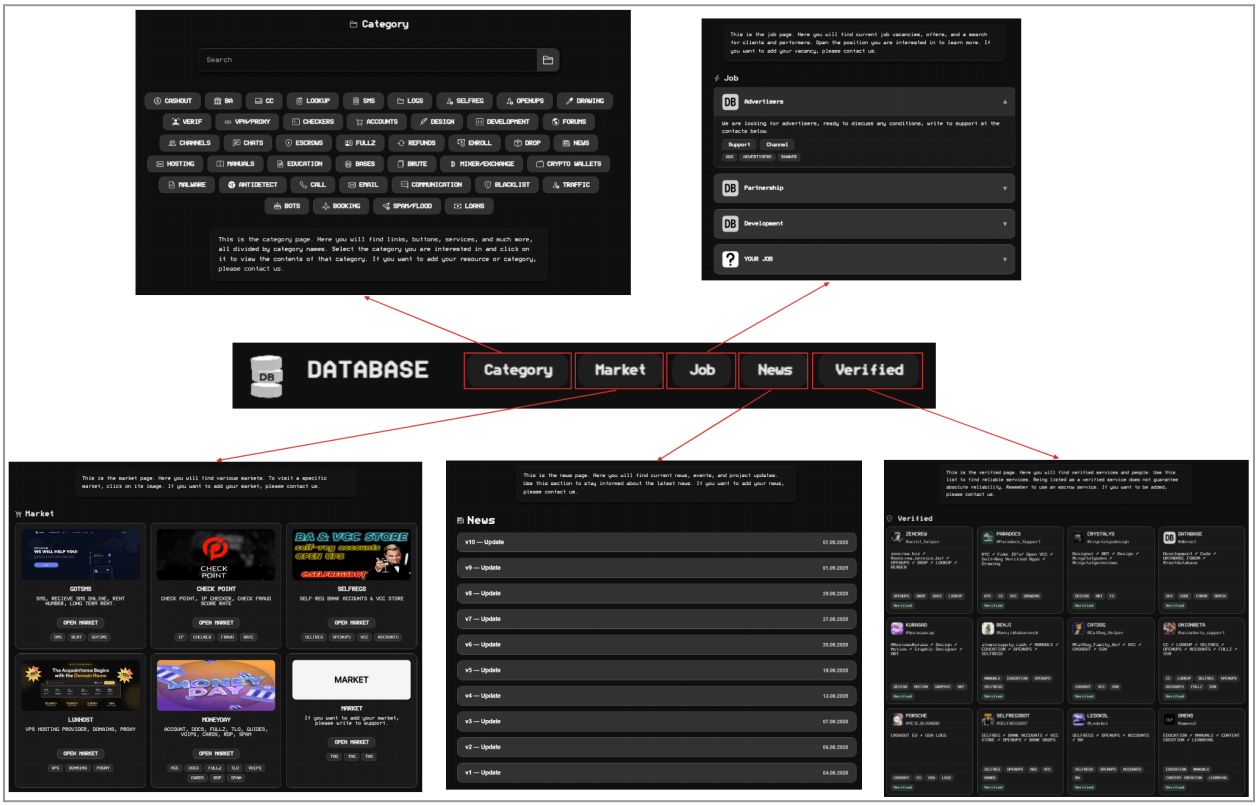


Interview screenshot with the creator of the database[.]forum

Our Findings

Using the header, the database exposes different sections such as Category, Market, Job, News, and Verified.

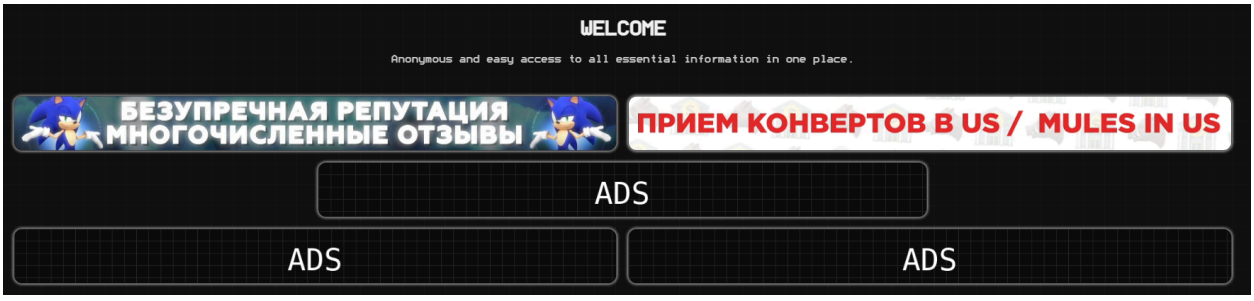
- **Category:** Allows filtering by types of tools and services indexed in the platform
- **Market:** Contains information and access to different markets related to capabilities that can be used by actors (No-phone support, Domains, Bank accounts)
- **Job:** This is a portal where positions or potential affiliate opportunities are advertised by people seeking to promote their tool
- **News:** Historical changelog since creation, showing improvements added to the database
- **Verified:** Portal of affiliates, collaborators, and/or developers who sell, advertise, or contribute to the database



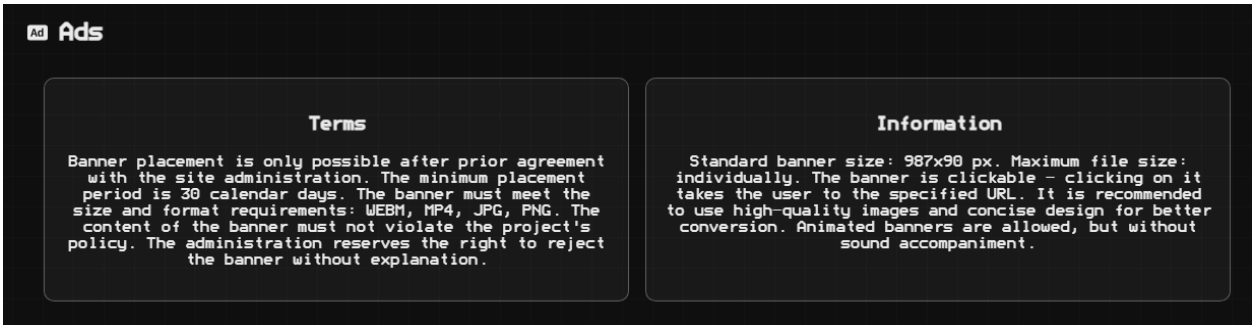
A diagram showing different parts of database[.]forum

This panel is updated, and more elements are added weekly. In addition to that, they added “Donations” to support database.forum financially, taking advantage of the boom in popularity of the database.

The donation part of database[.]forum



Ads part of database[.]forum

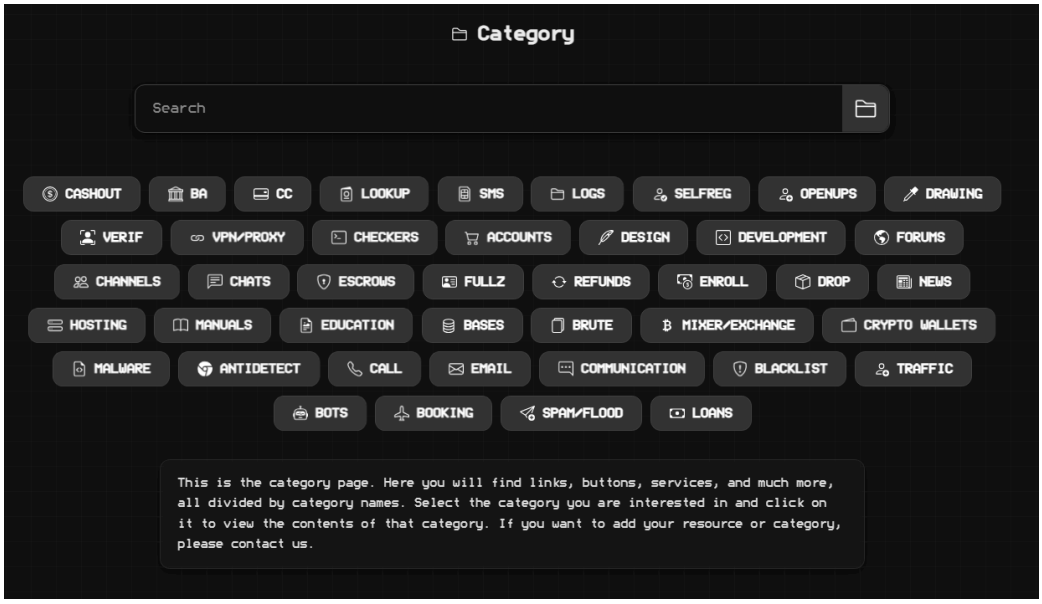


Information about advertisements on database[.]forum

Tool categories

Within the tag-based tool structure, there are different categories of significant importance. Some of these can be used by researchers or cybersecurity companies, while others are clearly for threat actors, such as stealers, EDR bypassers, malware, carding info, etc.

The vast majority of tools listed in the database rely on Telegram for support and typically require acceptance to join. Many share similar characteristics, the same narrative thread, and use Russian as the primary language.



Categories on database[.]forum

We can divide the tools according to intended use and how database.forum presents them:

High-risk tools — commonly used by threat actors

- **MALWARE:** Distribution and information about malicious software, including names like WebCrypter, Bulwark, or Aura Stealer.
- **STEALER:** Credential/information theft portals, for example, the new BEE stealer.
- **BOTS:** Automation for attacks or related tasks, such as CryptoBot, LUXHOST, or Luxury Socks.
- **SPAM/FLOOD:** Resources for carrying out attacks like DDoS or phishing, e.g., Cat d'phisher or MoneyDay.
- **SIP/SPOOF:** Identity spoofing, such as Loki Spoof.
- **Logs:** Tools or malware that provide logs, for example, LummaStealer.

Tools usable for illegal activities

- **LOOKUP:** Information searches (OSINT vs doxxing), with tools like Duck Lookup, Zen Find Up, etc.
- **CHECKERS:** Data validation (auditing vs verifying stolen data), such as OMG Checker or ATOM checker.
- **VPN/PROXY:** Legitimate privacy services as well as tools for hiding activity, like LuxProxy, BigMama, or VentasVPN.
- **ANTIDETECT:** Tracking and detection evasion and multi-account tools like Octo Browser, Dolphin-Anty, Gologin, or AdsPower.
- **TRAFFIC:** Traffic analysis or manipulation, possible click fraud, or fake engagement tools such as TOPTG, Banana Traff, or Connect Traffic.

Dual-use tools

- **LOOKUP:** Information searches (OSINT vs doxxing).
- **CHECKERS:** Data validation (auditing vs verification of stolen data).
- **VPN/PROXY:** Legitimate privacy vs concealment of illicit activities.
- **ANTIDETECT:** Tracking evasion vs security evasion.
- **TRAFFIC:** Traffic analysis vs artificial traffic generation.
- **EDUCATION:** Educational resources that may also hide hacking or fraudulent content.
- **MANUALS:** Resources similar to Education.
- **HOSTING:** Hosting services like aeza or LUXHOST.
- **COMMUNICATION:** Communication tools, sometimes general (Telegram) or used by threat actors (Utox, Pidgin).
- **DESIGN/DEVELOPMENT:** Development and design tools, such as Showy Design or Prince Harry Design.
- **BLACKLIST:** Lists of addresses used for evading security or detecting indicators, for example, Mnogo or Soprano BlackList.

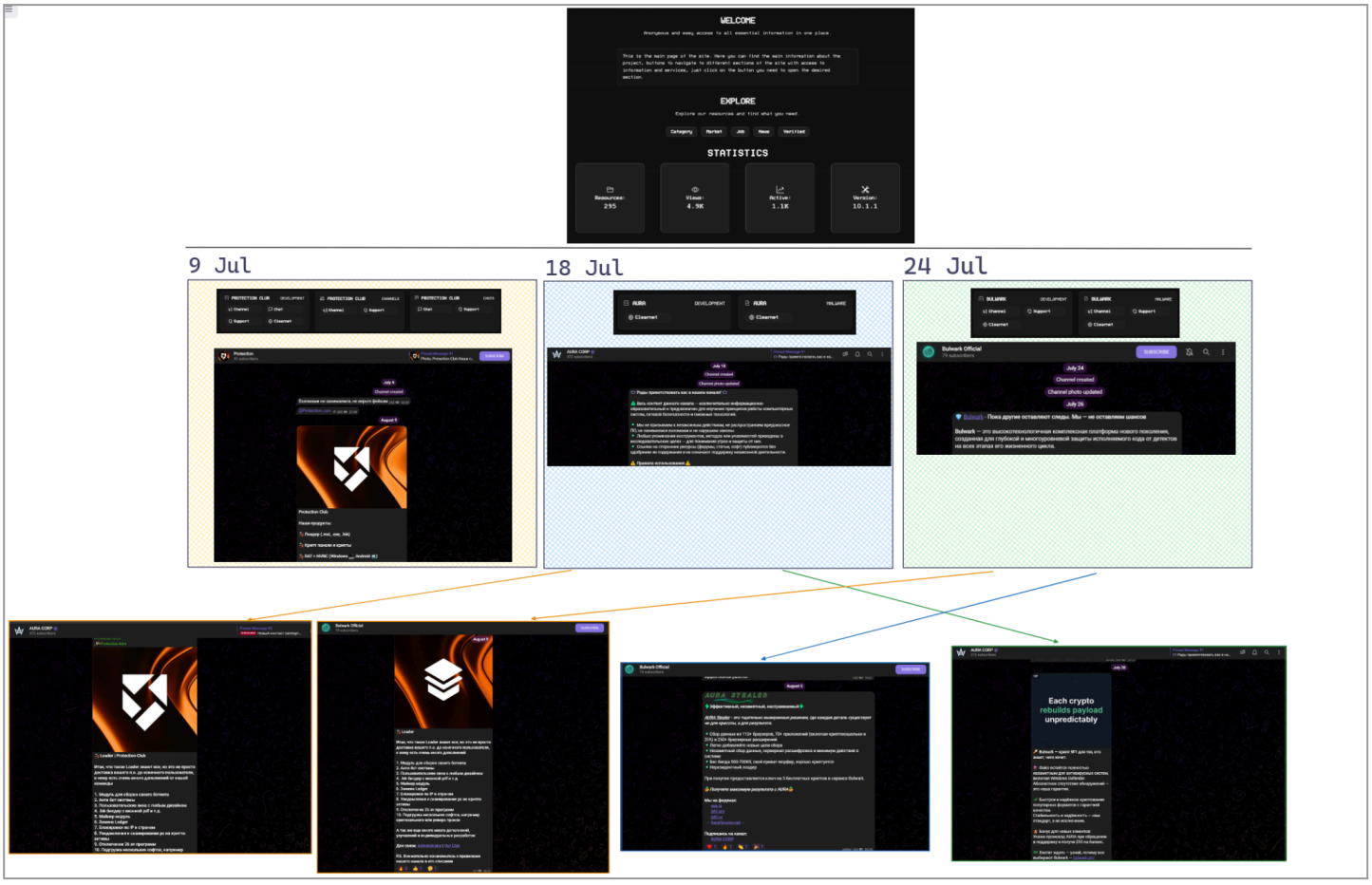
Communications

- **NEWS:** Sector news, leaks, and tool information, with channels like Spectrum, FraudFM, or WhatLeaks.
- **FORUMS:** Common deep/dark web communication channels: XSS, Exploit.in, LeakBase, etc.
- **CHANNELS/CHATS:** Communication with affiliates and tool developers listed in the database.

Correlation tools

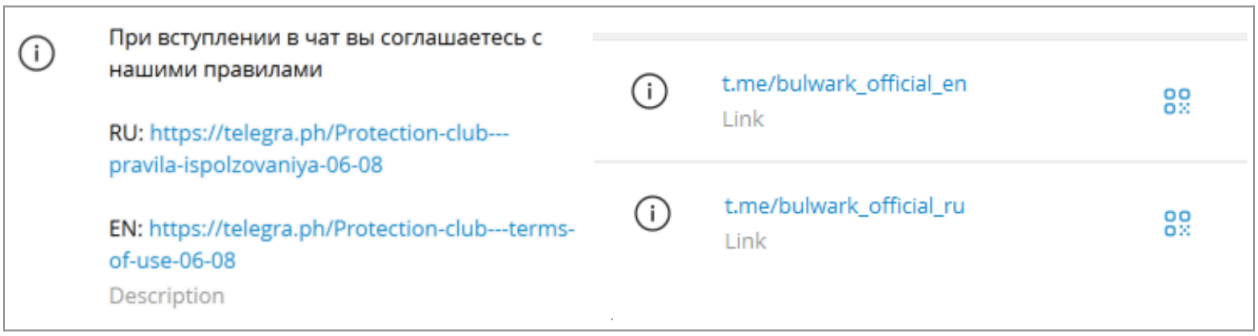
Within the large portfolio available in the database, some tools stand out, either because they are lesser-known but highly critical or because they are already well-known and have caused significant damage to companies.

In addition, another characteristics found to be most relevant is the connection that exists between some of the developers of each project, cross-promoting products on their platforms (mostly Telegram), creating a network of tools among collaborators from different products that range from malware to EDR bypass tools that can be used by multiple threat actors.

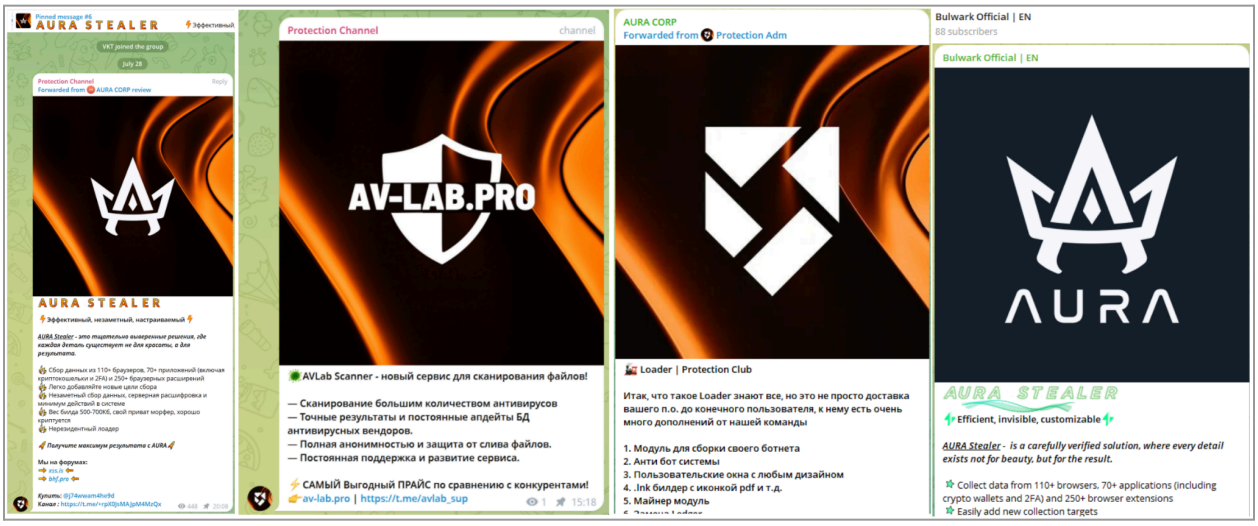


Connections between Bulwark, Aura Stealer, Protection Club

In this correlation, we found evidence that Bulwark, Aura Stealer, Protection Club, and others are closely related, both when they were created, because of the similar form of communication used by all of them, as well as sharing posts in their channels about homonymous tools, showing a degree of affiliation, as well as launching joint offers. This fact leads to further investigation of each of them and their involvement in any of the projects.



Screenshots from Protection Club's and Bulwark's Telegram channels, showing similarities in structures



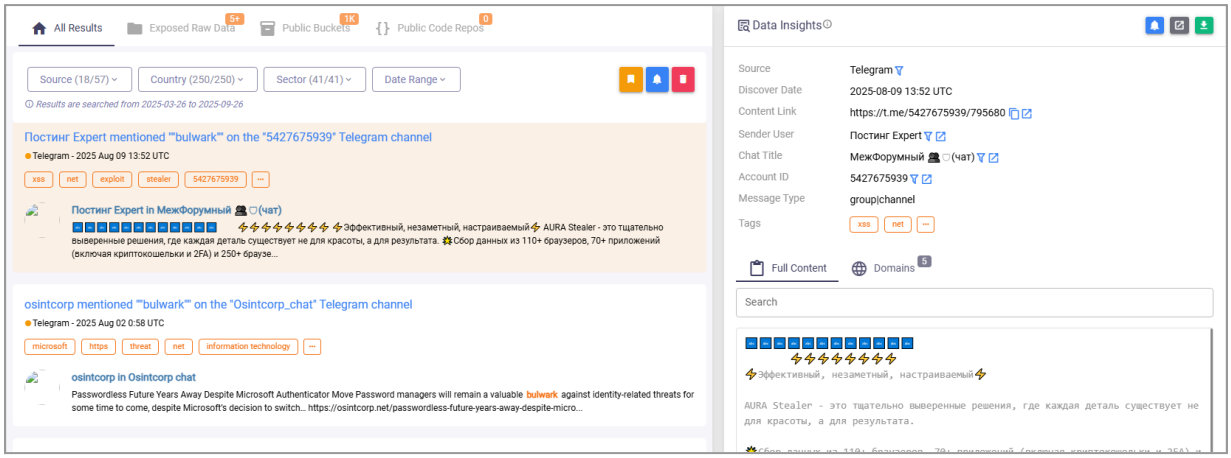
Screenshots from Aura Stealer, AV-Lab and Protection Club, showing connections between the creators

Bulwark

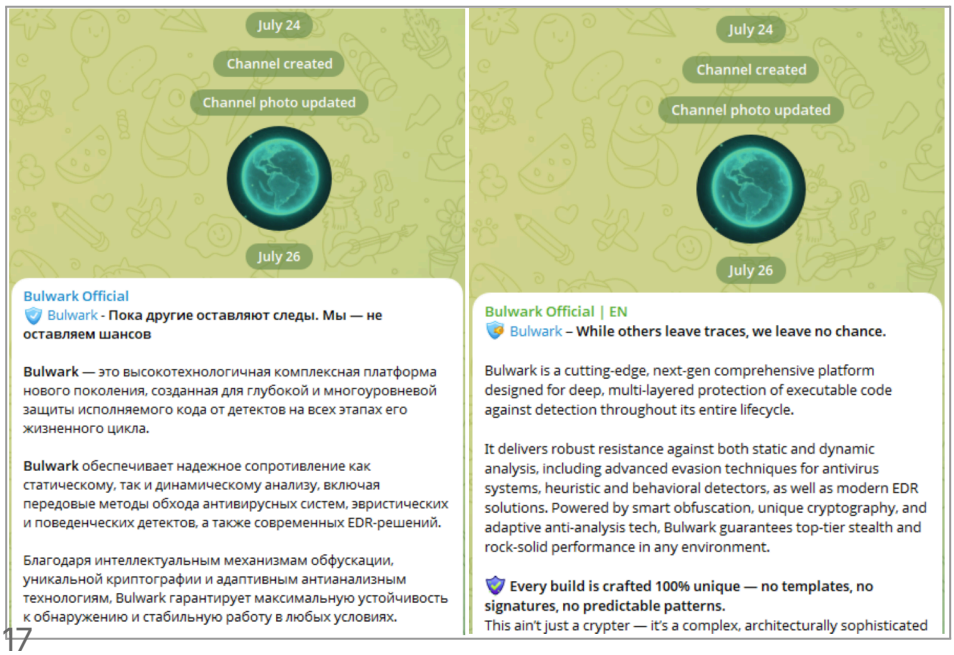
Overview

As mentioned above, Bulwark is a tool specially designed to bypass software such as EDR or Antivirus from all recognized brands on the market.

The project began to gain visibility in July 2025 through its newly created Telegram groups, where it openly advertises itself through two channels, one in Russian and the other in English.



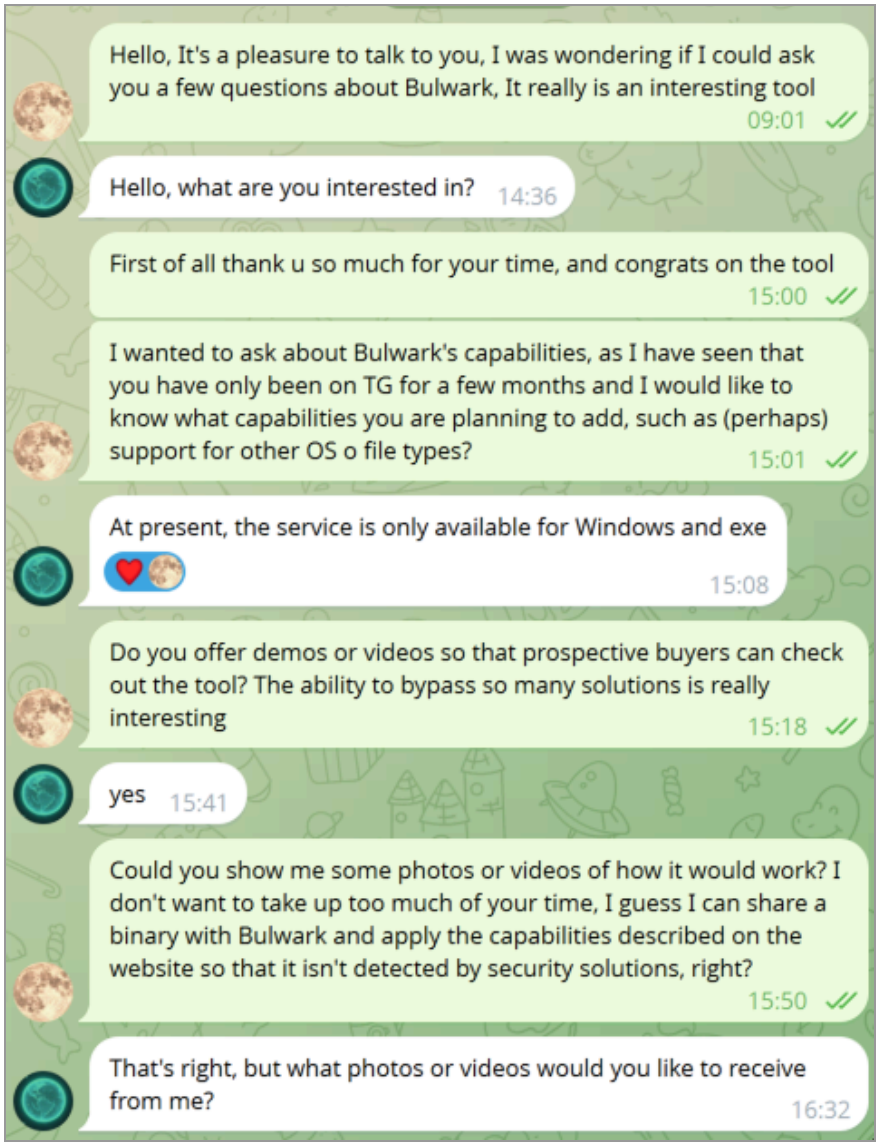
Detailed SOCRadar entry displaying Bulwark



Screenshot from Bulwark's Telegram channel

Interview

During this interview, we sought to verify the effectiveness and usability of the tool, as well as to obtain resources related to demos or images that could confirm the veracity of the work, and to understand whether it had any connection to other related products. The actor provided vague responses and avoided clarifying certain questions.

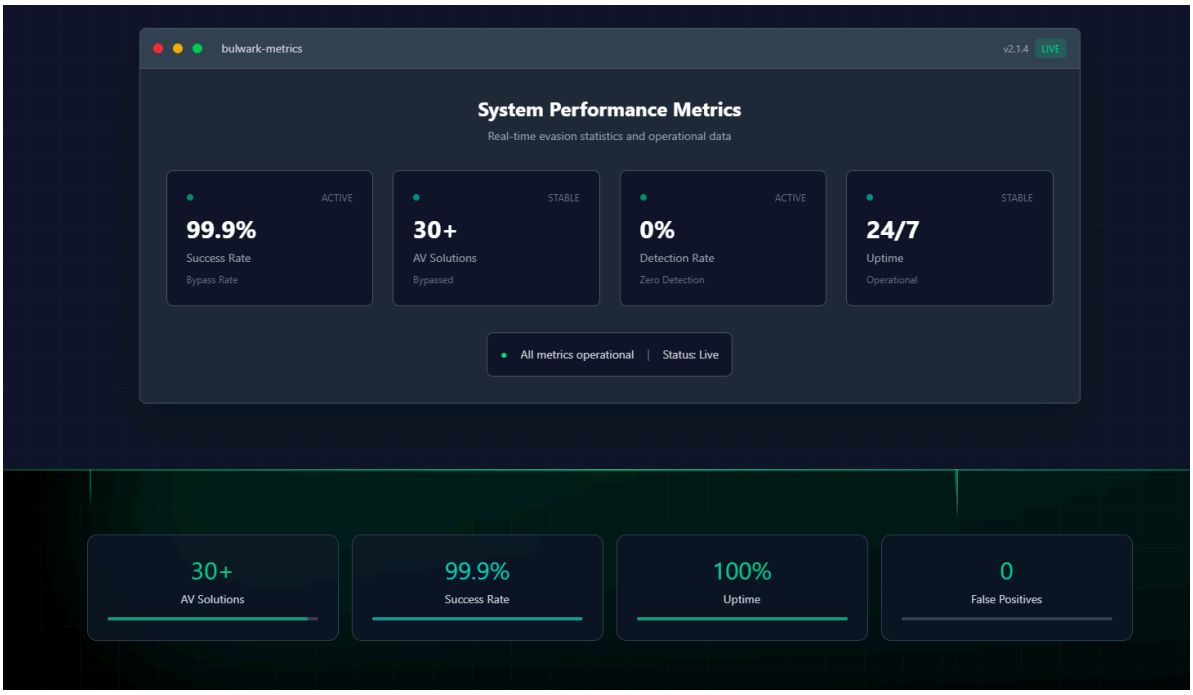


Interview screenshot with the Bulwark channel admin

Targeted Solutions

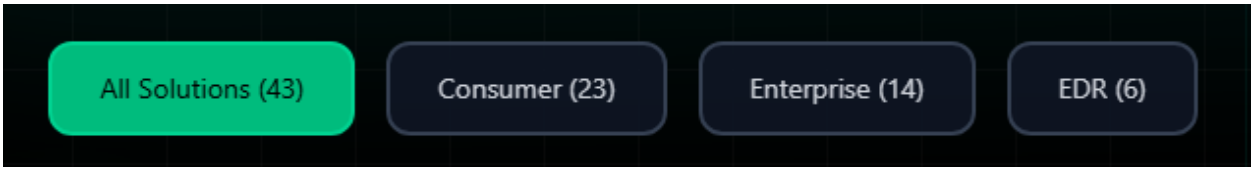
The tool claims to bypass a wide range of security solutions, a claim that obviously attracts the attention of many threat actors since they could use it to encapsulate malware so that it would not be recognized by security software or reverse-engineered, as it has protections against analysis.

According to their own statistics, they claim to have a 99.9% effectiveness rate in bypassing, in which it is capable of evading more than 30 security solutions with a 0% detection rate, numbers that are truly disturbing if true, since it could open a window and have the potential to expose most companies.



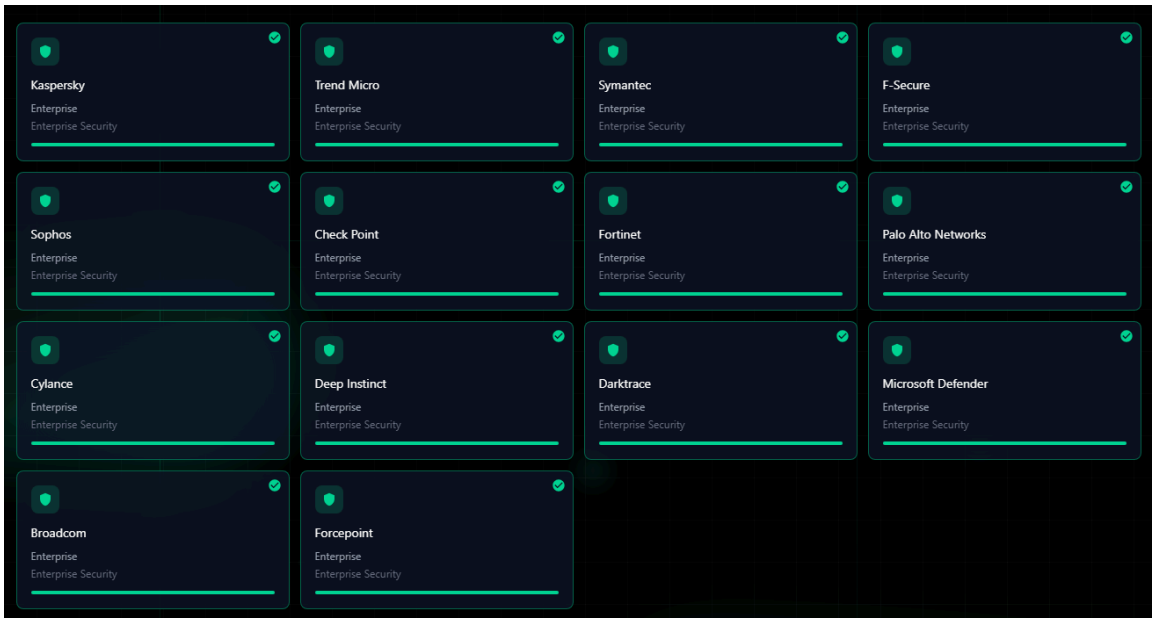
Statistics about the success rate of Bulwark

Within this portfolio of solutions that can bypass security measures, there are different categories, namely classic antivirus and EDR solutions. These can be distinguished by vendor and type of technology, with more than 40 in total.

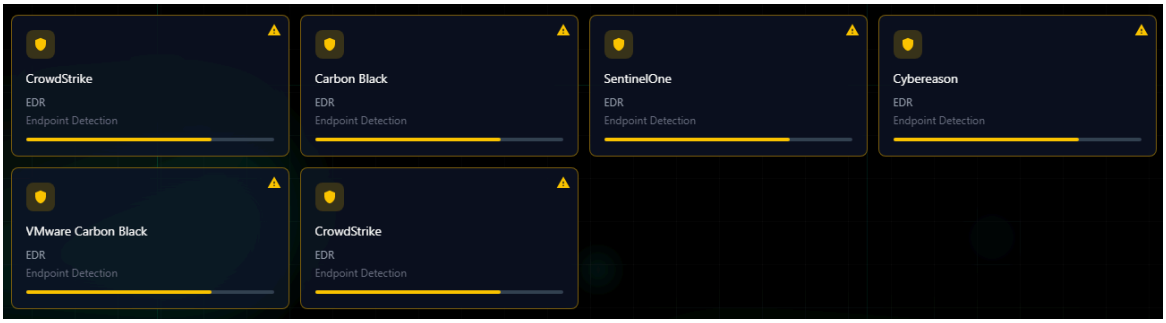


Different product categories that Bulwark can bypass

The AV and EDR solutions that this tool claims to bypass (you can see the detailed list in the following pages):



AV and EDR solutions that this tool claims to bypass



AV and EDR solutions that this tool claims to bypass

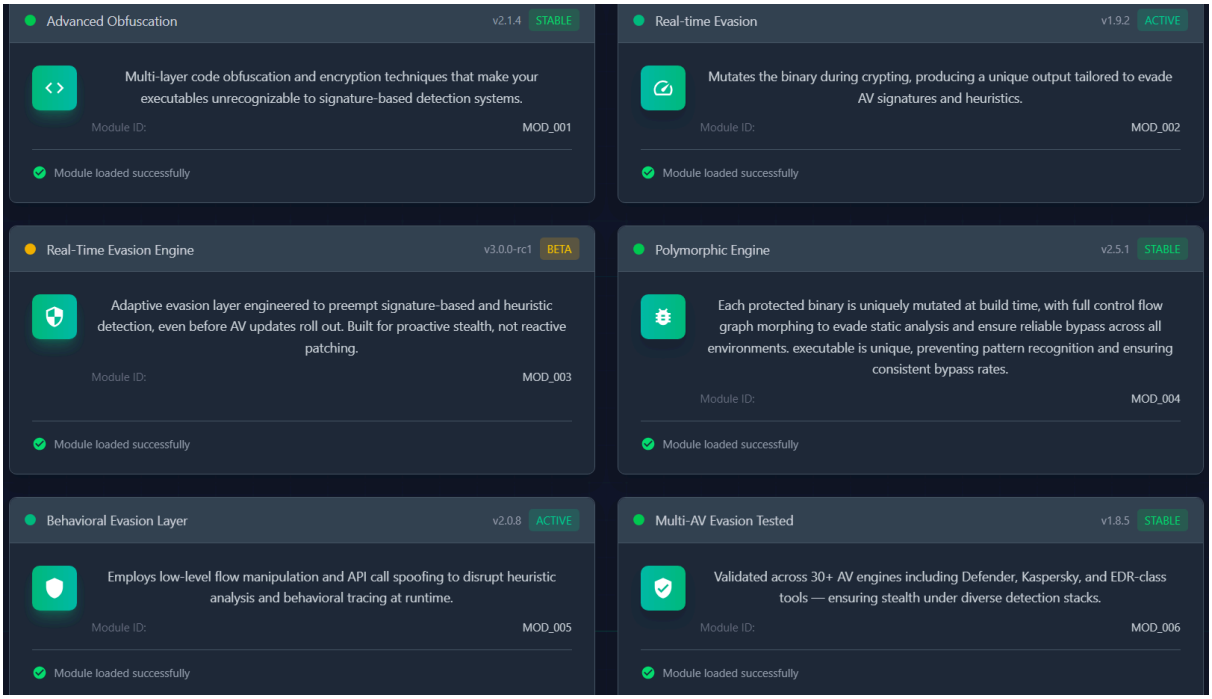
The detailed list of the AV and EDR solutions that this tool claims to bypass include:

Antivirus (theoretically, with virtually total coverage)				
AVG	Check Point	Forcepoint	Microsoft / Windows Defender	TotalAV
Avast	ClamAV	Fortinet	Norton	TrendMicro
Avira	Comodo	F-Secure	Palo Alto	Vipre
Baidu	Cylance	Immunet	Panda Security	Webroot
Bitdefender	Darktrace	Kaspersky	Qihoo 360	ZoneAlarm
Broadcom	Deep Instinct	MalwareBytes	Sophos	360 Total Security
BullGuard	Eset	McAfee	Symantec	

EDR (theoretically, with coverage of around 80%)
Crowdstrike
CyberEason
SentinelOne
VMware - CarbonBlack

Capabilities

There is a capabilities subsection where they list all the techniques they have implemented and that have been added and tested theoretically with satisfactory results.



Capabilities subsection

The capabilities they offer to affiliates are the following:

Advanced Obfuscation	Real-Time Evasion Engine	Behavioral Evasion Layer
Real-Time Evasion	Polymorphic Engine	Multi-AV Evasion Tested

Advanced Obfuscation

They offer a multi-layer code obfuscation that, statically, does not allow the comprehension of the code or makes it difficult, in order to try to evade signature-based detection systems (commonly used in antivirus software). Some of the techniques most commonly used for this purpose are:

- **Opaque predicates:** Causing functions to be always true or false
- **Dead code insertion:** Code that never executes but appears statically as if it does
- **Splitting:** Division of functions into parts, making static analysis incoherent and generating jumps dynamically
- **API hashing:** Use of hashes instead of import names to hinder static analysis

Real-Time Evasion

Changes in the binary during the execution process and/or at runtime, modifying the output or functions for each victim. Some of the techniques most commonly used to achieve this are:

- **Runtime polymorphism:** Changing instructions during execution using equivalent instructions to confuse the analyst
- **Dynamic API resolution:** Resolving APIs at runtime to avoid detectable static imports
- **Self-modifying code:** Modifying the code itself during execution to change its in-memory signature
- **Instruction substitution:** Replacing instructions with functionally equivalent ones during execution and/or runtime

Real-Time Evasion Engine

An adaptive engine designed to evade detection proactively, anticipating signature updates and heuristic analysis. Some of the techniques most commonly used for this purpose are:

- **Environment profiling:** Analysis of the execution environment to detect sandboxes, debuggers, or analysis tools used during reversing
- **Execution timing optimization:** Delaying execution until optimal conditions to avoid automated analysis, commonly creating timing checks or exceptions
- **Anti-analysis detection:** Active identification of dynamic analysis tools and sandboxes via process capture and/or fingerprinting

Polymorphic Engine

Each protected binary is unique at build time, with complete control flow graph morphing that prevents pattern recognition and ensures consistent bypass. Some of the techniques most commonly used for this purpose are:

- **Function reordering:** Random reorganization of function order in the binary to avoid structural signatures that look for code blocks
- **Register allocation randomization:** Random use of different registers for the same functional operations, affecting static analysis and complicating dynamic analysis
- **Instruction expansion:** Converting simple instructions into complex sequences that are functionally equivalent and can be redundant
- **Equivalent instruction substitution:** Replacing instructions with functionally equivalent ones having different opcodes, enabling jumps or calls through other instructions
- **Junk code insertion:** Inserting garbage code that does not affect functionality but alters signatures and hinders analysis

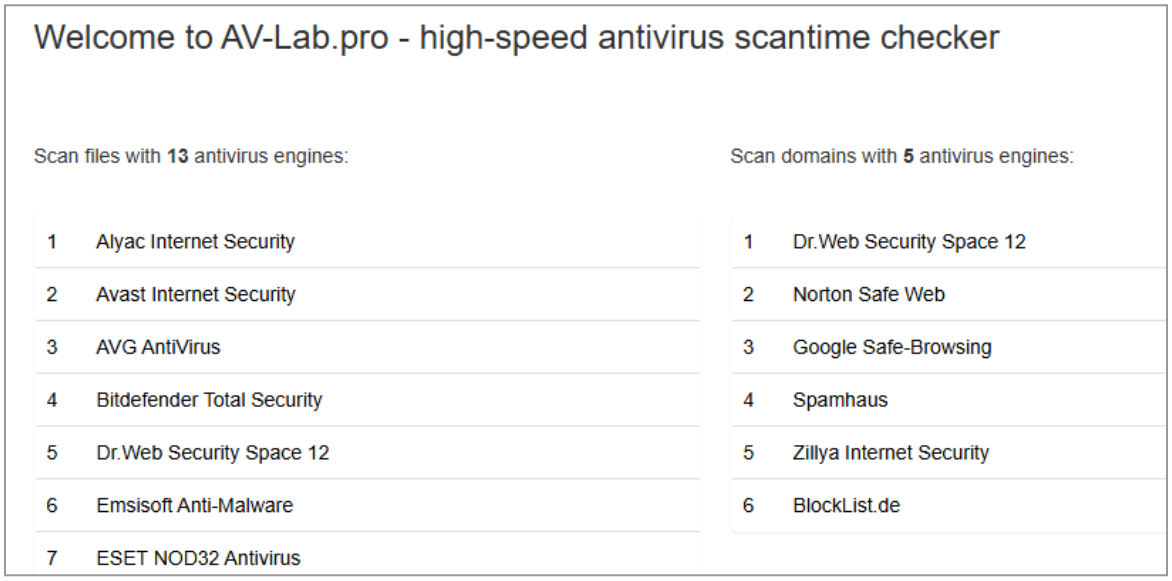
Behavioral Evasion Layer

Low-level manipulation of execution flow and spoofing of API calls to affect heuristic analysis and behavioral tracing, commonly used by EDR. Some of the techniques most commonly used for this purpose are:

- **API call spoofing:** Interleaving legitimate calls among malicious operations to hide patterns
- **Call stack manipulation:** Modifying the stack during execution to hide the real origin of calls that might draw analyst attention or lead to relevant functions
- **Normal process mimicking:** Imitating behavior of legitimate processes (notepad.exe, svchost.exe, etc.)

Multi-AV Evasion Tested

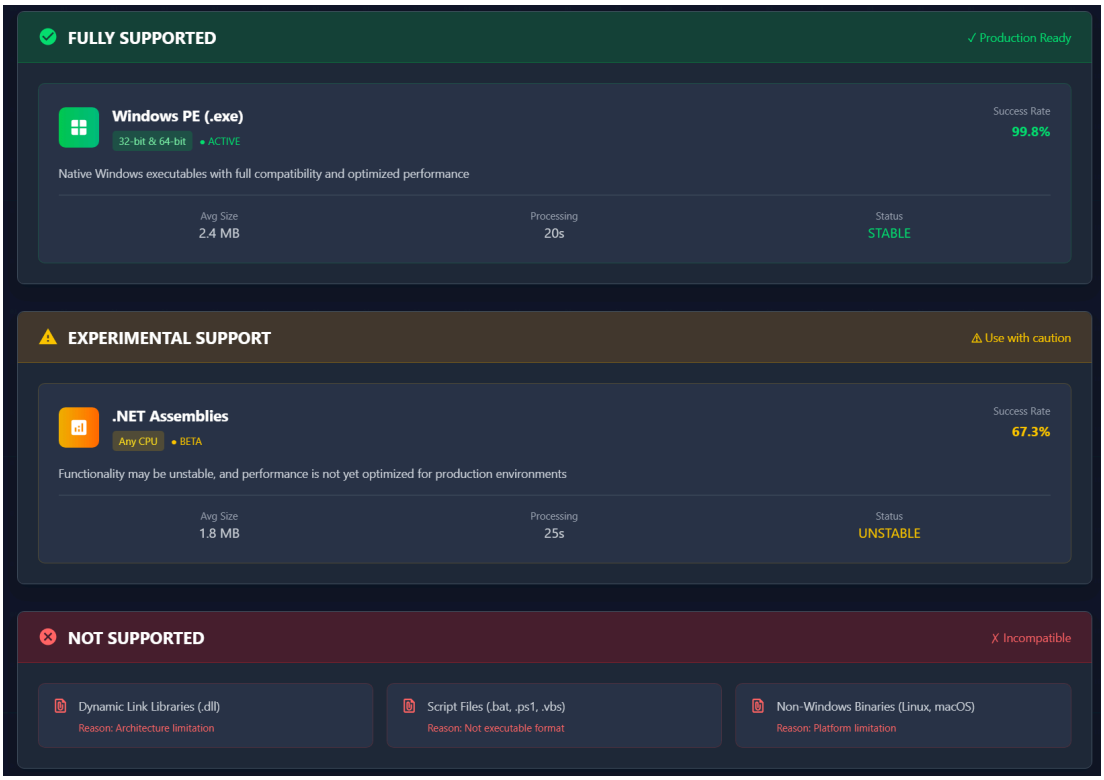
There is a theoretical validation across 30+ detection engines including Defender, Kaspersky, and EDR-class tools, ensuring stealth across various detection stacks. They may have used automated systems or testing tools against multiple AV and EDR engines, such as a tool used by potential affiliates called AV-LAB PRO.



The detection engines utilized by AV-LAB PRO to analyze files

Within the support they offer, they allow the application of these capabilities to portable executable (PE) binaries fully, both 32-bit and 64-bit. In addition, they are improving support for binaries written in .NET, but it still lags significantly behind the alleged success rate of .exe files.

Scripts and libraries are currently excluded, and their focus is centered on Windows, since they do not support other formats such as ELF (Linux) or Mach-O (macOS) at the moment.



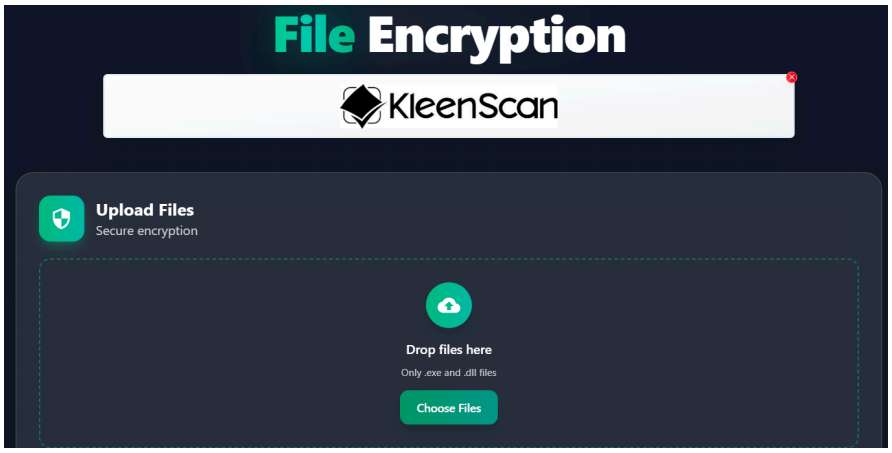
Scope of formats allowed by Bulwark

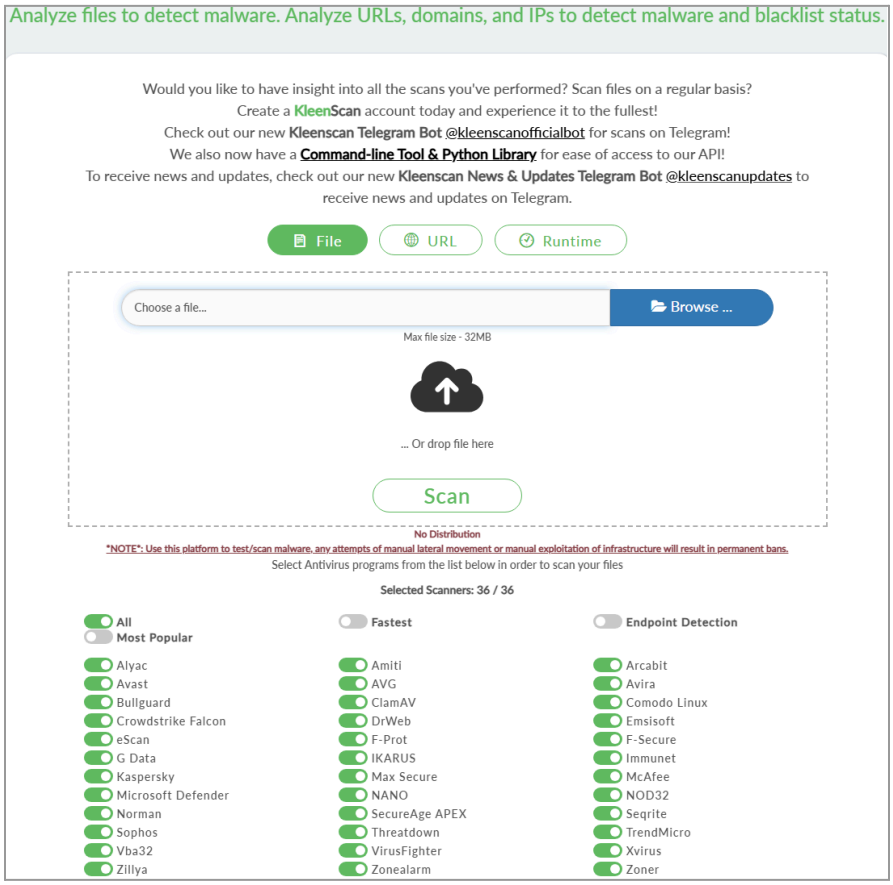
Private Panel

In the private dashboard, you can expand and manage the capabilities available on the platform: manage the plan you purchased, open a support ticket, interact with the API, etc. The dashboard is the front-facing interface that displays all information related to our user account and the capabilities we've acquired.
Bulwark Dashboard:

One of the most relevant features is the panel where we can upload PE files (.exe) so the service runs the obfuscation and the capabilities mentioned above. Additionally, the tool suggests using KleeenScan, a software to analyze and detect malware, possibly to check the binary before and after transformation.

The KleenScan screenshots are as follows:





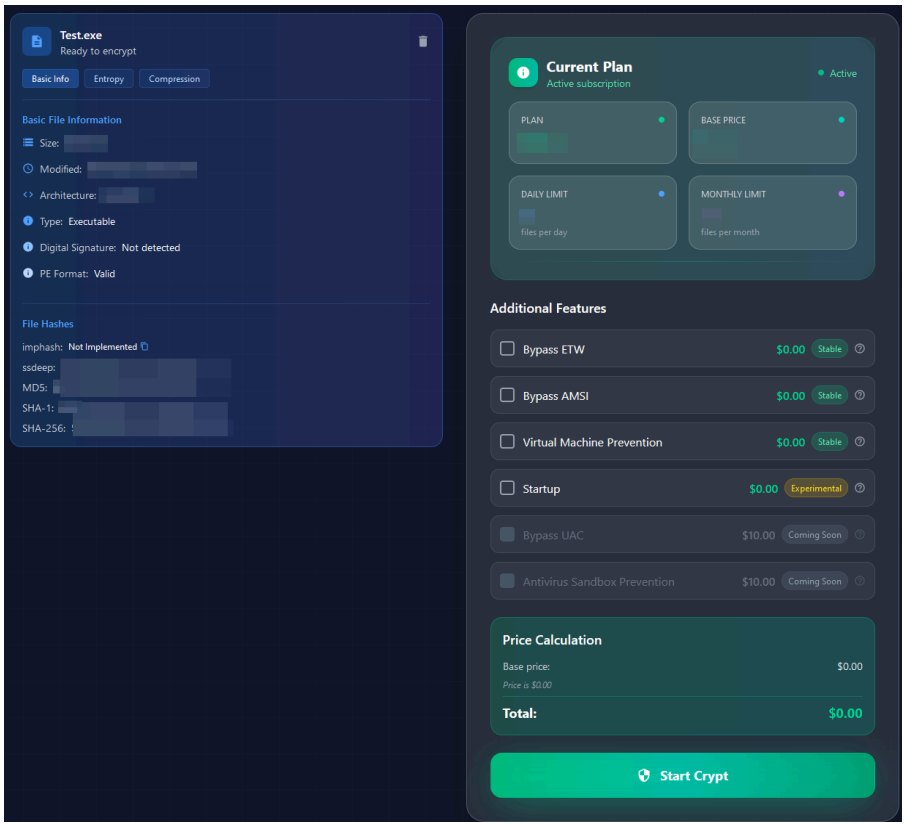
EDR Bypasser Tool Testing

When testing the tool, we checked Bulwark's capability level, as well as the total coverage claimed by the developers, trying to use a highly reported piece of malware in order to check whether various AVs and/or EDRs are able to detect it, with the capabilities that the tool provides.

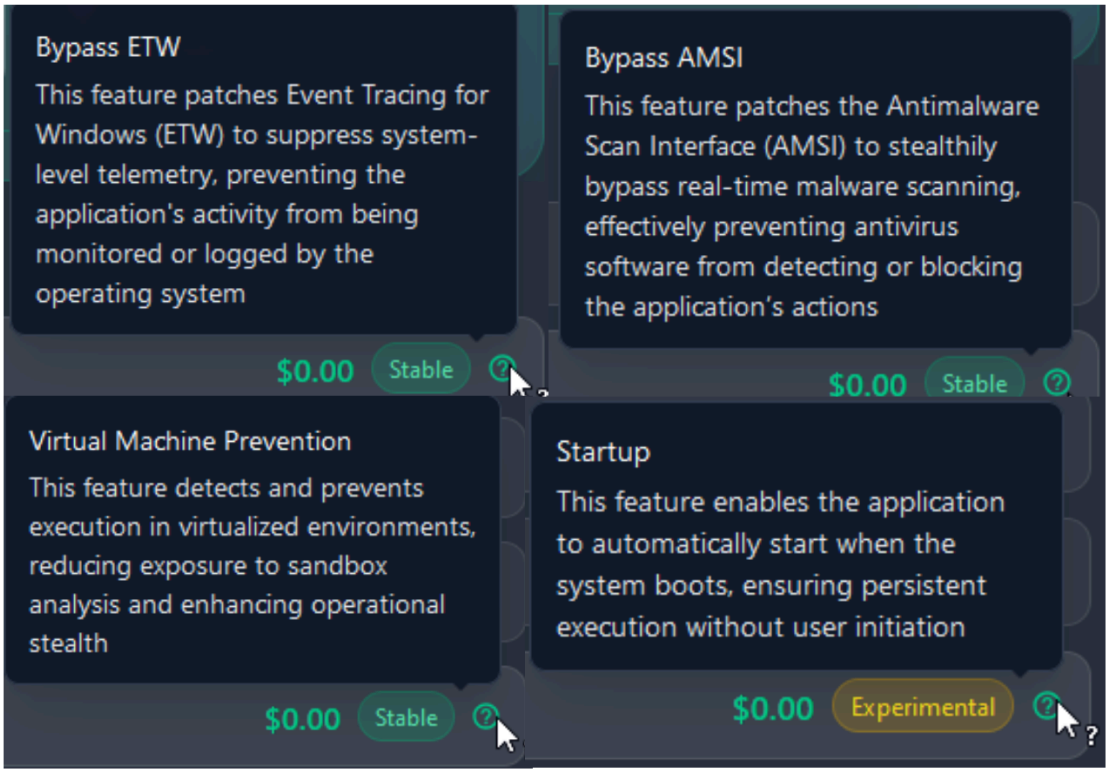
Binary Creation

After choosing the binary we want to protect from AV/EDR detection, in the "File Encryption" section we can submit the file to be converted.

In this section a quick static analysis is performed, capturing relevant information about the binary to be protected (Size, Arch, Hash...) and the capabilities that are currently active to be applied to the binary are shown.



Screenshot of the binary creation module

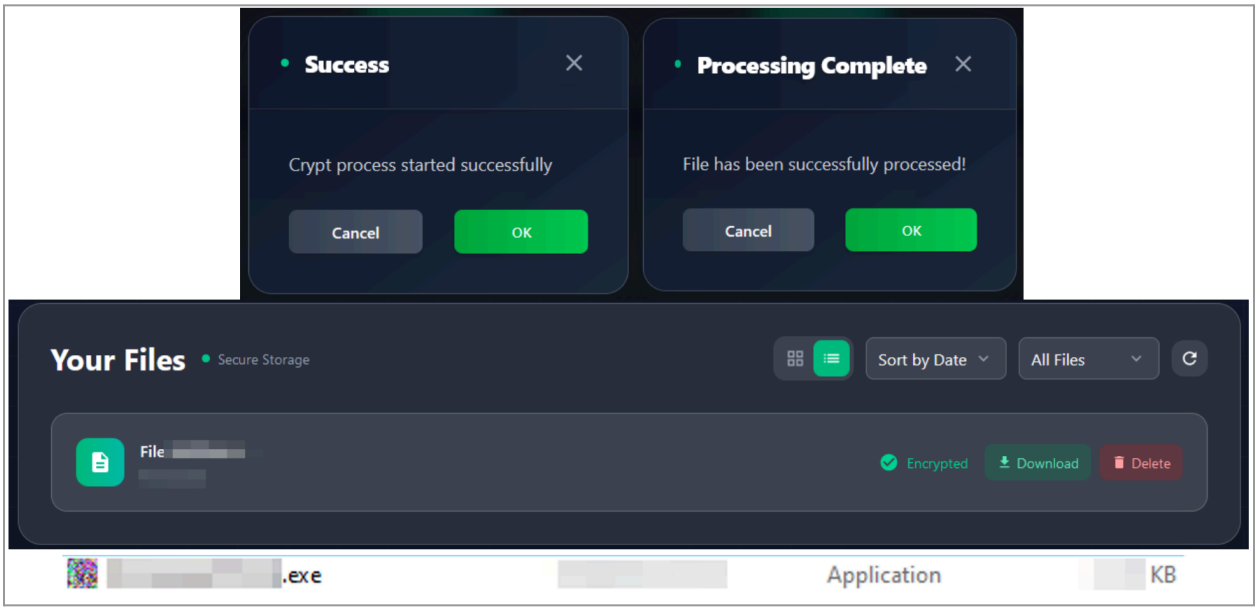


The capabilities of the product which we can add to the binary

These capabilities are extra elements we can add to those mentioned earlier, which give us additional evasion, analysis-resistance, or persistence capabilities:

- **Bypass ETW:** Will allow the binary to disable or evade Event Tracing for Windows (ETW), which is commonly used to monitor system events in real time such as process executions or calls to critical functions.
- **Bypass AMSI:** Will prevent scripts from being passed through the Antimalware Scan Interface (AMSI) if they are executed at any time; AMSI is responsible for real-time analysis of code coming from PowerShell, VBS, JS and others.
- **Virtual Machine Prevention:** Uses Anti-VM techniques to try not to be executed or analyzed in virtual machines, by checking system indicators at runtime to determine whether the host is a real machine or not.
- **Startup:** Generates persistence for the transformed binary by using the Startup folder to launch a copy of it, achieving execution of the potential malware on every system boot.

Once the chosen binary has passed through Bulwark, we obtain a much larger version that consequently has more functions and features than the original binary.



Output from Bulwark

Bulwark Binary overview

The transformation of the binary is total, performing changes in several parts that are visible both statically and dynamically.

Notable changes are observed in the use of resources, where bitmaps are used that typically hide parts of the code to be used and are loaded during runtime.

Icon file of the malicious file after Bulwark

```
001D0740 28 00 00 00 00 01 00 00 00 02 00 00 01 00 20 00
001D0750 00 00 00 00 00 00 04 00 13 0B 00 00 13 0B 00 00
001D0760 00 00 00 00 00 00 00 00 D9 8A 7B FF BA 82 A1 FF
001D0770 A5 A3 9A FF 91 FF D8 FF 5C FF E2 FF A0 C3 B2 FF
001D0780 B9 B3 D0 FF D8 FF BC FF 6C 96 E8 FF CE C4 FB FF
001D0790 BD 98 D0 FF 42 DC A1 FF C9 D0 DA FF DE DC 81 FF
001D07A0 8F E2 C3 FF B9 FF CE FF 8C FF BC FF 3B E0 9C FF
001D07B0 5E 9B D2 FF D1 B5 FF FF BB E6 BB FF AC FF E3 FF
001D07C0 5B FF D6 FF 99 FF FF FF 32 BC 72 FF 71 8E 7B FF
001D07D0 5D 8C C2 FF 4A BD D7 FF 2F F9 8E FF 78 FA 60 FF
001D07E0 5F FF B2 FF BB 91 8F FF DC CC DC FF 38 DC 62 FF
001D07F0 4A 9A C2 FF DD 89 6F FF 7D BE 8D FF EA 73 CC FF
001D0800 FF BE EA FF 8B BD BF FF F3 6C 47 FF FF DD 90 FF
001D0810 FF 74 3D FF 8A 77 55 FF CF BA 41 FF 71 D5 49 FF
001D0820 F7 9B E0 FF 83 C0 DA FF 75 98 B2 FF A5 71 94 FF
001D0830 F3 D9 99 FF D5 89 43 FF 64 70 9A FF 65 CB E4 FF
001D0840 B4 40 EA FF 5E 7E 52 FF 66 2D 43 FF 8D 64 93 FF
001D0850 74 BF 37 FF A0 61 E6 FF D2 63 B5 FF AF 74 62 FF
```

Hex code of the icon file after Bulwark

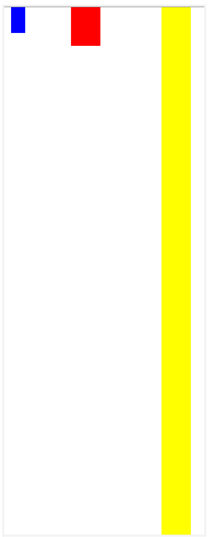
New sections appear where the entropy level increases considerably and where some of them show clear indications of containing code that is extracted during execution, showing the clear difference applied to the binary compared to the original.

headers	header[0]	header[1]	header[2]	header[3]
name	.text	.rdata	.data	.CRT
md5	E3BC4E5E303408C3921363D...	52A01B9F29B18CA848DAC7...	n/a	3F3F0E182F6D9D6A7931B82.
entropy	5.905	5.506	n/a	0.041
file-ratio (93.56%)	75.34 %	17.39 %	n/a	0.83 %
raw-address	0x00000400	0x0000BA00	0x00000000	0x0000E400
raw-size (57856 bytes)	0x0000B600 (46592 bytes)	0x00002A00 (10752 bytes)	0x00000000 (0 bytes)	0x00000200 (512 bytes)
virtual-address	0x00001000	0x0000D000	0x00010000	0x00011000
virtual-size (58009 bytes)	0x0000B40B (46091 bytes)	0x000028D2 (10450 bytes)	0x000005B8 (1464 bytes)	0x00000004 (4 bytes)
characteristics				
value	0x60000020	0x40000040	0xC0000040	0x40000040
writable	-	-	x	-
executable	x	-	-	-
shareable	-	-	-	-
self-modifying	-	-	-	-
virtualized	-	-	x	-

Header information of the file before Bulwark

headers	header[0]	header[1]	header[2]	header[3]	header[4]	header[5]
name	.text	.rdata	.data	.fptable	.rsrc	.reloc
md5	BF227A2F1968CB459057EF6...	D396697CF1202BB656E097B...	4E7C13FC17CDC45C496211...	BF619EAC0CDF3F68D496EA...	4D5BA867834DC993E8263D2...	C561EB20D2CF4AF215C0365...
entropy	6.661	4.695	7.679	0.000	6.427	6.729
file-ratio (99.95%)	61.34 %	15.36 %	8.77 %	0.02 %	12.64 %	1.82 %
raw-address	0x00000400	0x0014D400	0x001A0A00	0x001D0400	0x001D0600	0x00215000
raw-size (2222592 bytes)	0x0014D000 (1363968 bytes)	0x00053600 (341504 bytes)	0x0002FA00 (195072 bytes)	0x00000200 (512 bytes)	0x00044A00 (281088 bytes)	0x00009E00 (40448 bytes)
virtual-address	0x00001000	0x0014E000	0x001A2000	0x001D3000	0x001D4000	0x00219000
virtual-size (2224464 bytes)	0x0014CEC6 (1363654 bytes)	0x000535A2 (341410 bytes)	0x000307D0 (198608 bytes)	0x00000080 (128 bytes)	0x00044818 (280600 bytes)	0x00009C80 (40064 bytes)
characteristics						
value	0x60000020	0x40000040	0xC0000040	0xC0000040	0x40000040	0x42000040
writable	-	-	x	x	-	-
executable	x	-	-	-	-	-
shareable	-	-	-	-	-	-
self-modifying	-	-	-	-	-	-
virtualized	-	-	-	-	-	-

Header information of the file after Bulwark



Comparison of two binaries before and after Bulwark

An important expansion of imports is also observed, adding capabilities to:

- Create and manipulate processes
- Invoke other libraries and imports (allowing capability expansion)
- Perform injections into other processes
- Work with multiple threads and create them
- Anti-VM and Anti-Debugging techniques
- Work with the binary's resources

Additional imports added to the file after Bulwark

Possible SQL queries can also be seen, which could be used to store information or manipulate it during execution or to retrieve obfuscated elements.

```
UPDATE %Q.sqlite_master SET type='%s', name=%Q, tbl_name=%Q, rootpage=#%d, sql=%Q WHERE rowid=#%d
CREATE TABLE %Q.sqlite_sequence(name,seq)
CREATE TABLE
DELETE FROM %Q.%s WHERE %s=%Q
DELETE FROM %Q.sqlite_sequence WHERE name=%Q
DELETE FROM %Q.sqlite_master WHERE tbl_name=%Q and type!='trigger'
DELETE FROM %Q.sqlite_master WHERE name=%Q AND type='trigger'
UPDATE %Q.sqlite_master SET rootpage=%d WHERE #%d AND rootpage=#%d
SETUP
at most %d tables in a join
SEARCH
DELETE
UPDATE
DELETE FROM %Q.sqlite_master WHERE name=%Q AND type='index'
SET NULL
SET DEFAULT
CREATE TABLE %Q.%s(%s)
UPDATE "%w".sqlite_master SET sql = sqlite_rename_table(%Q, type, name, sql, %Q, %Q, %d) WHERE (type!='index' OR tbl_name=%Q COLLATE nocase)AND   name NOT LIKE 'sqliteX_%%' ESCAPE 'X'
UPDATE %Q.sqlite_master SET tbl_name = %Q, name = CASE WHEN type='table' THEN %Q WHEN name LIKE 'sqliteX_autoindex%%' ESCAPE 'X'     AND type='index' THEN 'sqlite_autoindex_' || %Q || substr(name,%d+18) ELSE name END WHERE t
UPDATE "%w".sqlite_sequence set name = %Q WHERE name = %Q
UPDATE sqlite_temp_schema SET sql = sqlite_rename_table(%Q, type, name, sql, %Q, %Q, 1), tbl_name = CASE WHEN tbl_name=%Q COLLATE nocase AND   sqlite_rename_test(%Q, sql, type, name, 1, 'after rename',0) THEN %Q ELSE tbl_name EN
SELECT 1 FROM "%w".sqlite_master WHERE name NOT LIKE 'sqliteX_%%' ESCAPE 'X' AND sql NOT LIKE 'create virtual%%' AND sqlite_rename_test(%Q, sql, type, name, %d, %Q, %Q)=NULL
SELECT 1 FROM temp.sqlite_master WHERE name NOT LIKE 'sqliteX_%%' ESCAPE 'X' AND sql NOT LIKE 'create virtual%%' AND sqlite_rename_test(%Q, sql, type, name, 1, %Q, %Q)=NULL
UPDATE "%w".sqlite_master SET sql = printf('%%.%ds, ',sql) || %Q || substr(sql,1+length(printf('%%.%ds',sql))) WHERE type = 'table' AND name = %Q
SELECT raise(ABORT,%Q) FROM "%w"."%w"
UPDATE "%w".sqlite_master SET sql = sqlite_drop_column(%d, sql, %d) WHERE (type=='table' AND tbl_name=%Q COLLATE nocase)
UPDATE "%w".sqlite_master SET sql = sqlite_rename_column(sql, type, name, %Q, %Q, %d, %Q, %d, %d) WHERE name NOT LIKE 'sqliteX_%%' ESCAPE 'X'  AND (type != 'index' OR tbl_name = %Q) AND sql NOT LIKE 'create virtual%%'
UPDATE temp.sqlite_master SET sql = sqlite_rename_column(sql, type, name, %Q, %Q, %d, %Q, %d, 1) WHERE type IN ('trigger', 'view')
```

Additional SQL queries added after Bulwark

Technical Details

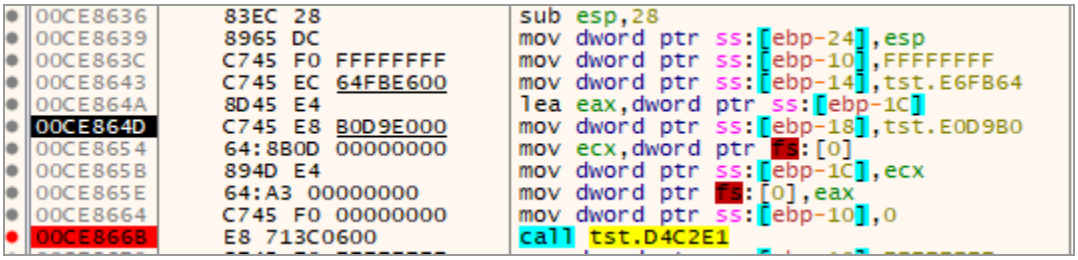
During a controlled run, it is confirmed that the original malware's capabilities have not been altered and, although the execution flow changes, it performs the same runtime tasks while adding the ones selected previously in the file-creation panel.

The binary does not appear to be recognized with any packer, but different sections can be observed from which it can collect information during runtime to perform injections or dynamic code loading.

A screenshot from Detect it Easy

Initially, the binary tries to protect itself against any type of analysis. Statically, it presents functions that are unreadable or difficult to trace with disassembly, and on the other hand, it performs different Anti-Analysis and Anti-DBG techniques to hinder tracking tasks during debugging.

SEH (FS:[0]) usage can be located to control exceptions and try to execute process termination if debugging is detected.



Debugging view of the executable

There are different techniques where functions modify their content during execution causing jumps to other functions, obscuring the execution flow.

Debugging view of the executable

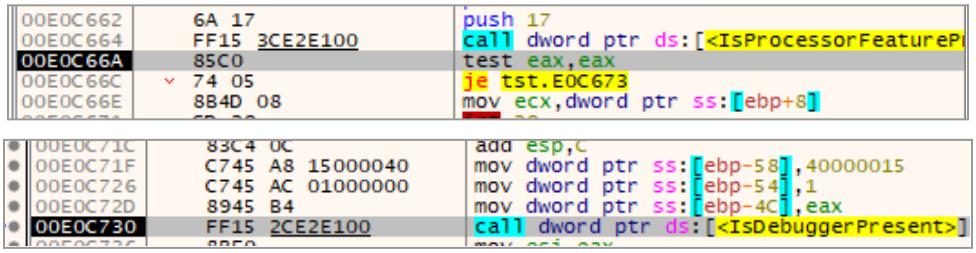
TEB and PEB access to locate Debugger execution flags (TEB.BeingDebugged, PEB.BeingDebugged...)

```
int v5[13]; // [esp+0h] [ebp-34h] BYREF

v5[4] = (int)v5;
v5[8] = (int)&unk_59FB64;
v5[7] = (int)sub_53D9B0;
v5[6] = (int)NtCurrentTeb()->NtTib.ExceptionList;
v5[9] = 0;
sub_47C2E1();
return 0;
}
```

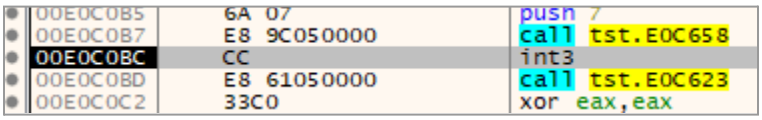
Execution flags related with TEB

More common techniques are also found where some calls to *IsProcessorFeaturePresent* or *IsDebuggerPresent* are used to locate debuggers. Assembly code views of the executable:



Debugging views of the executable

Other techniques are performed to hinder analysis, such as abuse of CC (INT3) for BreakPoint detection and blocking.

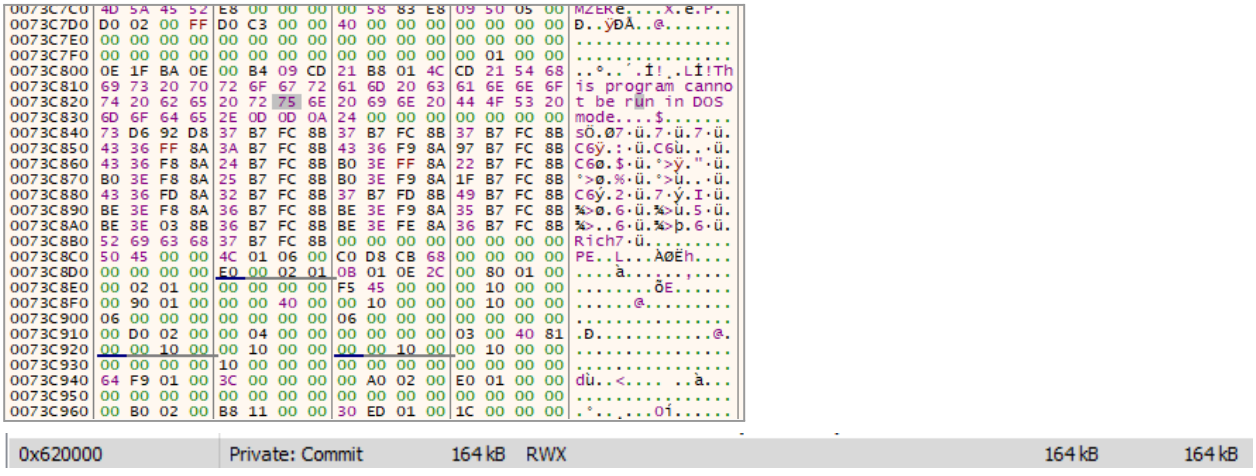


Debugging view of the executable

During runtime, it also invokes other libraries that will be used by both Bulwark and the embedded binary (The malware).

After this, a Portable Executable (PE) is extracted from memory, performing different operations which it constructs, making use of different threads to access different functions.

Extraction from memory of the PE file:

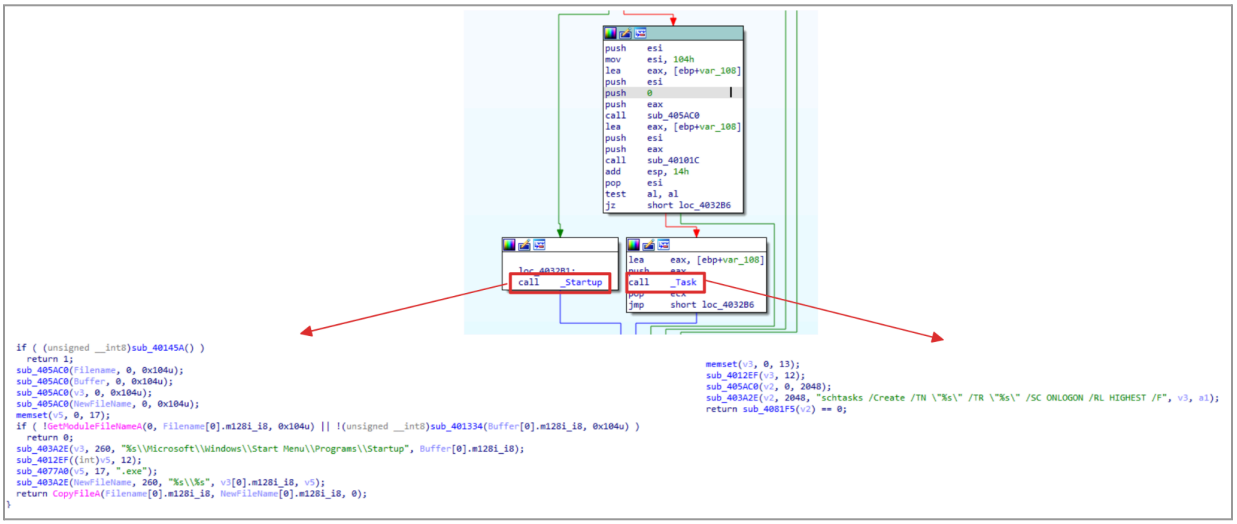


This binary is responsible for performing persistence, where it not only has the capability to use startup, but can also create it using tasks.

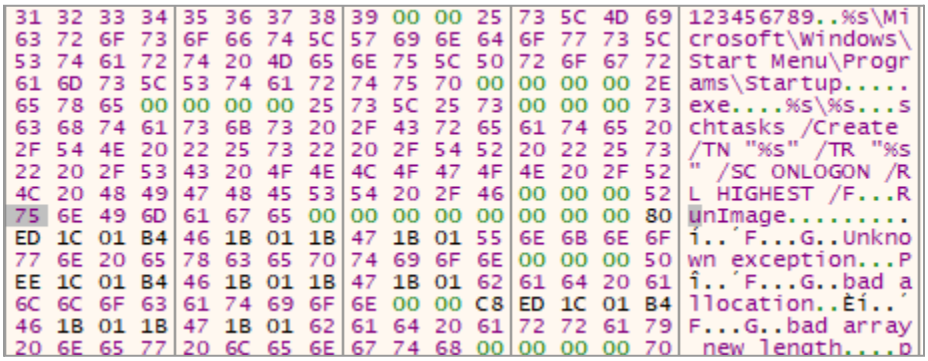
Additional strings added to the file after Bulwark:

```
%s\Microsoft\Windows\Start Menu\Programs\Startup
%s\%s
schtasks /Create /TN "%s" /TR "%s" /SC ONLOGON /RL HIGHEST /F
%S#[k
.exe
.com
.exe
.bat
.cmd
```

```
OpenProcessToken
GetNativeSystemInfo
GetEnvironmentVariableA
VirtualProtect
WriteFile
FindFirstFileExW
FindNextFileW
GetCurrentProcessId
TerminateProcess
GetCurrentThreadId
GetExitCodeProcess
CreateProcessW
GetEnvironmentStringsW
SetEnvironmentVariableW
RaiseException
```



Persistence functions seen in IDA



```
31 32 33 34 35 36 37 38 39 00 00 25 73 5C 4D 69 123456789..%s\Mi
63 72 6F 73 6F 66 74 5C 57 69 6E 64 6F 77 73 5C crosoft\Windows\
53 74 61 72 74 20 4D 65 6E 75 5C 50 72 6F 67 72 Start Menu\Progr
61 6D 73 5C 53 74 61 72 74 75 70 00 00 00 00 2E ams\Startup.....
65 78 65 00 00 00 00 25 73 5C 25 73 00 00 00 73 exe....%s\%s...s
63 68 74 61 73 6B 73 20 2F 43 72 65 61 74 65 20 chtasks /Create
2F 54 4E 20 22 25 73 22 20 2F 54 52 20 22 25 73 /TN "%s" /TR "%s
22 20 2F 53 43 20 4F 4E 4C 4F 47 4F 4E 20 2F 52 " /SC ONLOGON /R
4C 20 48 49 47 48 45 53 54 20 2F 46 00 00 00 52 L HIGHEST /F...R
75 6E 49 6D 61 67 65 00 00 00 00 00 00 00 00 80 unImage.........
ED 1C 01 B4 46 1B 01 1B 47 1B 01 55 6E 6B 6E 6F í..´F...G..Unkno
77 6E 20 65 78 63 65 70 74 69 6F 6E 00 00 00 50 wn exception...P
EE 1C 01 B4 46 1B 01 1B 47 1B 01 62 61 64 20 61 î..´F...G..bad a
6C 6C 6F 63 61 74 69 6F 6E 00 00 C8 ED 1C 01 B4 llocation..Èí..´
46 1B 01 1B 47 1B 01 62 61 64 20 61 72 72 61 79 F...G..bad array
20 6E 65 77 20 6C 65 6E 67 74 68 00 00 00 00 70  new length....p
```

Persistence strings extracted from debugger

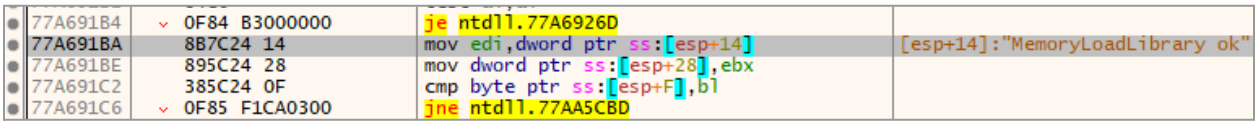
Subsequently, checkpoint tasks are performed, where it is verified if the load has been executed successfully.

```
OutputDebugStringA("RunImage");
sub_4032E7(&v1, &v2);
sub_4032C2(&unk_421030, 25655, &byte_421010);
return sub_40151D(1, v1, v2);
```

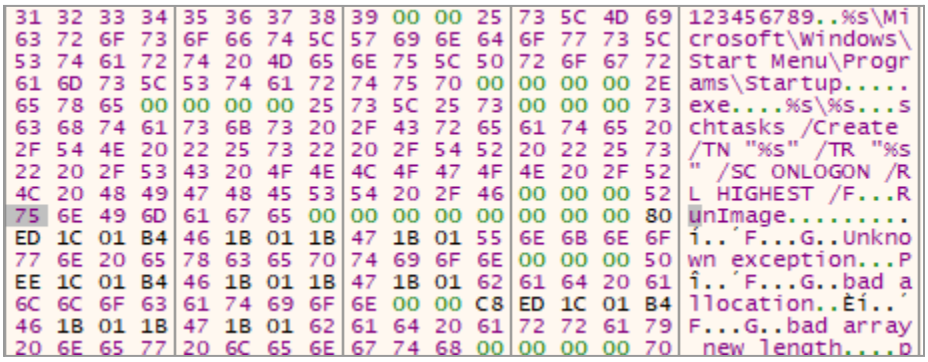
Function shown by IDA

```
v3 = sub_403F75(a2, a3);
if ( !v3 )
{
  OutputDebugStringA("MemoryLoadLibrary failed");
  return 0;
}
OutputDebugStringA("MemoryLoadLibrary ok");
if ( a1 )
{
  if ( (*(_WORD *)(*(_DWORD *)(a2 + 60) + a2 + 22) & 0x2000) != 0 )
  {
    OutputDebugStringA("Launching DLL in separate thread");
    v5 = (_DWORD *)sub_4042C3(4);
    if ( v5 )
      *v5 = v3;
    else
      v5 = 0;
    Thread = CreateThread(0, 0, StartAddress, v5, 0, 0);
    WaitForSingleObject(Thread, 0xFFFFFFFF);
    if ( !Thread )
    {
      OutputDebugStringA("Failed to create entry thread");
      sub_4042F3(v5);
      return 0;
    }
    CloseHandle(Thread);
  }
  else
  {
    OutputDebugStringA("Calling EXE Entry Point");
    sub_403E73(v3);
  }
}
```

Function shown by IDA



Loading a library during debugging

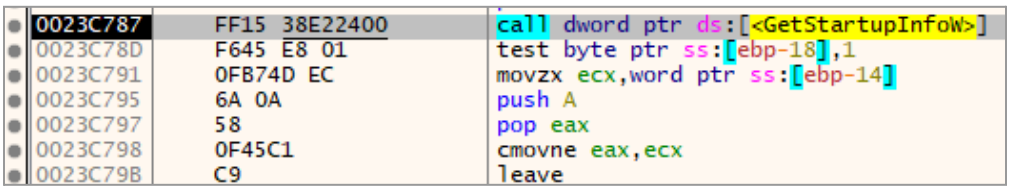


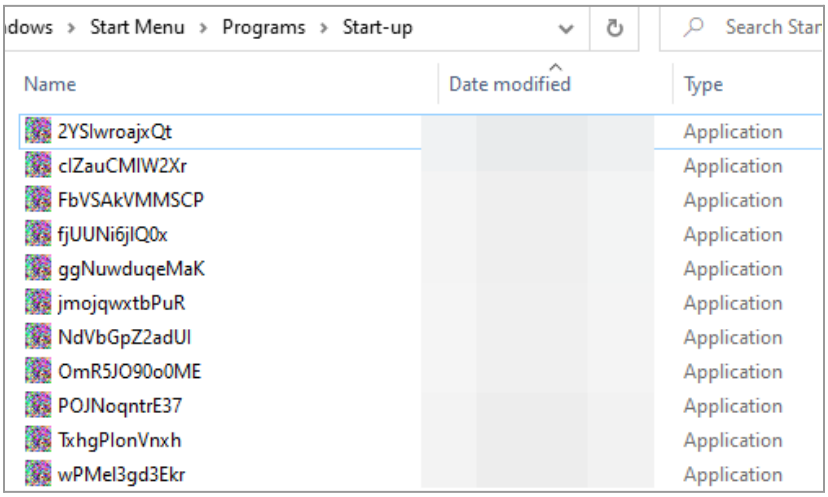
Strings extracted for persistence

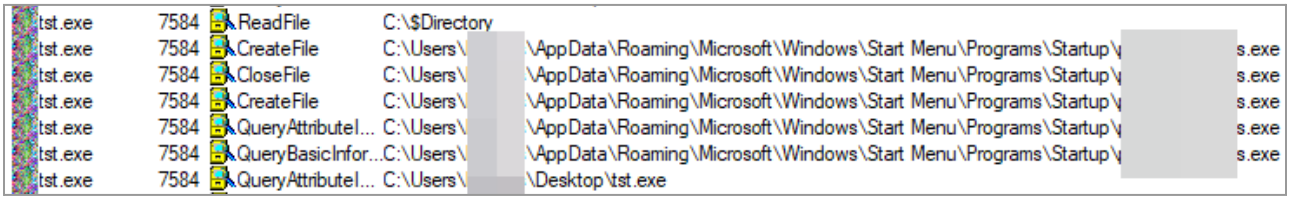
The final result will be a copy of the original binary in the path ***C:\Users\<user>\AppData\Roaming\Windows\Programs\Startup***, using a name that will be created randomly in each iteration, which does not involve any routine to avoid executing twice, stacking in each execution, and following the same pattern.

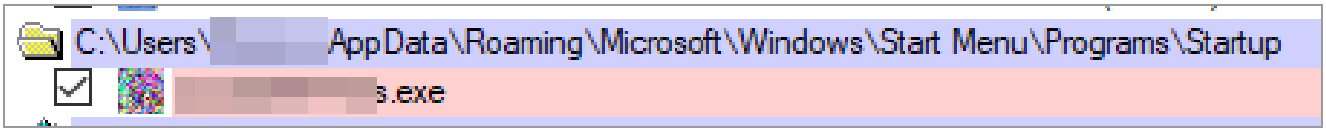
Regex: [a-zA-Z]{11,13}\.exe

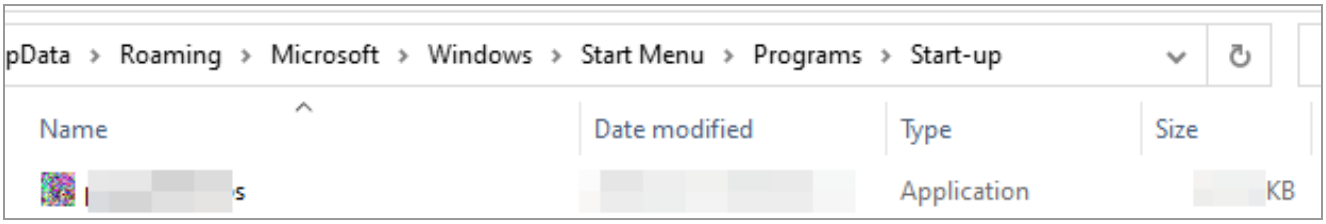
Persistence procedure of Bulwark:











After this, the binary performs different function accesses to deobfuscate and extract the encapsulated binary (Our Malware).

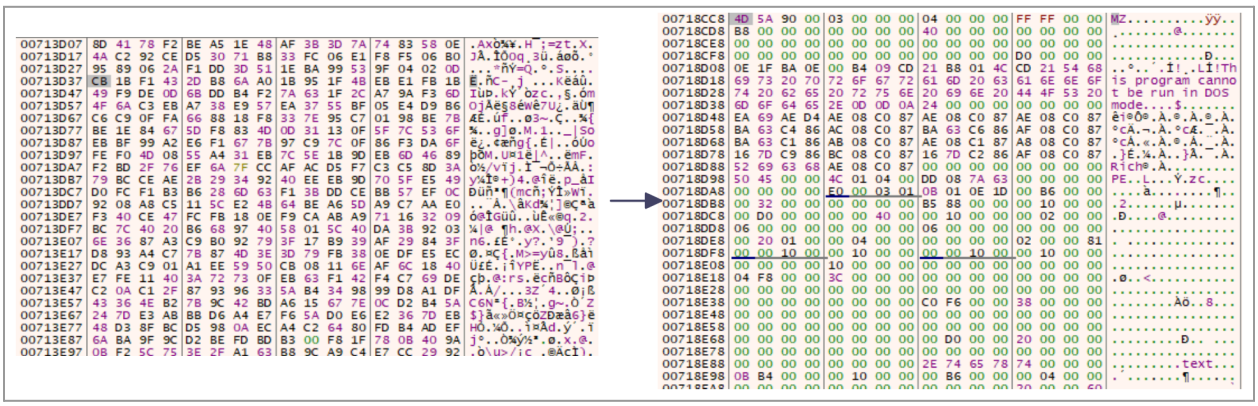
```
00622294C    C1E6 08          shl esi,8
0062294F     0BF0             or esi,eax
00622951     C1E7 08          shl edi,8
00622954     43               inc ebx
00622955     8975 F8          mov dword ptr ss:[ebp-8],esi
00622958     895D FC          mov dword ptr ss:[ebp-4],ebx
0062295B     8BC7             mov eax,edi
0062295D     C1E8 0B          shr eax,B
00622960     0FAFC1           imul eax,ecx
00622963     3BF0             cmp esi,eax
00622965     73 1E            jae 622985
00622967     8BF8             mov edi,eax
00622969     C745 DC 06000000 mov dword ptr ss:[ebp-24],6
00622970     B8 00080000      mov eax,800
00622975     2BC1             sub eax,ecx
00622977     6A 04            push 4
00622979     C1E8 05          shr eax,5
0062297C     0345 BC          add eax,dword ptr ss:[ebp-44]
0062297F     59               pop ecx
00622980     894D D8          mov dword ptr ss:[ebp-28],ecx
00622983     EB 20            jmp 6229A5
00622985     2BF0             sub esi,eax
00622987     C1E9 05          shr ecx,5
0062298A     2BF8             sub edi,eax
0062298C     8975 F8          mov dword ptr ss:[ebp-8],esi
0062298F     8B45 BC          mov eax,dword ptr ss:[ebp-44]
00622992     6A 06            push 6
00622994     2BC1             sub eax,ecx
00622996     C745 DC 07000000 mov dword ptr ss:[ebp-24],7
0062299D     C745 D8 05000000 mov dword ptr ss:[ebp-28],5
006229A4     59               pop ecx
006229A5     8B5D F0          mov ebx,dword ptr ss:[ebp-10]
006229A8     0FB7C0           movzx eax,ax
006229AB     894D B8          mov dword ptr ss:[ebp-48],ecx
006229AE     66:8943 02       mov word ptr ds:[ebx+2],ax
006229B2     8BC3             mov eax,ebx
006229B4     8B5D FC          mov ebx,dword ptr ss:[ebp-4]
006229B7     0FB70401         movzx eax,word ptr ds:[ecx+eax]
006229BB     8945 BC          mov dword ptr ss:[ebp-44],eax
006229BE     0FB7C8           movzx ecx,ax
```

Extraction of the encapsulated binary

Extraction of the encapsulated binary

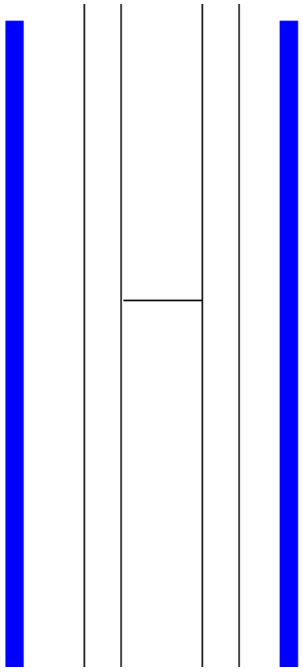
```
00622BD7     81FF 00000001    cmp edi,1000000
00622BDD     73 0C            jae 622BEB
00622BDF     0FB603           movzx eax,byte ptr ds:[ebx]
00622BE2     C1E6 08          shl esi,8
00622BE5     C1E7 08          shl edi,8
00622BE8     0BF0             or esi,eax
00622BEA     43               inc ebx
00622BEB     8B45 9C          mov eax,dword ptr ss:[ebp-64]
00622BEE     81FA 00020000    cmp edx,200
00622BF4     8958 1C          mov dword ptr ds:[eax+1C],ebx
00622BF7     8B5D E8          mov ebx,dword ptr ss:[ebp-18]
00622BFA     8958 18          mov dword ptr ds:[eax+18],ebx
00622BFD     8B5D E4          mov ebx,dword ptr ss:[ebp-1C]
00622C00     8958 28          mov dword ptr ds:[eax+28],ebx
00622C03     8B5D E0          mov ebx,dword ptr ss:[ebp-20]
00622C06     8958 30          mov dword ptr ds:[eax+30],ebx
00622C09     8B5D CC          mov ebx,dword ptr ss:[ebp-34]
00622C0C     8958 34          mov dword ptr ds:[eax+34],ebx
00622C0F     8B5D C8          mov ebx,dword ptr ss:[ebp-38]
00622C12     8958 38          mov dword ptr ds:[eax+38],ebx
00622C15     8B5D C4          mov ebx,dword ptr ss:[ebp-3C]
00622C18     8978 20          mov dword ptr ds:[eax+20],edi
00622C1B     8970 24          mov dword ptr ds:[eax+24],esi
00622C1E     8958 3C          mov dword ptr ds:[eax+3C],ebx
00622C21     5F               pop edi
00622C22     8950 44          mov dword ptr ds:[eax+44],edx
00622C25     8948 40          mov dword ptr ds:[eax+40],ecx
00622C28     1BC0             sbb eax,eax
00622C2A     5E               pop esi
00622C2B     40               inc eax
00622C2C     5B               pop ebx
00622C2D     C9               leave
00622C2E     C2 0400          ret 4
00622C31     8B4D F4          mov ecx,dword ptr ss:[ebp-C]
00622C34     BA 12010000      mov edx,112
00622C39     83E9 0C          sub ecx,C
00622C3C     EB 99            jmp 622BD7
00622C3E     8B55 EC          mov edx,dword ptr ss:[ebp-14]
00622C41     81C2 02020000    add edx,202
00622C47     EB 8E            jmp 622BD7
```

The final result is a second binary extracted from memory.



The final result of the second binary

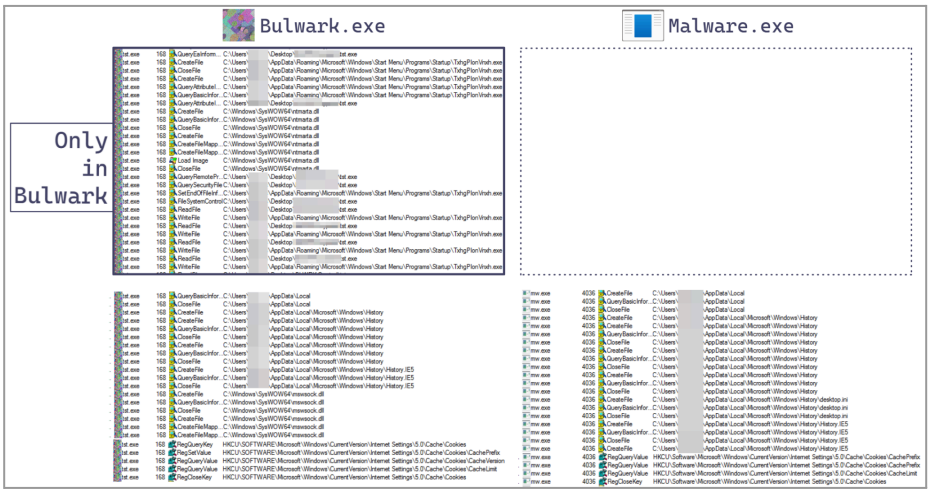
After the diff between the one extracted from memory and the original malware, it is verified that it is identical, so from this phase, the malware that was inside the Bulwark bypasser enters execution.



Comparison of the malware before Bulwark and after extracted from Bulwark (during execution)

The execution of the main malware is not modified by Bulwark in any way, since in telemetry it is observed how the modified executable performs Bulwark functions moments before the binary execution, performing the same functions that were predefined in it, without altering the order or content.

Comparison of the behaviour between Bulwark and the original malware:



Target AV/EDR PoC

Knowing that the modified binary performs the same tasks in a controlled environment, it is necessary to test it against AV and EDR to determine whether it can perform the evasion described by the Threat Actor.

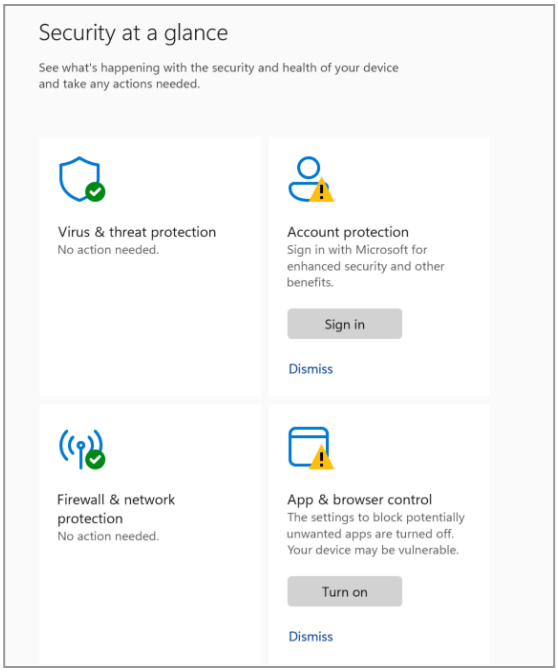
Antivirus (AV)

In the panel of software it is stated that Bulwark theoretically can bypass a large number of antivirus products that are widely used by millions of people worldwide (Kaspersky, Bitdefender, Avast, AVG, Microsoft Defender, etc.)

We carried out tests against some of them to determine whether there was any change in execution or whether these solutions blocked it.

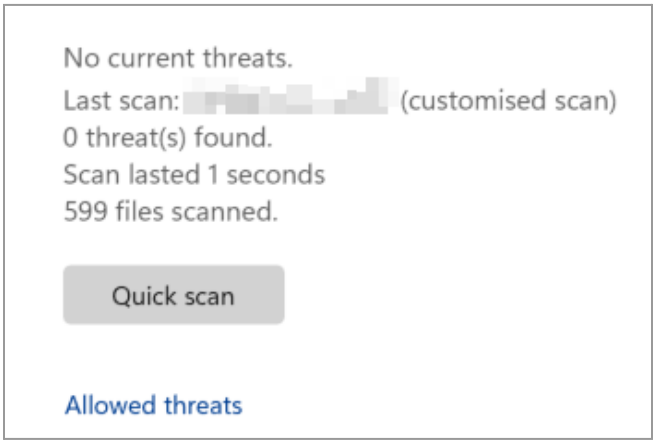
Microsoft Defender

Starting with one of the most used ones, and given that Bulwark is only usable on Windows, the compressed binary is transferred to a device with Microsoft's standard protections.



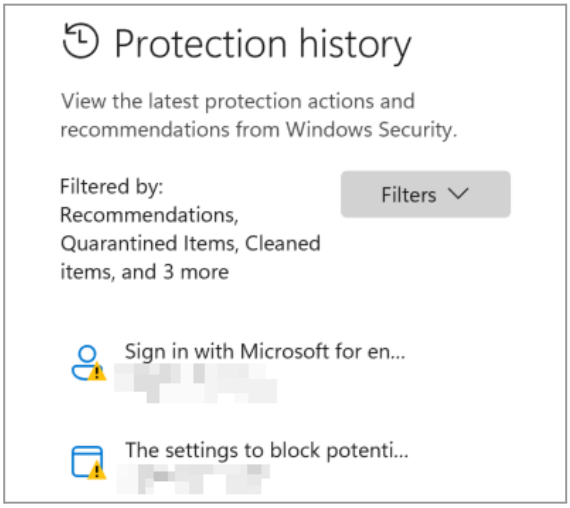
After performing a desktop scan, Windows Defender is not able to determine any threat, so it does not see anything harmful statically in the binary

Icon file after Bulwark



No action from Windows Defender is observed during execution

During execution, no action from Windows Defender is observed either, so it can be determined that, with the modifications made to the binary, the native OS AV is not capable of providing a first line of defense against the threat.

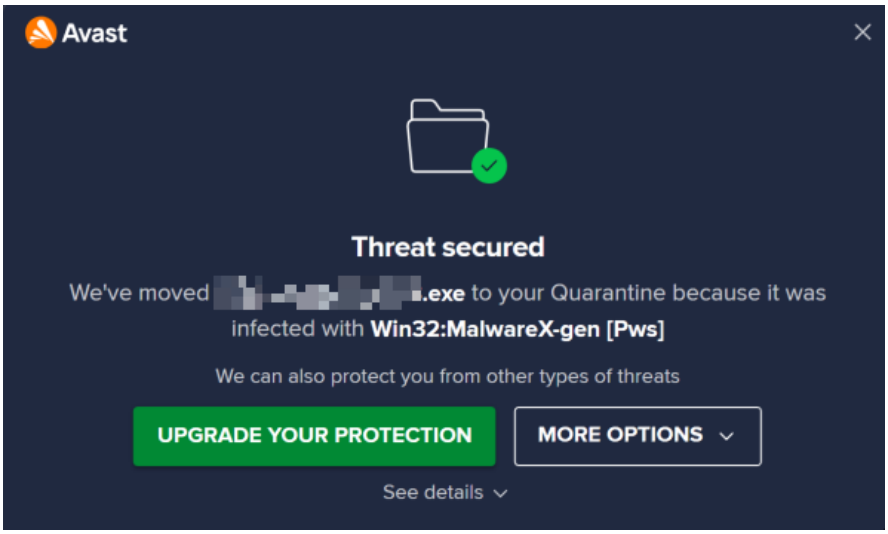


No action from Windows Defender is observed during execution

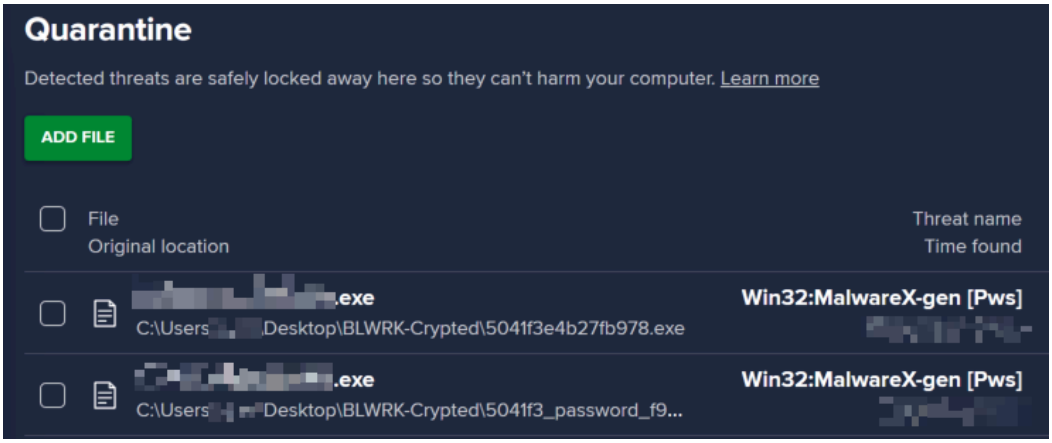
Avast

Among the software that is theoretically bypassable, there are several from GEN, which is the brand that currently encompasses multiple antivirus signatures such as Avast, AVG, Avira, Norton, etc.

Therefore, the same test is performed against one of them, Avast. Extraction of the binary itself is not even allowed in this case, because the AV solution blocks it instantly and sends the binary to quarantine.



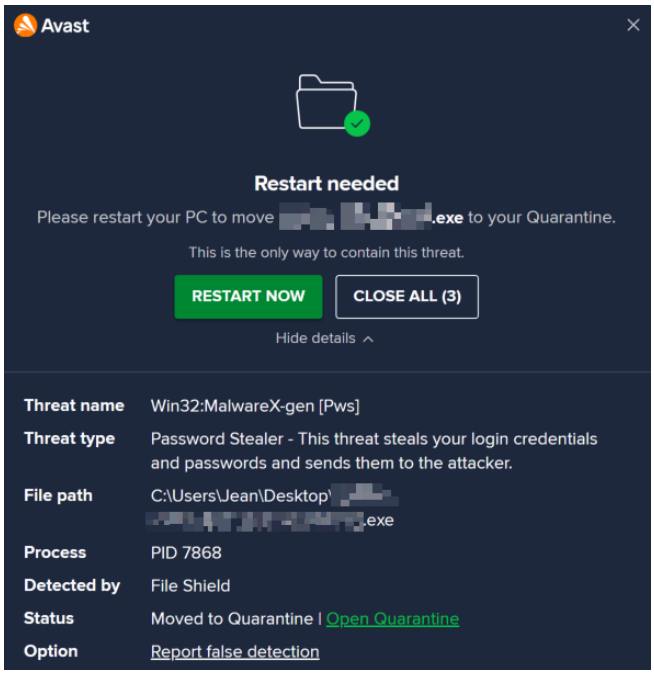
Extraction of the binary itself is not allowed

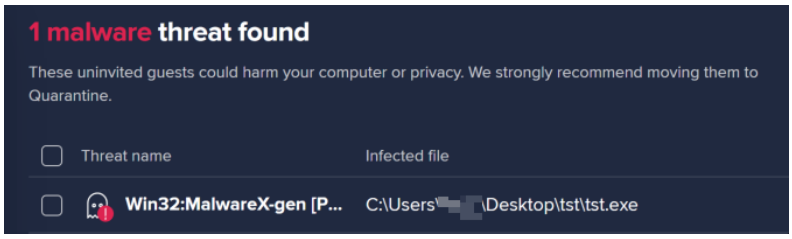


The AV solution blocks it instantly and sends the binary to quarantine

Since it is possible that it was detected contextually or by a signature, an attempt is made to rescue the binary from quarantine to try to execute it and check whether it could reach to the execution step.

After an initial execution, Avast acts on the binary again, preventing its use and successfully stopping the malware:





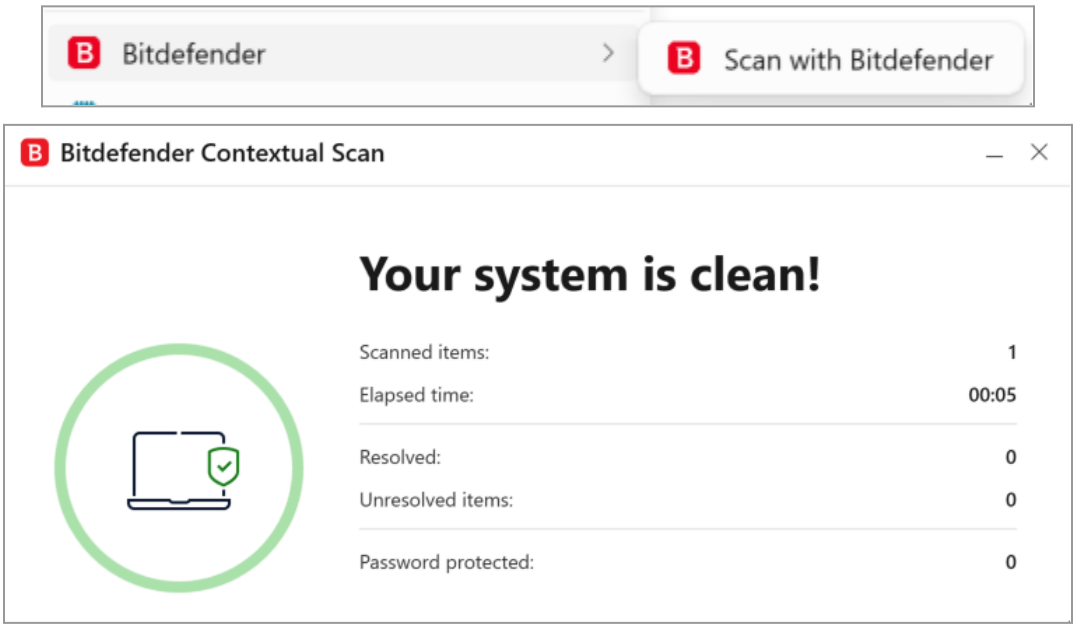
AVG

The same test is carried out with another product from the same GEN group, AVG, and it behaves the same way, being blocked both on decompression of the binary and on its execution.

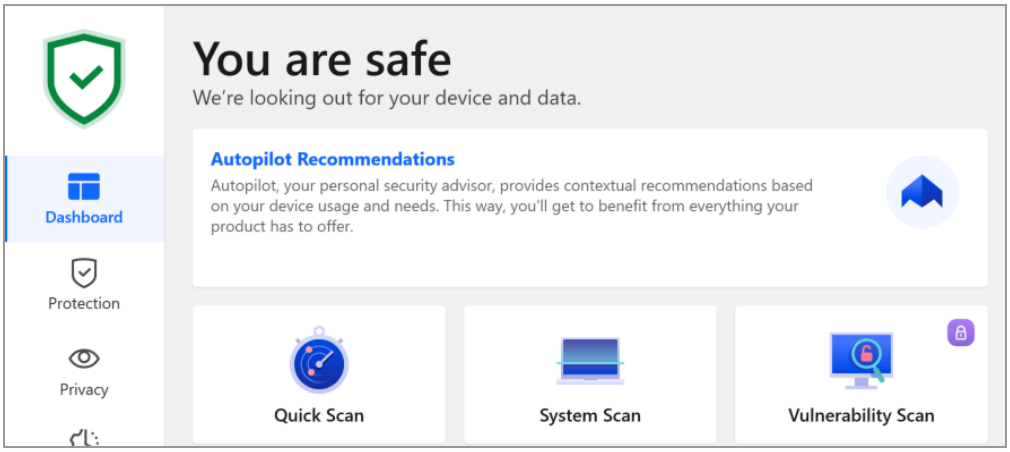
The file being blocked both on decompression of the binary and on its execution

Bitdefender

The same tests are performed in the controlled environment using another antivirus, Bitdefender, and it does not present problems when decompressing or when scanning the sample with its engine:



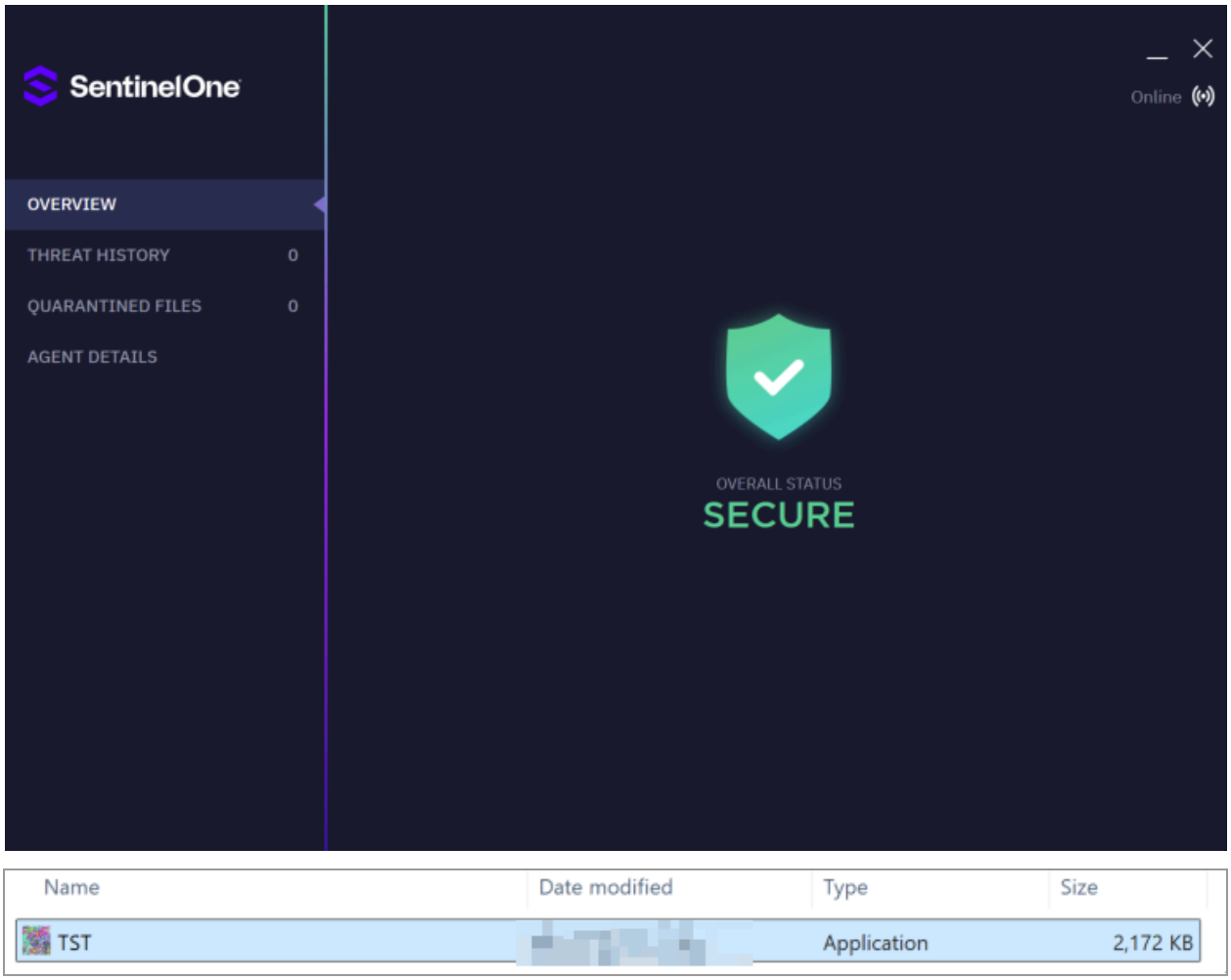
An attempt is made to execute the binary, and it performs its actions normally without alerting or being stopped during execution:



Endpoint Detection & Response (EDR)

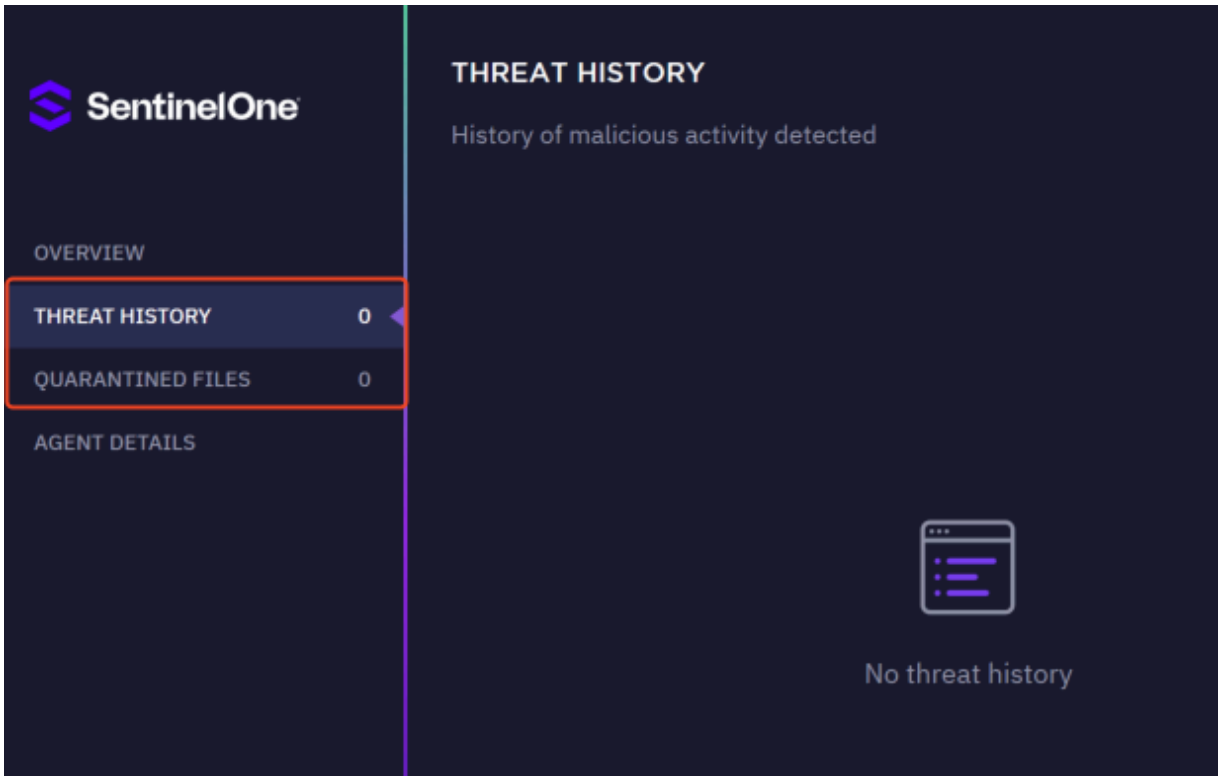
Within the category of solutions that Bulwark is capable of bypassing are EDRs, the software commonly used by companies to protect and monitor infrastructures. SentinelOne was among the products tested by the adversary and was subsequently targeted by the tool.

After an initial extraction, SentinelOne neither detected nor took any action on the binary, so it did not flag anything malicious:



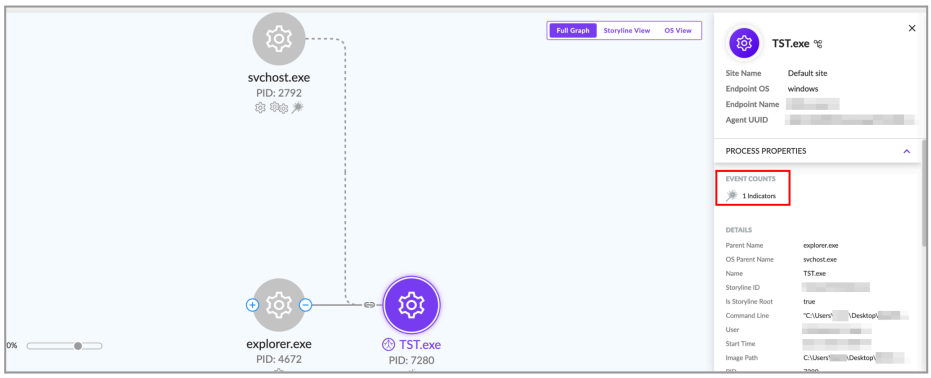
The file was unaffected

After execution, no alerts or actions were raised against the original binary; the binary ran similarly to previous cases where AV products failed to detect Bulwark.



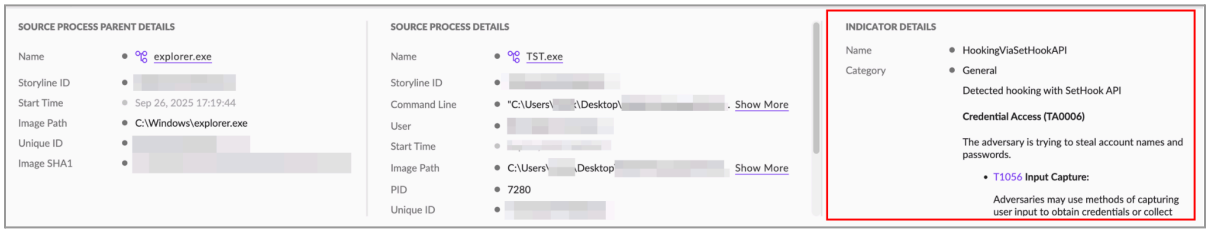
SentinelOne threat history

After reviewing telemetry, the binary's execution is observable, and it does generate an indicator that alerts analysts to suspicious activity.

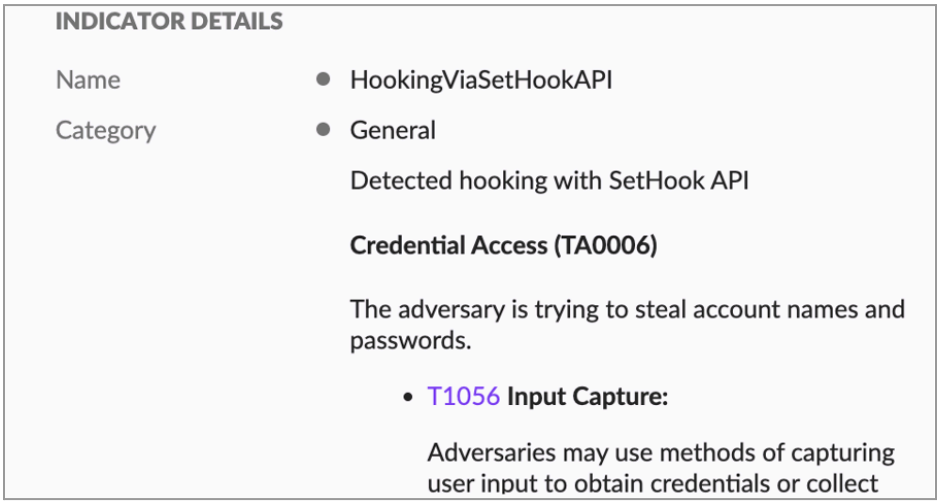


SentinelOne telemetry

Investigating the execution, we found that the related indicator corresponds, based on MITRE, to Credential Access (TA0006), where attempts were made to obtain device credentials using techniques such as Input Capture (T1056), a capability that characterizes the binary and is present in its feature set.

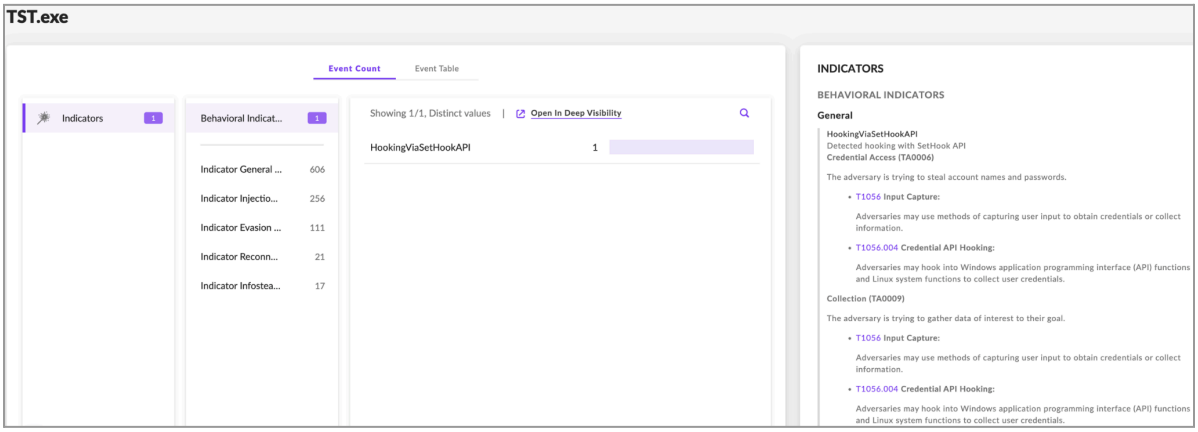


Indicator Details from SentinelOne



Indicator Details from SentinelOne

Examining the binary further, those indicators expand: a SetHook API was detected, which is commonly used to obtain system information or to collect sensitive data. This is tied into various credential access and data-collection techniques.



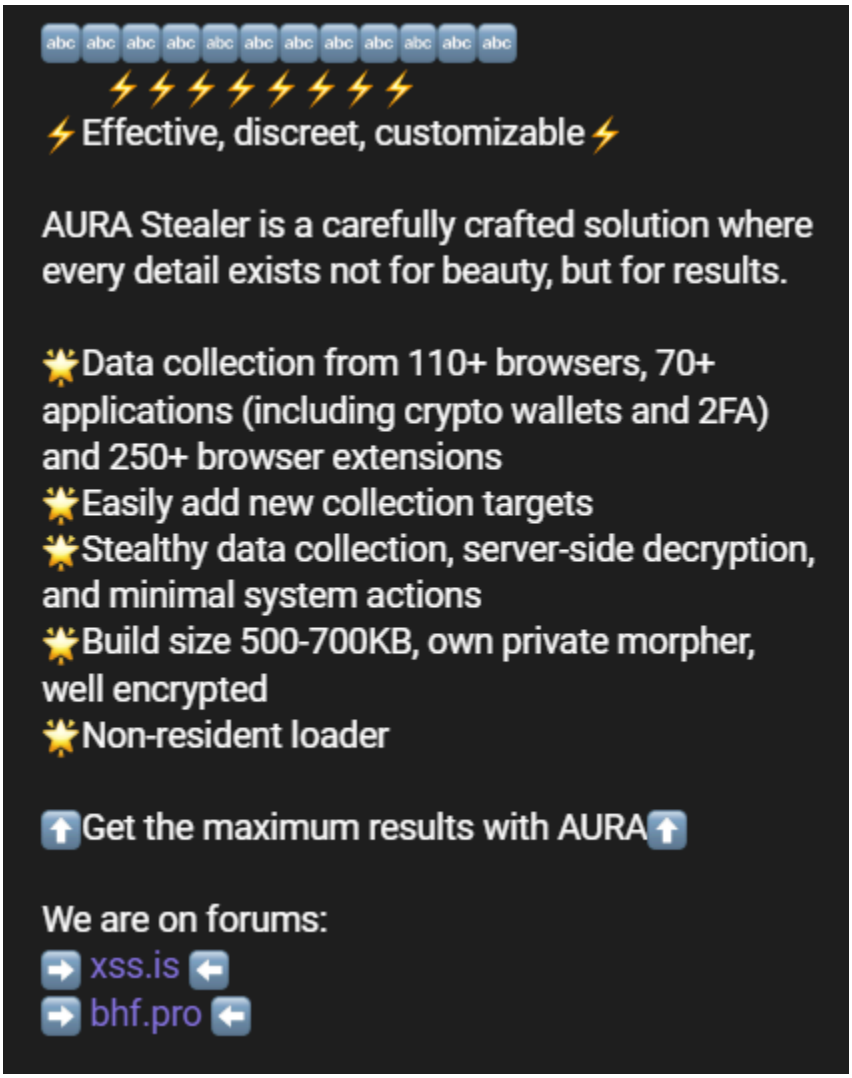
Indicators from SentinelOne

Thus, the malware did execute, but it did not complete its full lifecycle, because the EDR intervened at later stages, disrupting many of its capabilities and alerting analysts in the console. Bulwark was only partially effective in this instance.

Aura Stealer

Aura is another tool accessible through the database, which is a recent Stealer that appeared in July, like the rest of the correlated tools from 2025.

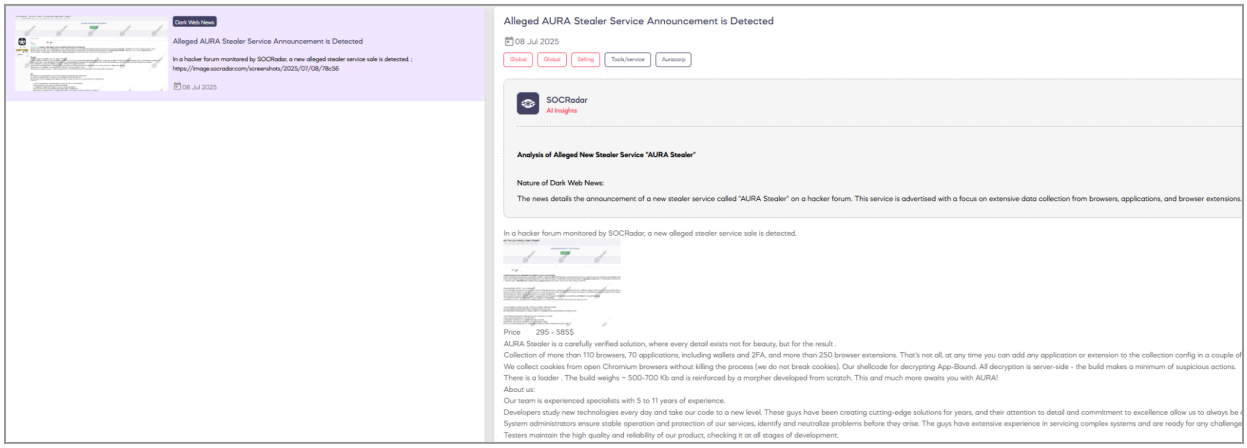
It promises capabilities similar to other stealers, being customizable through a builder and capable of obtaining information from browsers and other applications to subsequently exfiltrate them to the affiliates' C&C.

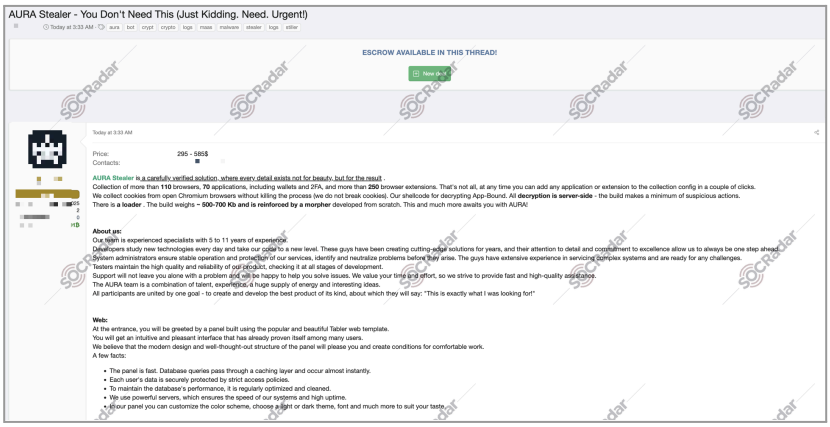


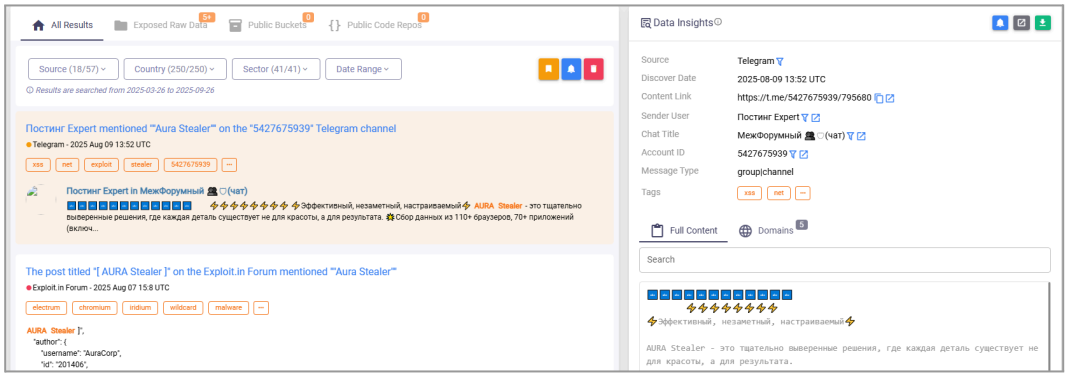
Screenshot of the Telegram advertisement

Aura advertises on both Telegram and Dark Web forums to promote itself, following a similar model to the other affiliated companies.

Promotional Content on Dark Web and Telegram sources, obtained from SOCRadar Dark Web Module:



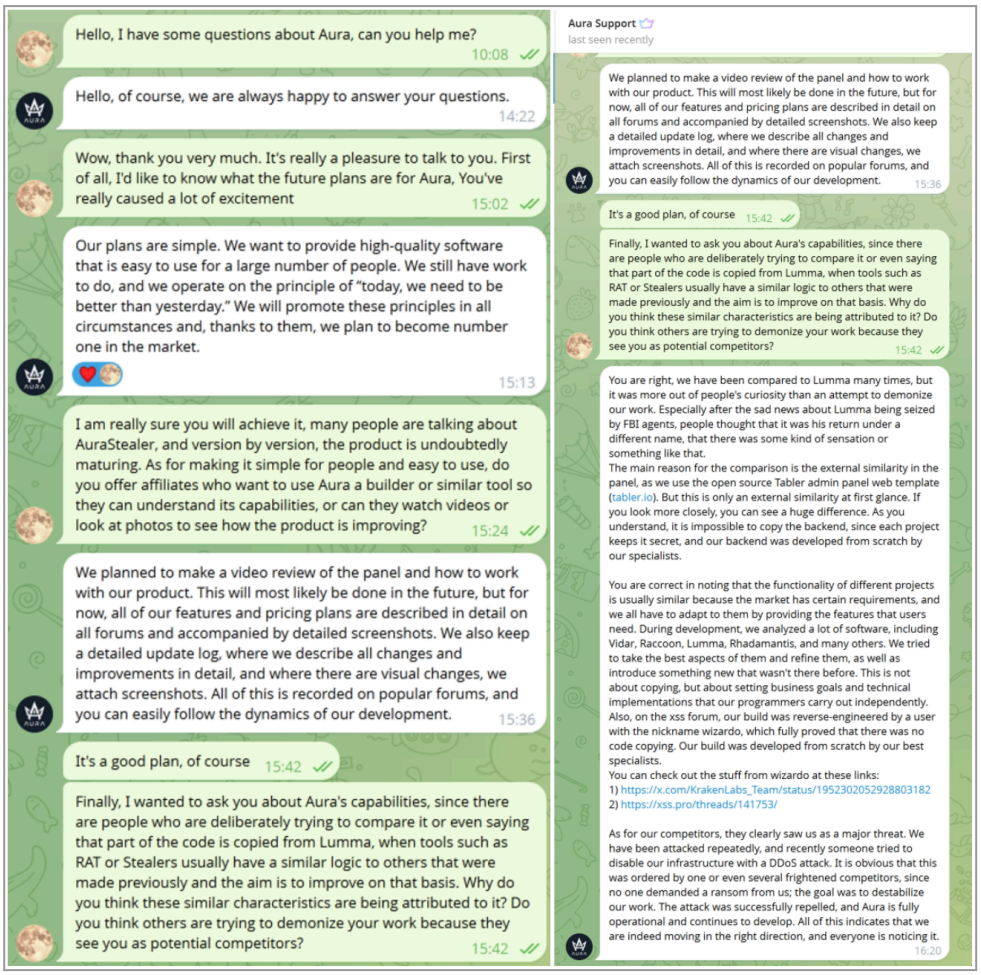




Interview

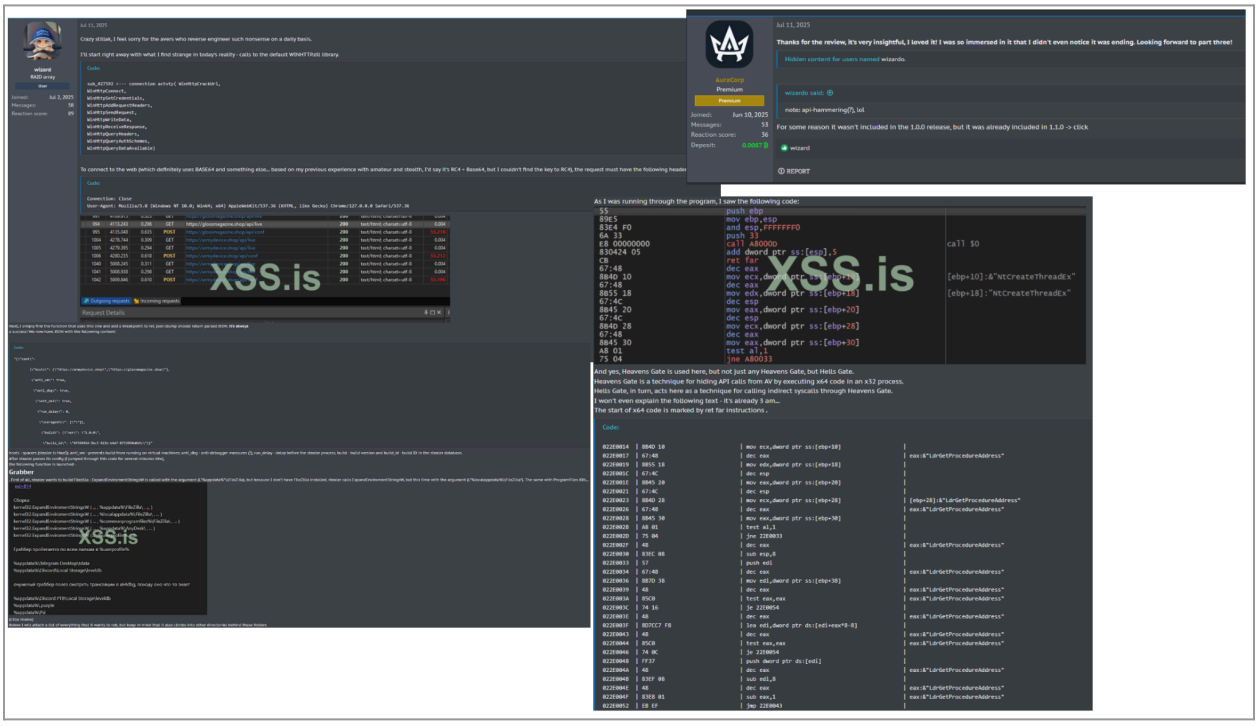
During this interview, the TA was very open and collaborative, and pertinent questions could be asked to learn more about the Stealer. These questions were focused on clarifying different aspects:

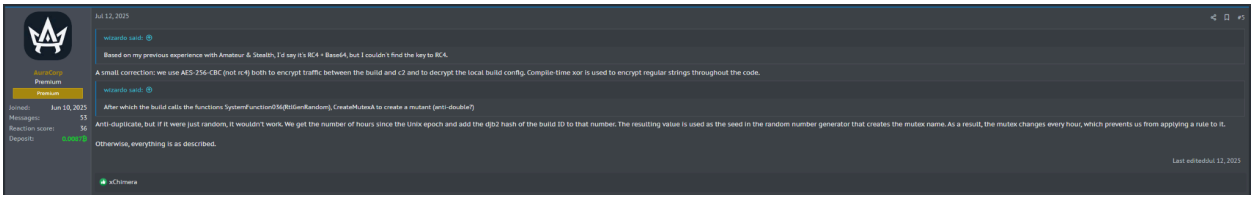
- If it were possible to obtain photos, videos, or a demo of Aura Stealer
- Extended capabilities and what differentiated it from other Stealers
- Relationships with other similar products
- Controversy with Lumma Stealer and possible code copying



Interview screenshot

As the TA mentioned, communication takes place with a user called "wizard", who analyzes the binary to determine some of the techniques, even finding some flaws or improvements that could be made, as well as resolving doubts about the Stealer. At the end of the conversation, the TA himself gives him $100 for the work performed.





Posts from Dark Web forums about the malicious product

During this analysis, clear differences from Lumma are shown, as the Threat Actor commented during the interview.

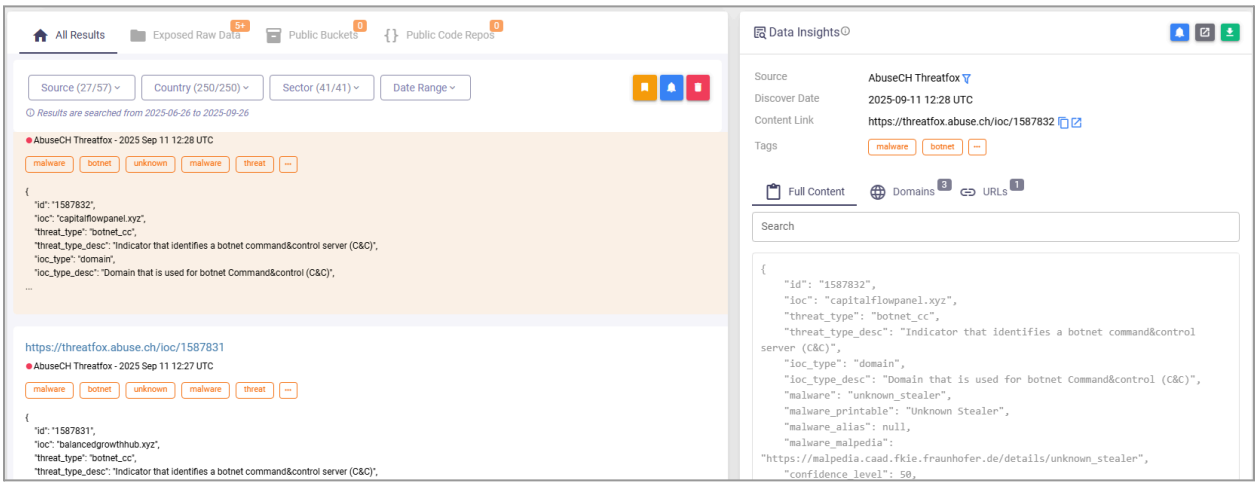
Features

Among Aura Stealer's capabilities, similarity to other stealers can be expected, both from Lumma, previously mentioned, and from others where the TA himself has commented on his inspiration for Aura's development, such as Vidar, Rhadamanthys, or Raccoon.

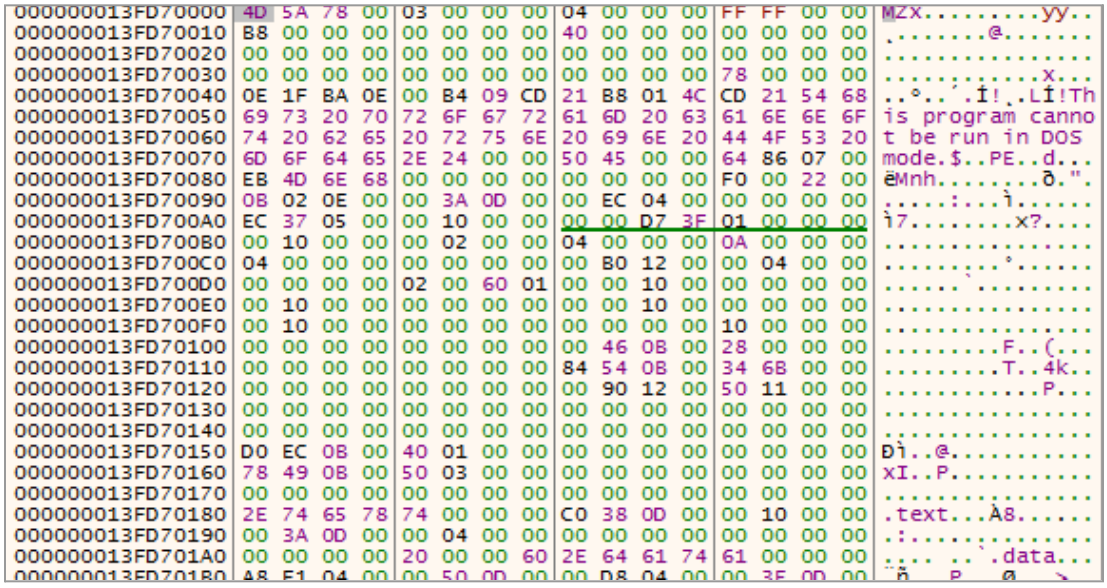
From these capabilities, we can highlight some such as:

- Password and wallet checking
- Cookie and credential theft
- Clipboard information retrieval
- Password collection in txt format (AllPasswords.txt)
- Screenshots
- Device/User info
- Communication with Telegram bots

During collection tasks, the SOCRadar platform is used to obtain valid IOCs from various sources.

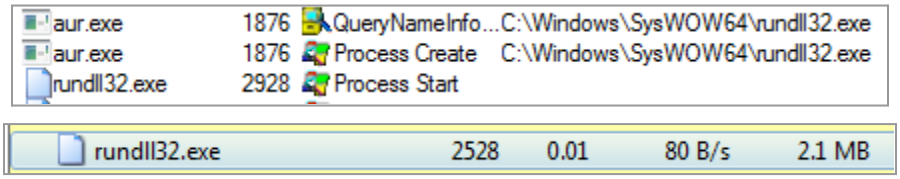


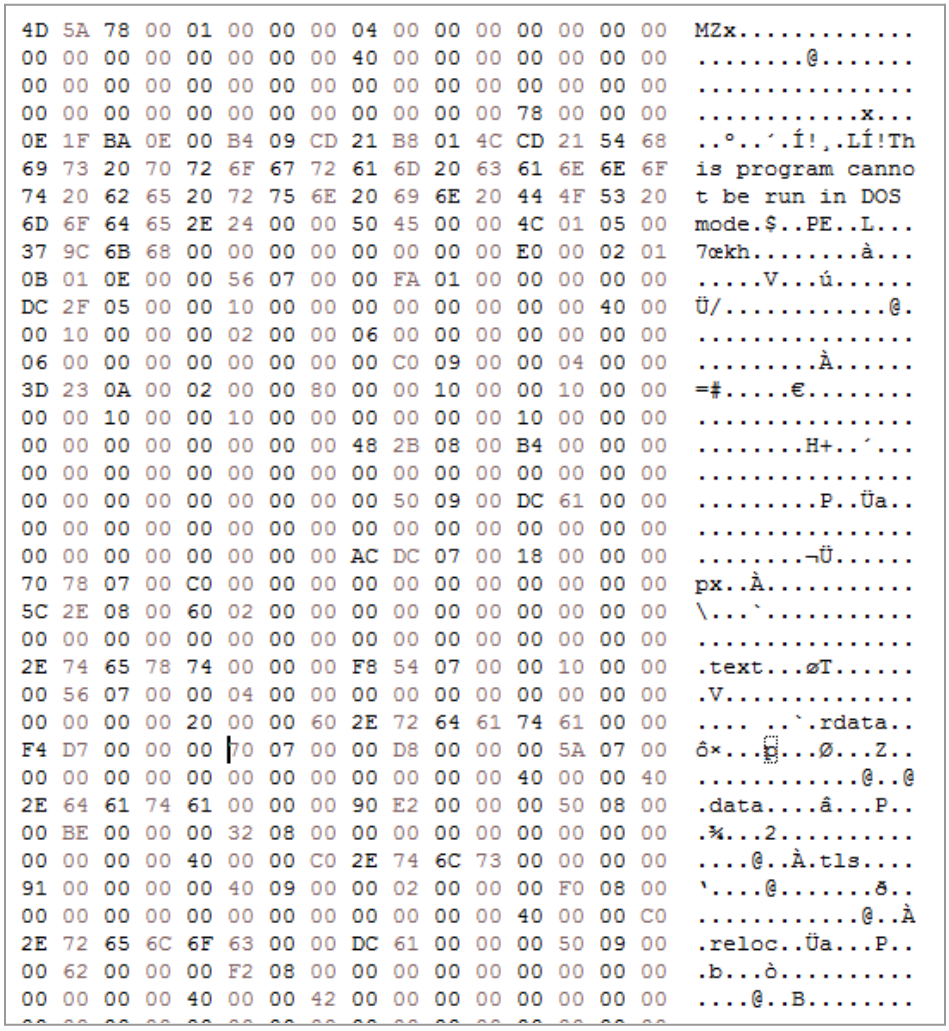
After collecting IOCs, we checked how it behaves in execution; it commonly seeks another process to perform injection with Aura Stealer capabilities.



Extraction of the binary of Aura Stealer

rundll32.exe process injection attempt:

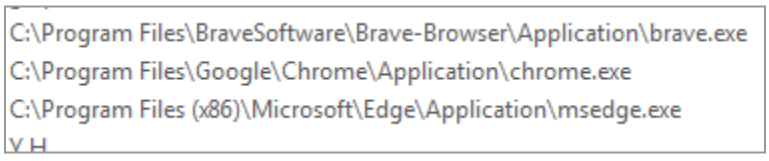




```
4D 5A 78 00 01 00 00 00 04 00 00 00 00 00 00 00  MZx.............
00 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00  ........@.......
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 78 00 00 00  ............x...
0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68  ..°..´.Í!¸.LÍ!Th
69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F  is program canno
74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20  t be run in DOS
6D 6F 64 65 2E 24 00 00 50 45 00 00 4C 01 05 00  mode.$..PE..L...
37 9C 6B 68 00 00 00 00 00 00 00 00 E0 00 02 01  7œkh........à...
0B 01 0E 00 00 56 07 00 00 FA 01 00 00 00 00 00  .....V...ú......
DC 2F 05 00 00 10 00 00 00 00 00 00 00 00 40 00  Ü/............@.
00 10 00 00 00 02 00 00 06 00 00 00 00 00 00 00  ................
06 00 00 00 00 00 00 00 00 C0 09 00 00 04 00 00  .........À......
3D 23 0A 00 02 00 00 80 00 00 10 00 00 10 00 00  =#.....€........
00 00 10 00 00 10 00 00 00 00 00 00 10 00 00 00  ................
00 00 00 00 00 00 00 00 48 2B 08 00 B4 00 00 00  ........H+..´...
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 50 09 00 DC 61 00 00  .........P..Üa..
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 AC DC 07 00 18 00 00 00  ........¬Ü......
70 78 07 00 C0 00 00 00 00 00 00 00 00 00 00 00  px..À...........
5C 2E 08 00 60 02 00 00 00 00 00 00 00 00 00 00  \...`...........
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
2E 74 65 78 74 00 00 00 F8 54 07 00 00 10 00 00  .text...øT......
00 56 07 00 00 04 00 00 00 00 00 00 00 00 00 00  .V..............
00 00 00 00 20 00 00 60 2E 72 64 61 74 61 00 00  .... ..`.rdata..
F4 D7 00 00 00 70 07 00 00 D8 00 00 00 5A 07 00  ô×...p...Ø...Z..
00 00 00 00 00 00 00 00 00 00 00 00 40 00 00 40  ............@..@
2E 64 61 74 61 00 00 00 90 E2 00 00 00 50 08 00  .data....â...P..
00 BE 00 00 00 32 08 00 00 00 00 00 00 00 00 00  .¾...2..........
00 00 00 00 40 00 00 C0 2E 74 6C 73 00 00 00 00  ....@..À.tls....
91 00 00 00 00 40 09 00 00 02 00 00 00 F0 08 00  '....@.......ð..
00 00 00 00 00 00 00 00 00 00 00 00 40 00 00 C0  ............@..À
2E 72 65 6C 6F 63 00 00 DC 61 00 00 00 50 09 00  .reloc..Üa...P..
00 62 00 00 00 F2 08 00 00 00 00 00 00 00 00 00  .b...ò..........
00 00 00 00 40 00 00 42 00 00 00 00 00 00 00 00  ....@..B........
```

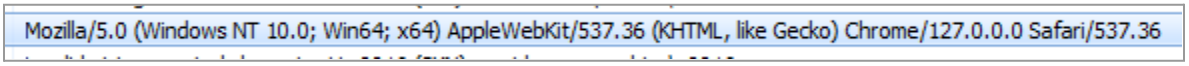
Extracted Aura Stealer binary

In these executions, browser searches can be observed, as well as attempts at external requests to maintain communications with the C2.



```
C:\Program Files\BraveSoftware\Brave-Browser\Application\brave.exe
C:\Program Files\Google\Chrome\Application\chrome.exe
C:\Program Files (x86)\Microsoft\Edge\Application\msedge.exe
```

Browser searches conducted by Aura Stealer

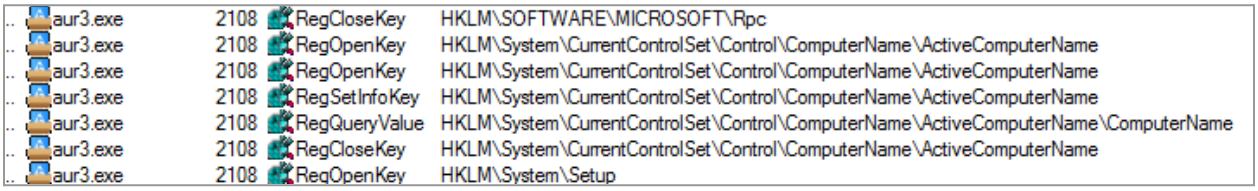


Connection attempts to browser agents

Examples:

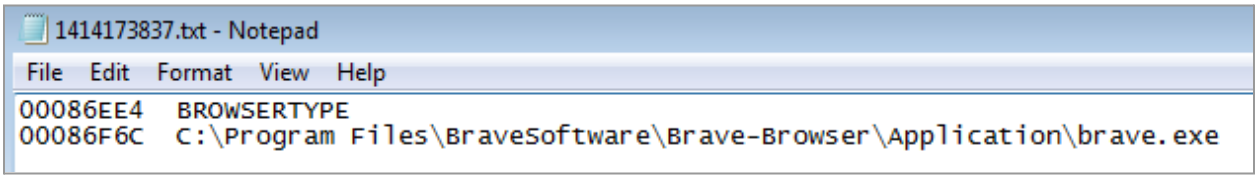
```
C:\users\<user>\AppData\Roaming\Opera Software\Opera Stable\Local State
C:\users\<user>\AppData\Local\Microsoft\Edge\User Data\Login Data
C:\users\<user>\AppData\Local\CocCoc\Browser\User Data\Local State
```

Likewise, all versions collect device information necessary to maintain communication with Telegram and send a record of each infected system's characteristics to the panel.

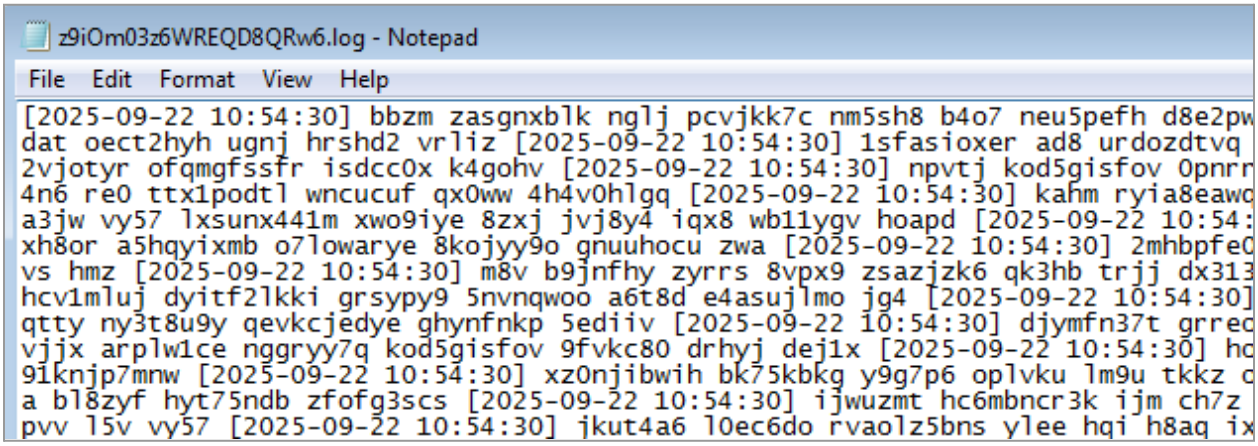


Query attempts to retrieve information from registry

Similarly, there are less refined, earlier versions or those that have served as tests where they are somewhat noisier and save information about where they have collected information or hardcoded logs with obtained information.



Collected information

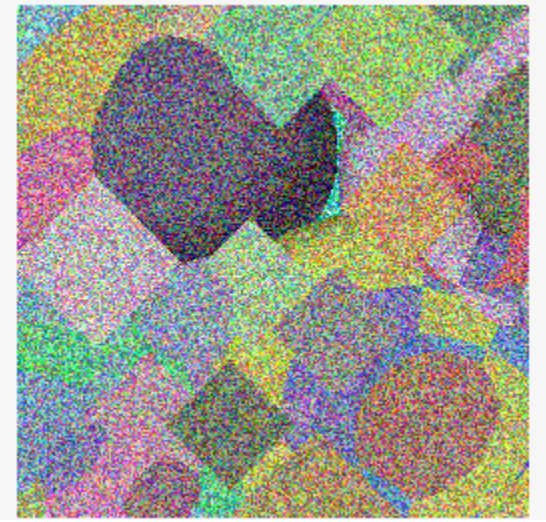


Collected information

The adversary shows images of information collection from devices affected by Aura where passwords, screenshots, previously seen information (Browsers, system information...), etc., can be collected.

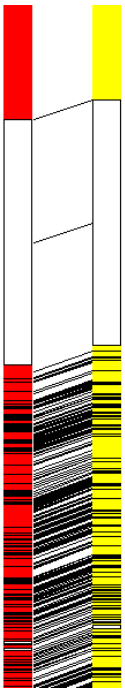
Detailed showcase of collected information

Versions of Aura with Bulwark as an added feature are also located, which makes analysis difficult but keeps their capabilities intact.



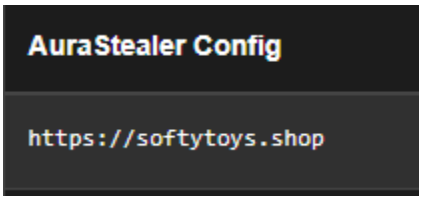
Aura Stealer's icon after Bulwark

Comparing the Bulwark test binaries with the modified Aura Stealer ones, great similarity between them is appreciated; however, the execution result is different.



Comparison of Aura Stealer before and after Bulwark

The C&C addresses are varied, using different ".stop" and ".shop" panels:

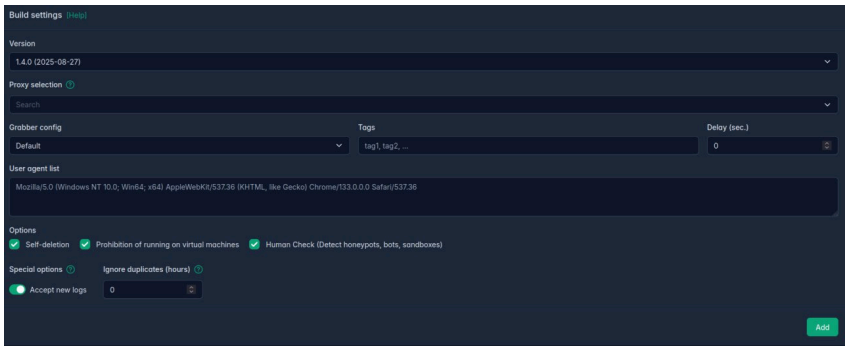


Builder

AuraStealer presents a Builder, like most similar malware, where different changes can be made to the final binary, so the affiliate who buys Aura can have some decision-making power over the final executable, being able to add or remove features from it.

This Builder is still in development and doesn't have as many options as other competitors, a sign of its short life, where it can be observed that changes can be made to:

- Proxy
- Domain or address where it will connect
- User agent used (By default, the one seen during analysis is found)
- Auto-deletion of the original binary
- Anti-VM tactics
- Anti-Sandbox, Anti-HoneyPot, and other anti-analysis techniques
- Log creation



Screenshot of the builder

Pricing

As stated in underground forums and their Telegram accounts, Aura Stealer has a model similar to its competitors, with rates depending on the capabilities you want. It allows the user to create different Stealers. The two main subscription models presented by the TA are:

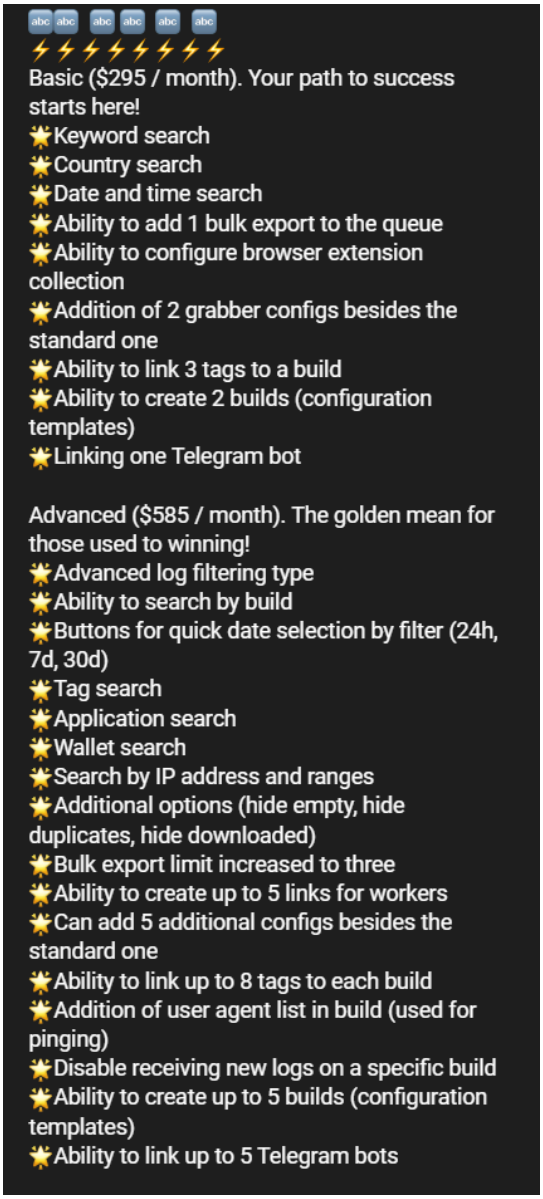
Basic - $295/month

- Keyword search
- Country search
- Date and time search
- Ability to add 1 bulk export to the queue
- Ability to configure browser extension collection
- Addition of 2 grabber configs besides the standard one
- Ability to link 3 tags to a build
- Ability to create 2 builds (configuration templates)
- Linking one Telegram bot

Advanced - $585/month

- Advanced log filtering type
- Ability to search by build
- Buttons for quick date selection by filter (24h, 7d, 30d)
- Tag search
- Application search
- Wallet search
- Search by IP address and ranges
- Additional options (hide empty, hide duplicates, hide downloaded)
- Bulk export limit increased to three
- Ability to create up to 5 links for workers
- Can add 5 additional configs besides the standard one
- Ability to link up to 8 tags to each build
- Addition of the user agent list in the build (used for pinging)
- Disable receiving new logs on a specific build
- Ability to create up to 5 builds (configuration templates)
- Ability to link up to 5 Telegram bots

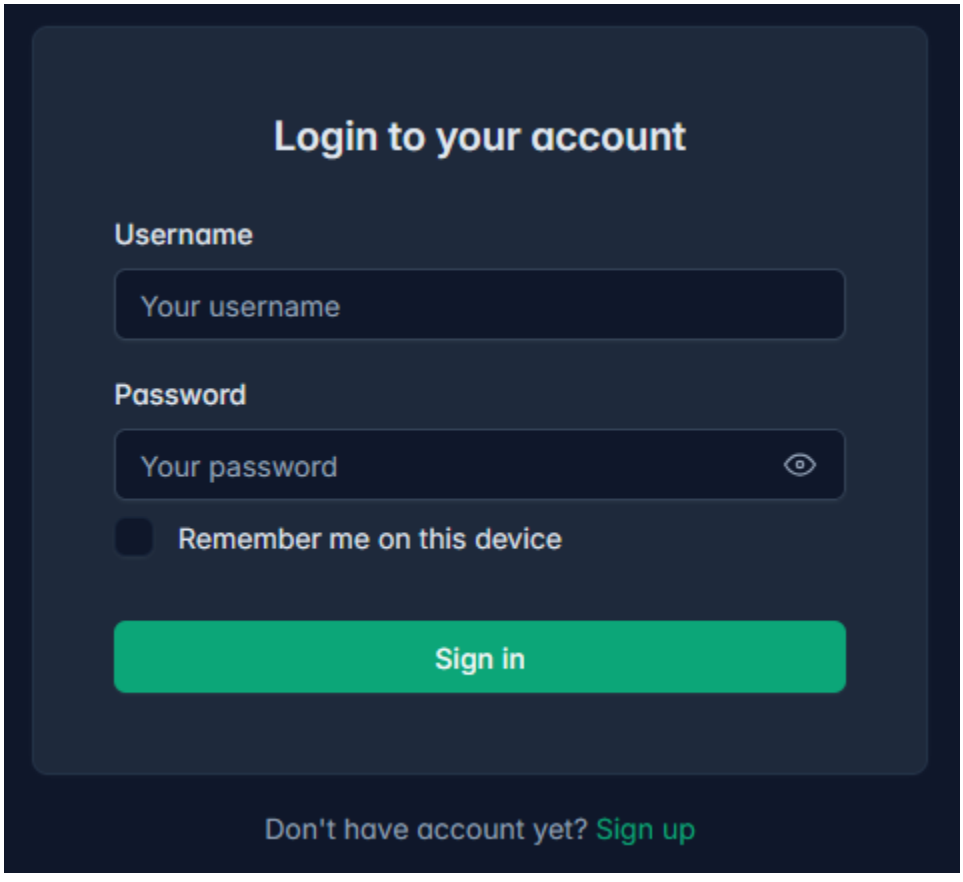
Pricing message:



Panel and communications

The developers have prepared a panel on tabler.io, to which they report information from affected devices, with a user system typically based on affiliates who acquire access to the platform being common.

In the publicly available information about Aura Stealer, different panels are located where C2 communication takes place both from the actor and affiliates, and where they can log in to access their panel.

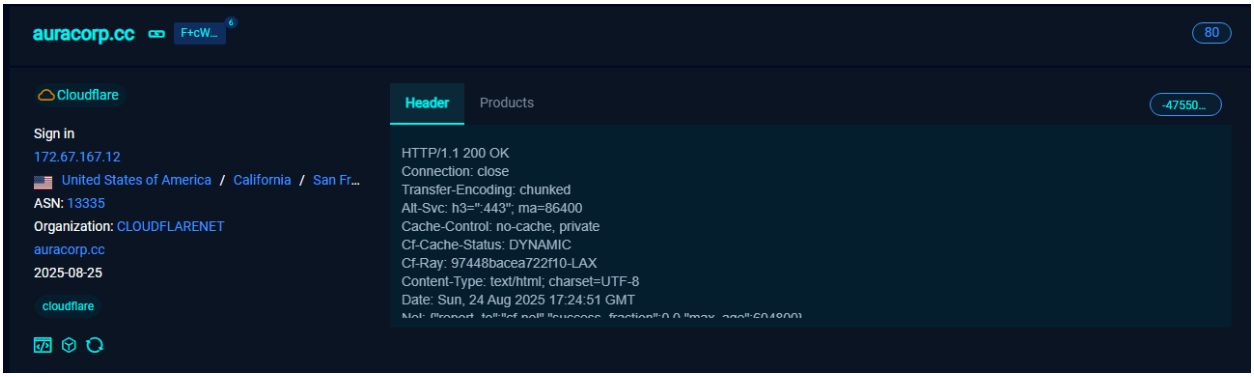


Login page

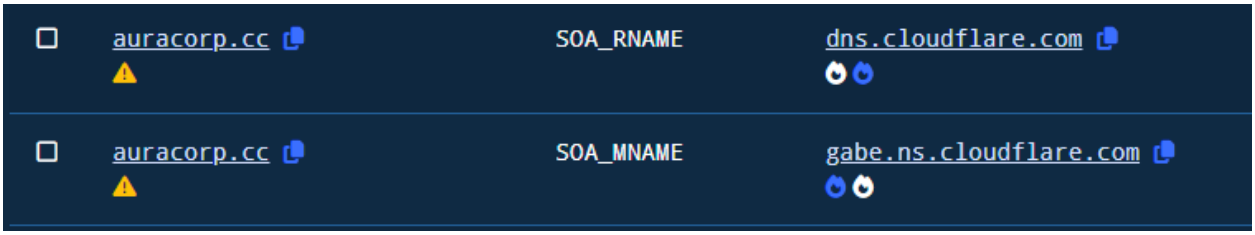
Following similar patterns, attempts are made to reach other domains that may belong to the same infrastructure, where various aspects may coincide:

They are behind Cloudflare
They use TLDs like .shop, .xyz and .cc
ASN: 13335

All domains used have been under Cloudflare, making pivoting difficult on many parameters.



Example of pivoting using FOFA search engine



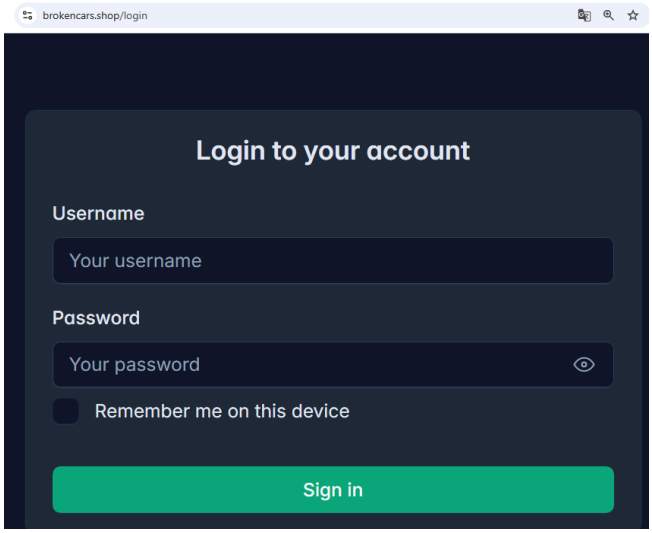
Example of pivoting using Validin

Through platforms like URLscan, similar domains are reached that share the same infrastructure and resemble each other in various aspects, potentially leading to other domains that are still active.

opencamping.shop/login	3 days
brokencars.shop/login	7 days
radioengineering.shop/login	15 days
luxgames.shop/login	15 days
cartdetails.shop/login	15 days
softytoys.shop/login	25 days
glossmagazine.shop/login	2 months
mail.sipkopnas.id	2 months
armydevice.shop/login	2 months
www.sipkopnas.devportation.com	2 months
secondhandcloth.shop/login	2 months
ps.525522.xyz	6 months

URLScan Results

Example of pivoting using FOFA search engine



Login page

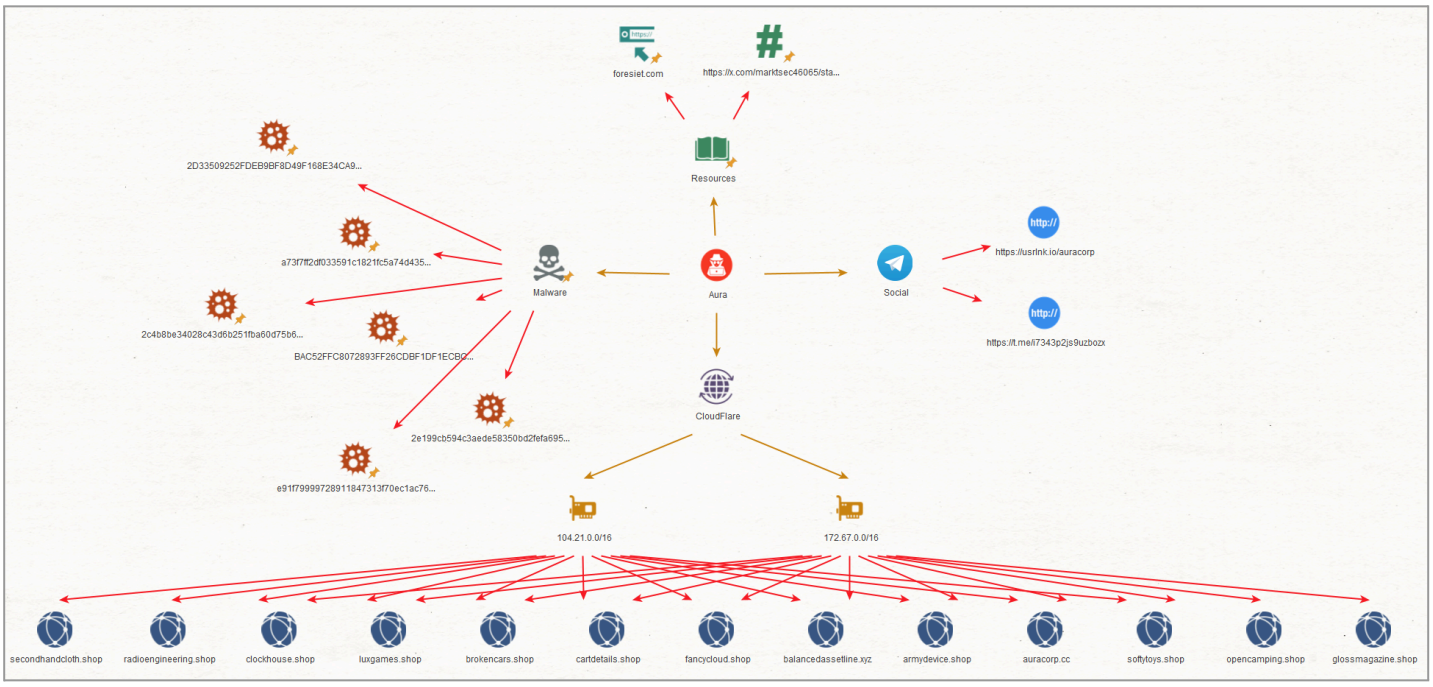
Some of them are either no longer active, having started to appear this type of addresses in June-July 2025, or on the contrary have already been reported and/or have been detected as potential malicious domains.

One of the reported malicious domains

During this analysis, various active and inactive domains with the same infrastructure are reached.

softytoys[.]shop
auracorp[.]cc
secondhandcloth[.]shop
armydevice[.]shop
opencamping[.]shop
glossmagazine[.]shop
balancedassetline[.]xyz
radioengineering[.]shop
secondhandcloth[.]shop
luxgames[.]shop
cartdetails[.]shop
softytoys[.]shop
glossmagazine[.]shop
armydevice[.]shop
fancycloud[.]shop
clockhouse[.]shop

After the analysis, all information can be consolidated in a graphic format to correlate all adversary indicators.

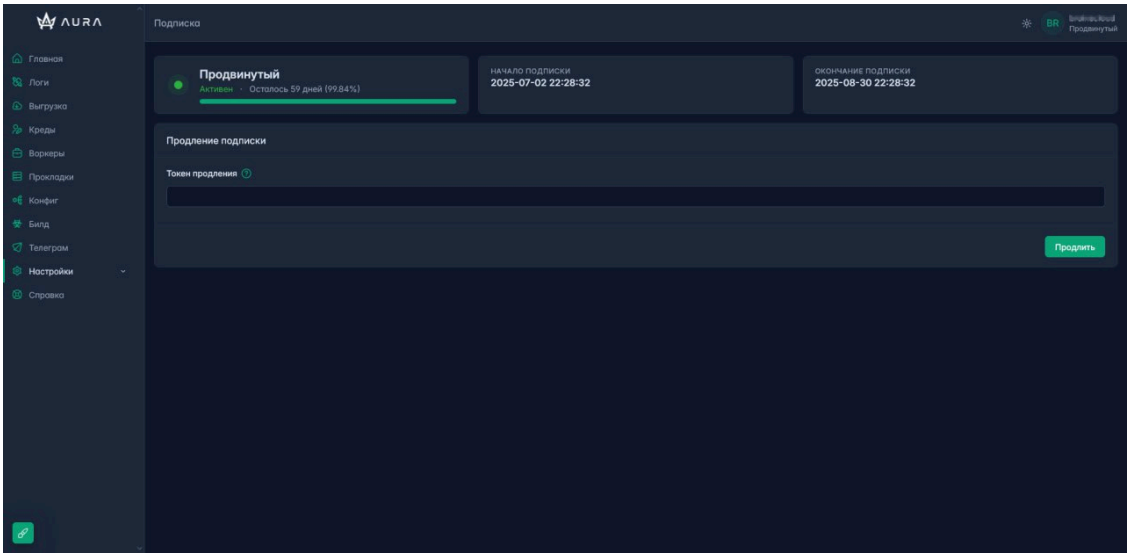


Correlation of all adversary indicators

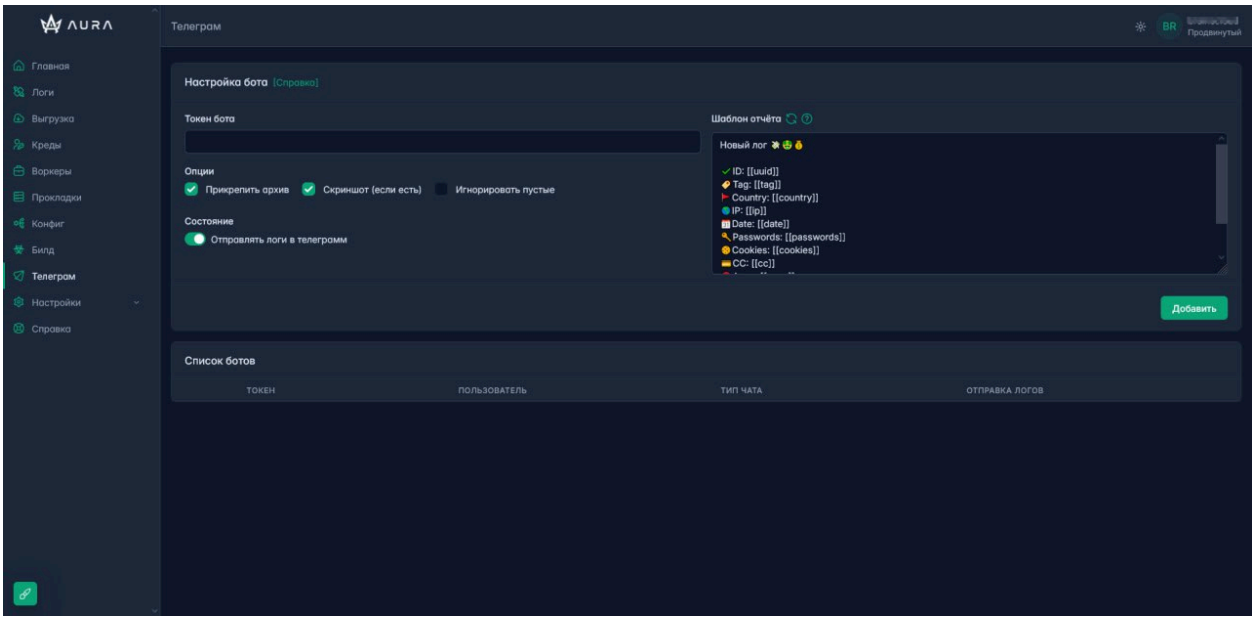
What we can visualize as an affiliate or the adversary themselves in the panel is access to information related to deployed malware, where searches or queries can be performed by regions or countries, a common practice where we can easily access data from each Stealer victim.

Aura Stealer dashboard

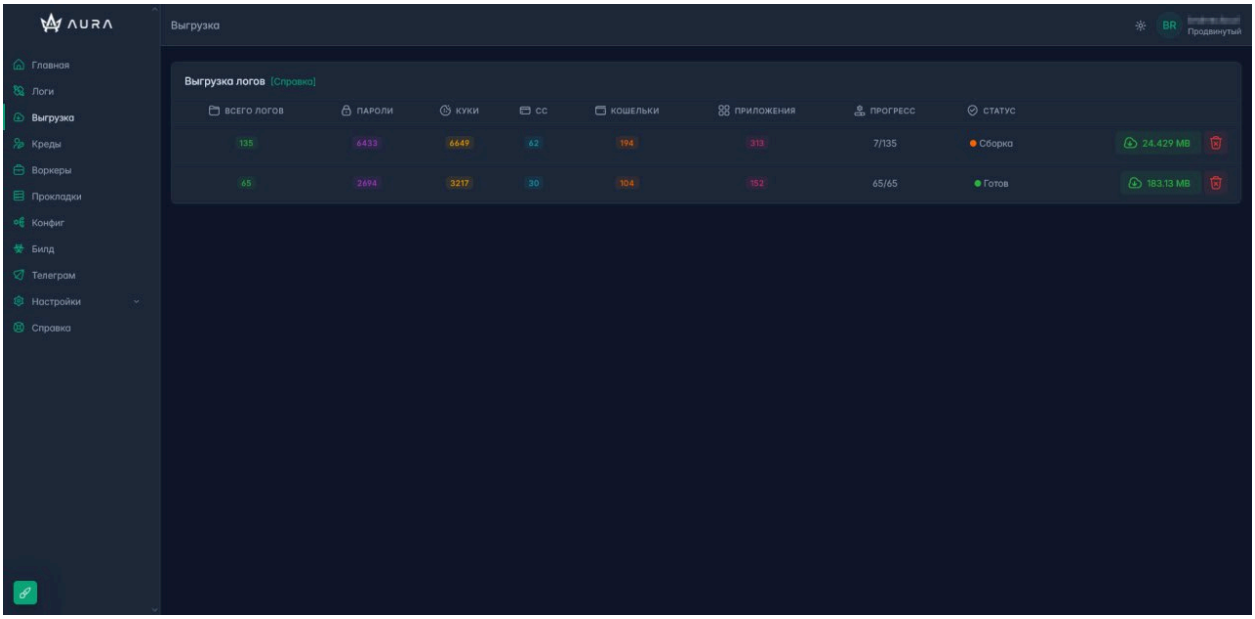
From this control panel, freedom is given to modify some parameters or how information will be received in the panel, providing granularity and adaptability to actors who wish to use the tool, so the affiliate can find information related to victims, change configuration, download agents or victim information, etc.



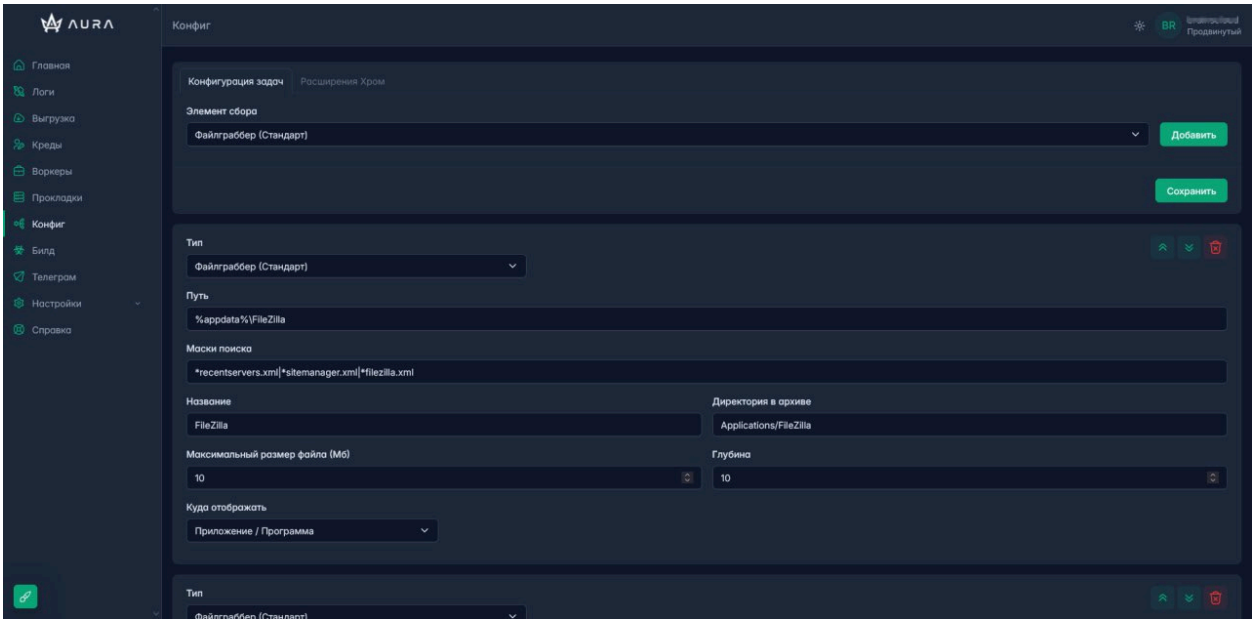
Aura Stealer control panel



Aura Stealer control panel



Aura Stealer control panel



Aura Stealer control panel

Other related tools

Protection Club

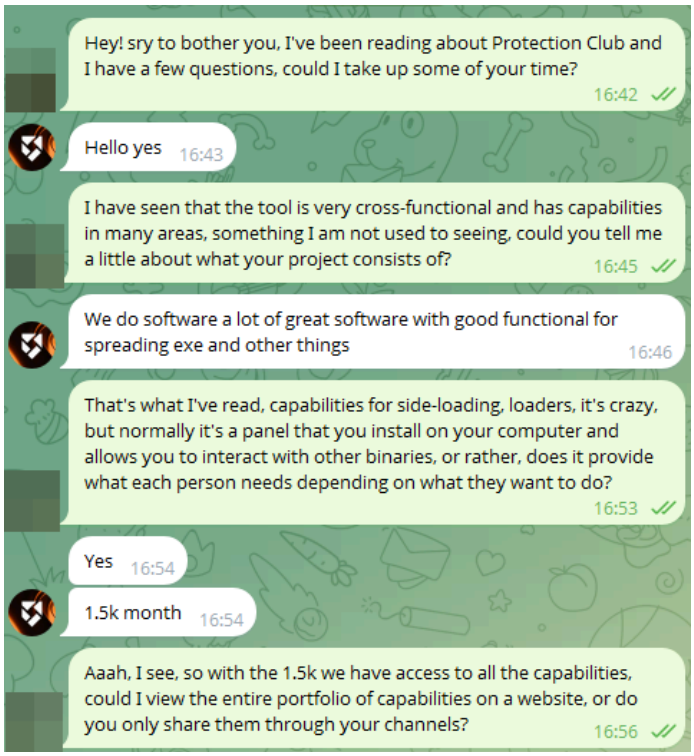
Protection Club is another affiliate group, which has direct relationships with Bulwark, Aura, etc., whose capabilities are software creation, being a group of developers who often create these tools dedicated to bypasses, loaders, side loadings, etc.

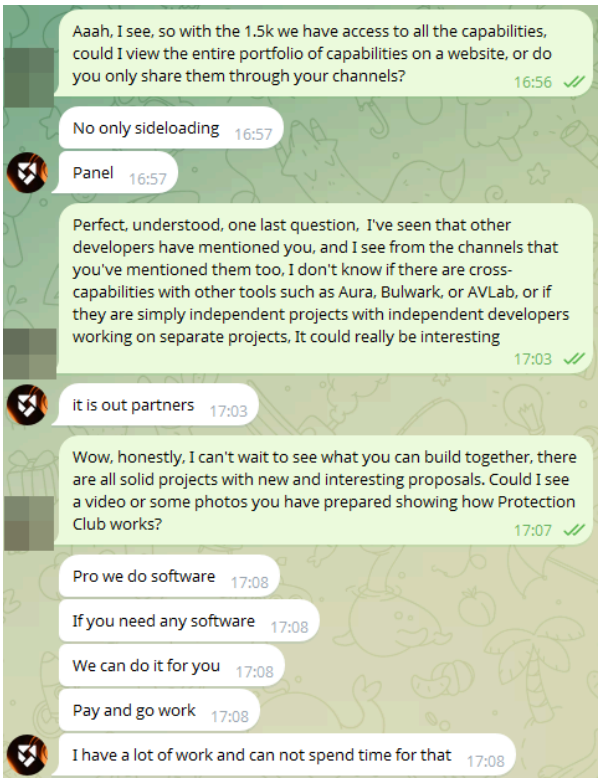
Interview

The Protection Club team was busy and was not very responsive to the questions; however, attempts were made to extract the following information:

- What capabilities their tools have, and how they focus their business
- If there is any demo, information, images, or videos where their work can be seen
- What relationships do they have with other actors seen through the Database (Aura, Bulwark, etc)

Interview screenshots:





Business Model

Protection Club is a team of developers who advertise through Telegram and other platforms like the database mentioned earlier, so that affiliates can request tools from them or they can serve those they already have. Within their social networks, they have advertised:

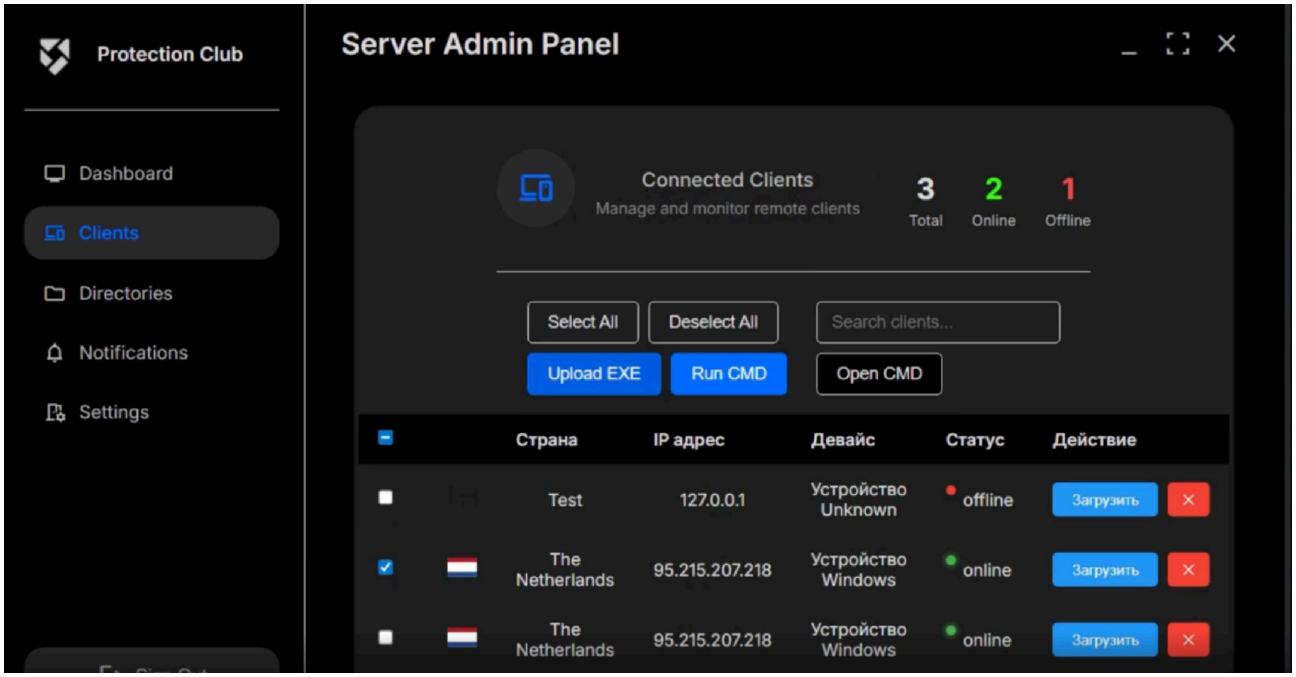
- Loaders focused on botnets with bypass techniques
- RATs targeting different OS
- Phishing panels
- Bypassers
- Side Loaders

Linktree of Protection Club

The Actor mentions the possibility of paying $1.5k per month to access a panel that allows us to obtain side-loading functionality, a technique they work with more regularly; however, they are open to requests to create other software within their scope.

Advertisement of the malicious product on a Telegram channel

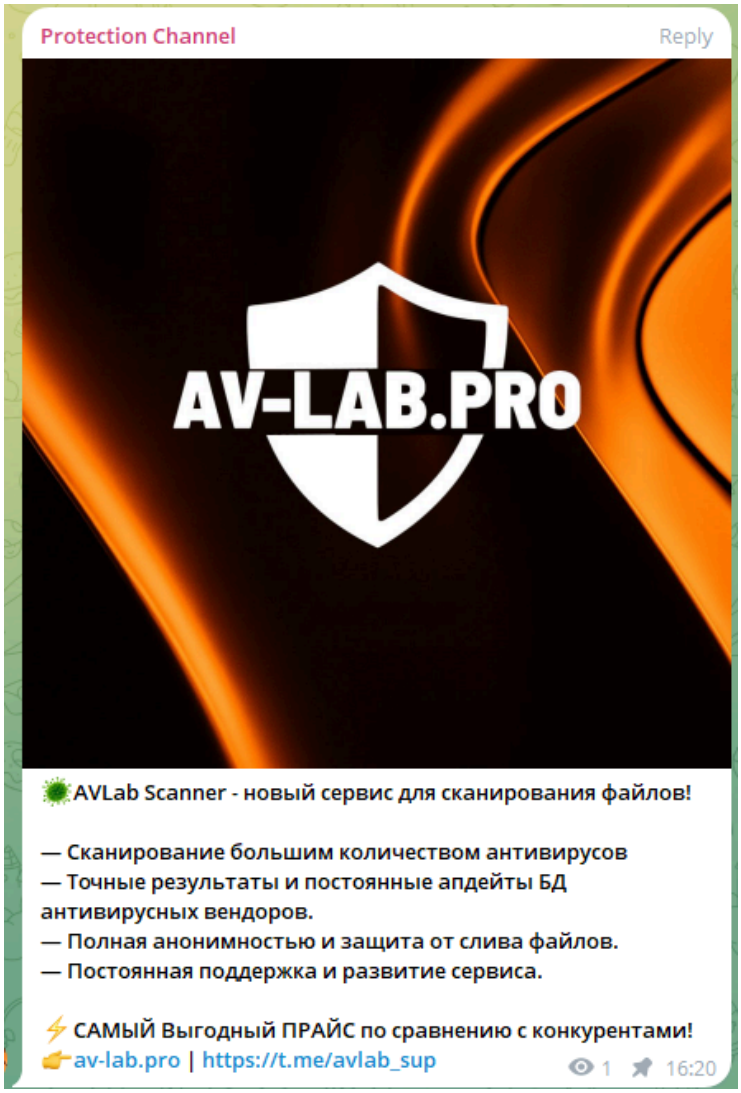
Within the panel, clients or affiliates can be observed, as well as access to directories, where it is understood that there will be information or access to the software they develop for us, acting as a launcher where they can attach their work so that those who buy Protection Club have easy access from their devices.



Protection Club Admin Panel

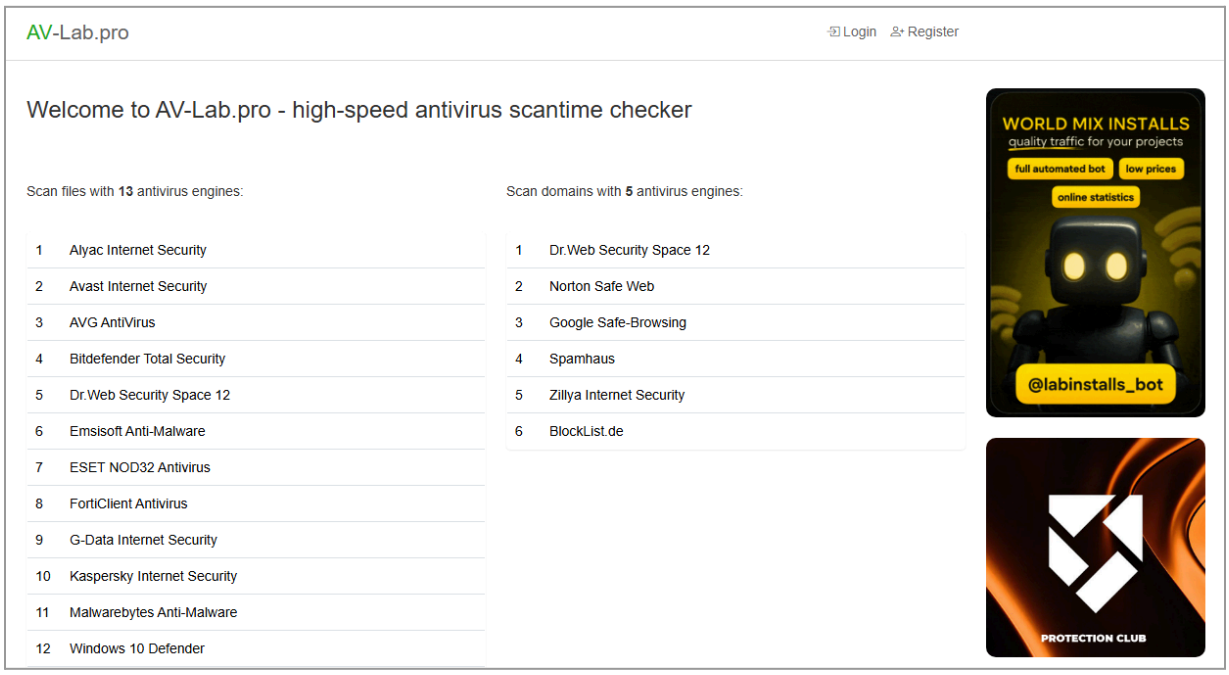
AV-Lab

Among the related entities where information has been shared recurrently in homonymous channels is AV-LAB, which allows us, like other tools such as VirusTotal, to analyze samples in real time to locate whether engines can detect it.



Advertisement of the malicious product on a Telegram channel

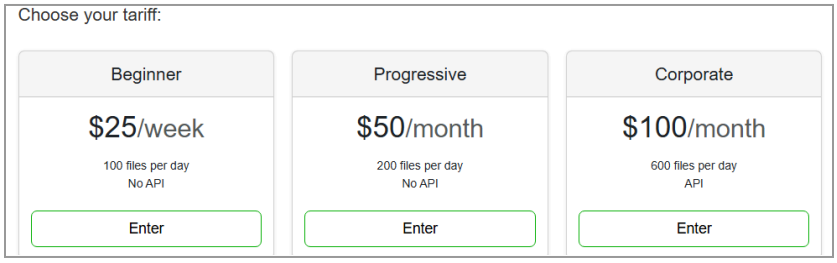
On the page itself, advertising for related entities such as Protection Club can be found:



Business Model

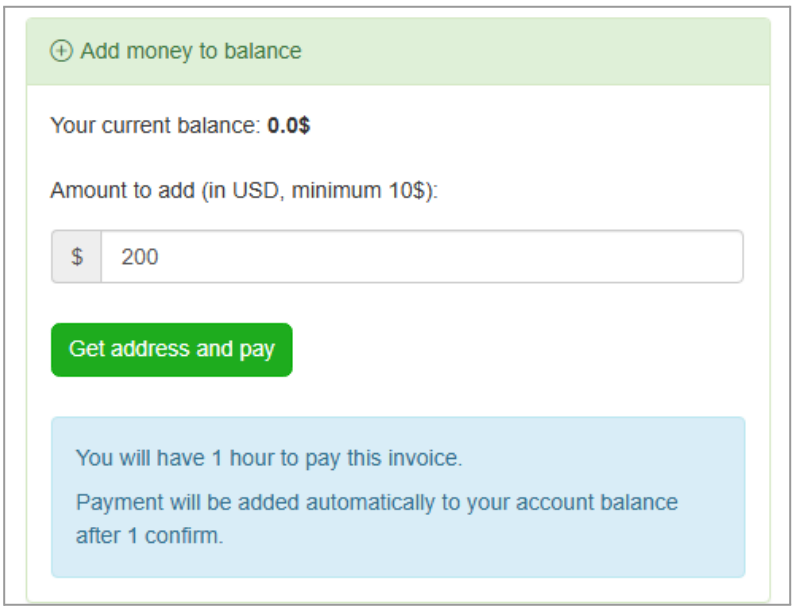
AV-LAB, like the other partners (Aura, Bulwark, Protection Club, etc.), promises good capabilities at a competitive price. In this case, offering a business model focused on three packages, which, as is usually common, tends to be more economical and competitive as we approach the more premium versions, in this case, "Corporate":

- **Beginner:** Allows us 100 files per day, without the possibility of using the API at $25 per week
- **Progressive:** Allows double the files per day, 200, without API capabilities for $50 per month.
- **Corporate:** The most premium version, where they allow us 600 files per day, with API usage capability for $100 per month



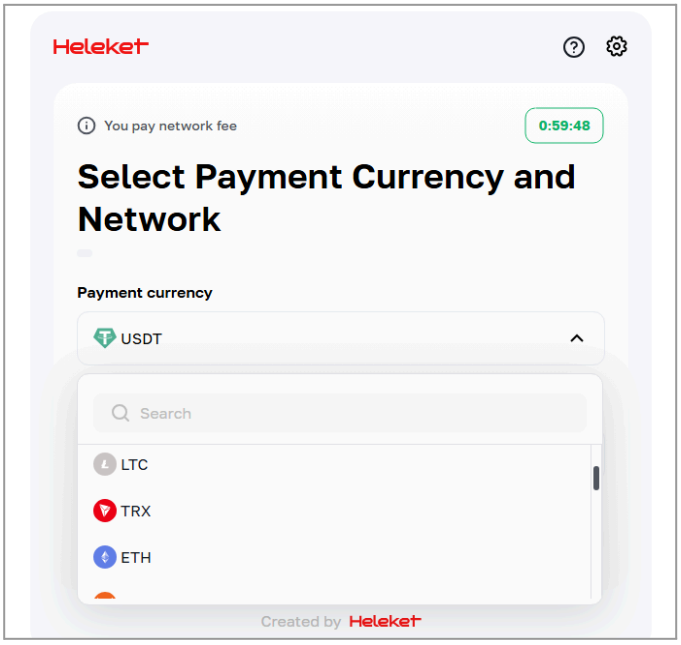
AV-LAB tariffs

As users, we can choose one of the available modes as long as we have funds, otherwise, we must pay through cryptocurrencies to be able to use the tool.

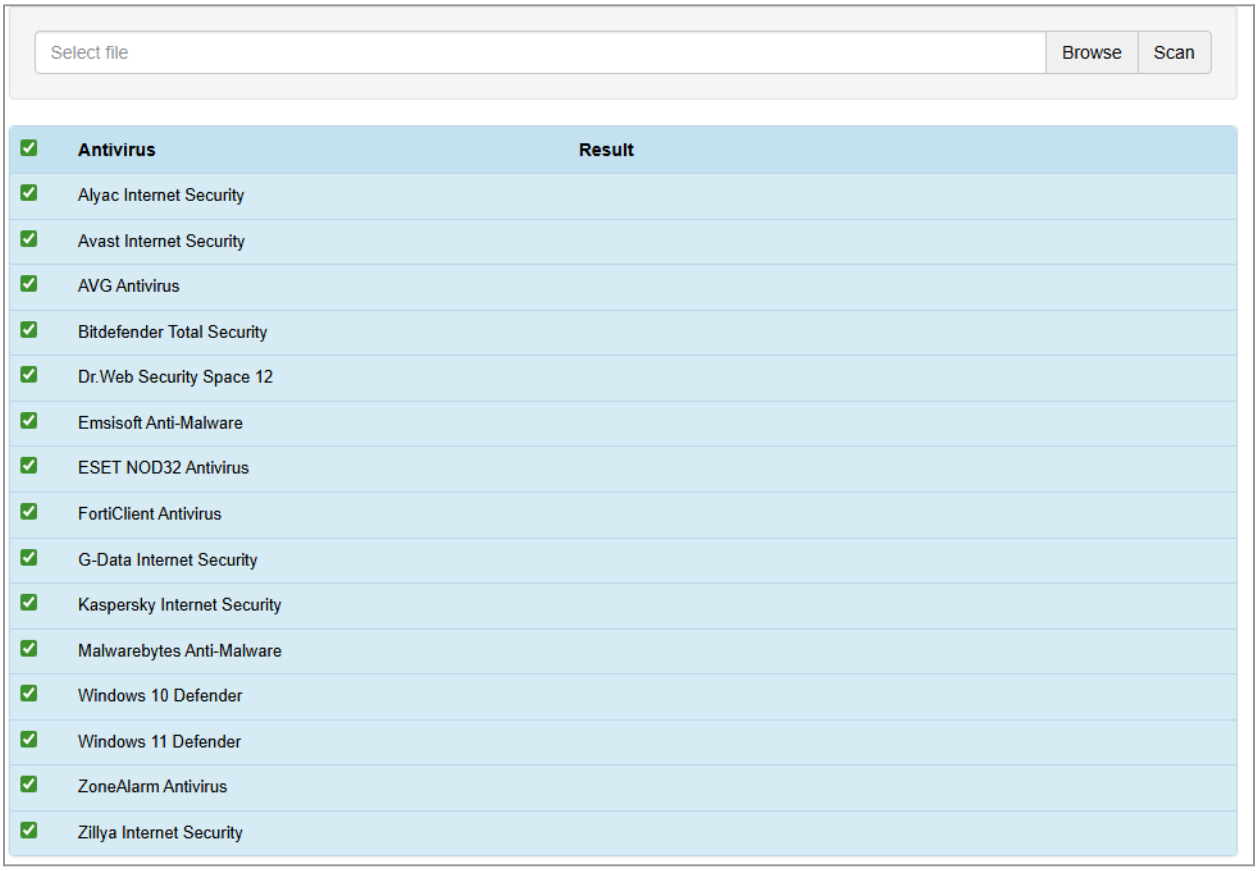


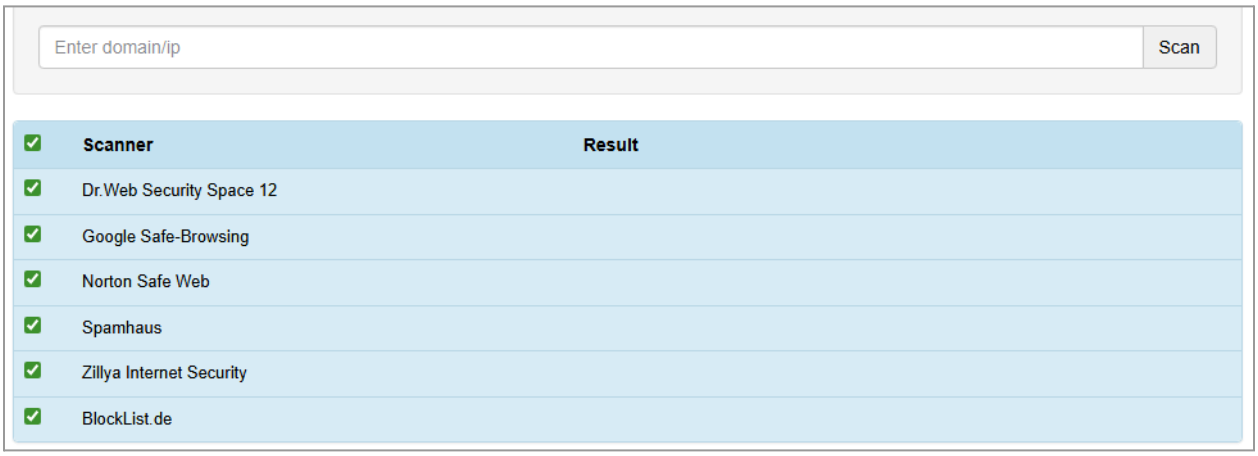
AV-LAB payment method

AV-LAB payment method



Within the tool's capabilities, it allows for the launch of files and the checking of domains or IPs in different engines:

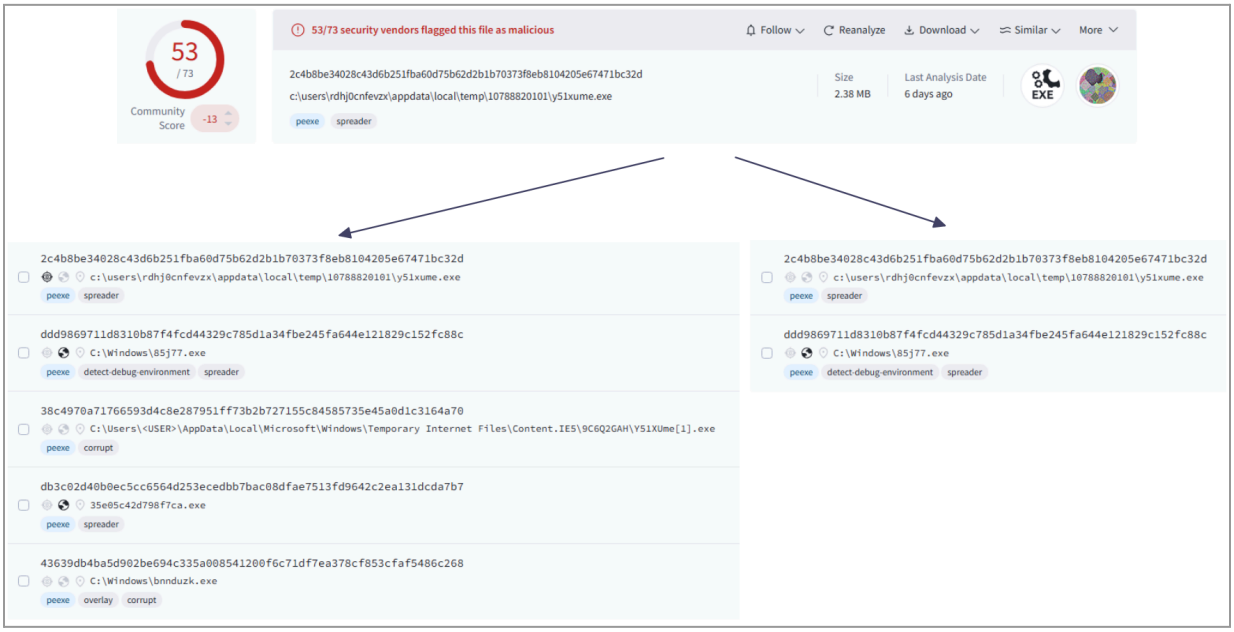




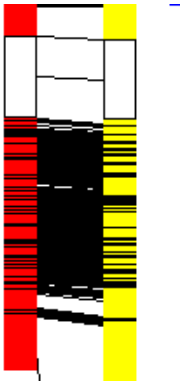
Conclusion

The relationships shown by the multiple tools found in the database indicate connections both in terms of capabilities and tooling and among the creators themselves, revealing a very similar business model in which they complement one another and form a network of criminal projects that serve each other. They enjoy a large window of exposure thanks to their activity on social platforms, using both Telegram and the Dark Web, and are now more accessible than ever via the database search engine, allowing their tools to reach more people and enabling escalation to the development of customized tools or malware, including security-solution bypassers and stealers, which are already a reality.

On Bulwark's side, it is possible to see how the tool is being used publicly, since the samples obtained were used to pivot to other very similar ones that used Bulwark, which contain the same headers and functions as those analyzed, showing an imminent danger to companies.



VirusTotal scan results



Comparison between original bulwark and other pivoted versions of bulwark

IOCs

2D33509252FDEB9BF8D49F168E34CA938D6BDC2730695B7394FAF5D8F785D27D
e91f79999728911847313f70ec1ac76ff5965b43c929bc4db7c2f55d62f353d2
BAC52FFC8072893FF26CDBF1DF1ECBCBB1762DED80249D3C9D420F62ED0DC202
2c4b8be34028c43d6b251fba60d75b62d2b1b70373f8eb8104205e67471bc32d
a73f7ff2df033591c1821fc5a74d435d5718486a3fcd9030ac8b046abef61ed7
4f19aec7ae80d0595d0299470900415434ecd3b1fc03f495b69617e6489f359d
db3c02d40b0ec5cc6564d253ecedbb7bac08dfae7513fd9642c2ea131dcda7b7
f2607505337d23ecee2017a24463d00dba41a127db02ec4b347e10dcdeafe43b
8f9e60a5a448126c684c2f53b6d397d751dbc6a9d005d8365ed3c4e38868f96e
bf60ddab670c7ed28632ddbbb2e871be853a51003441abb5e79641515df70217
ddd9869711d8310b87f4fcd44329c785d1a34fbe245fa644e121829c152fc88c

Domains

softytoys[.]shop
auracorp[.]cc
secondhandcloth[.]shop
armydevice[.]shop
opencamping[.]shop
glossmagazine[.]shop
balancedassetline[.]xyz
radioengineering[.]shop
secondhandcloth[.]shop
luxgames[.]shop
cartdetails[.]shop
softytoys[.]shop
glossmagazine[.]shop
armydevice[.]shop
fancycloud[.]shop
clockhouse[.]shop

TA Sites

Database webpage	https://database[.]forum/index.html
Database TG	https://t[.]me/rootdatabase
Database Admin	https://t[.]me/dbroot
Database Support	https://t[.]me/sup_database
Database forum	https://t[.]me/+tqX1gmdIfikxMDM6
Database Onion website	goxq6gj4nplseo3ziivgpht24skew55sly5zazldix7bb3pfeubxt2id[.]onion
Database Bitcoin	bc1qhsm7lssgzljvetx276y90wl7r7rw9jcjtsjymg
Database Litecoin	LLPrQ4w1MJX56GcvP4bs7twMzbWnXZiAkP
Aura linker	https://usrlnk[.]io/auracorp
Aura Support	https://t[.]me/i7343p2js9uzbozx
Aura principal channel	https://t[.]me/+PPVWLEojMQxhMWUx
Protection Club Admin TG	https://t[.]me/protection_adm
Protection Club Channel	https://t[.]me/protection_club
Protection Club Chat	https://t[.]me/+O54K8aj91OVlNTMy
Av-lab webpage	https://av-lab[.]pro/
Bulwark webpage	https://bulwark[.]pro/

References

- https://foresiet.com/blog/aura-stealer-malware-analysis/
- https://www.linkedin.com/posts/underdark-ai_cybersecurity-infostealer-malware-activity-7355555460499828738-DIUp/
- https://x.com/marktsec46065/status/1942449298018320616
- https://x.com/KrakenLabs_Team/status/1952302052928803182
- https://darkwebinformer.com/ioc-alert-aurastealer-command-and-control-infrastructure/