2025 BERLIN

24 - 26 September, 2025 / Berlin, Germany

TRACKING THE IOT BOTNET'S BLOODLINE: CODE FOOTPRINTS DON'T LIE

Chanbin Jeon, ChangGyun Kim & SeungBeom Lim

SANDS Lab, South Korea

chanbining@sandslab.io
cgkim@sandslab.io
lsb51490@sandslab.io

ABSTRACT

In January 2025, we identified a previously unreported IoT bot named 'x86', confirmed to be a variant of the Gafgyt (BASHLITE) family, which had infected over one million IoT devices. Upon execution, the bot connects to a command-and-control server in Germany and sends the message 'Joined RebirthReborn As'. Unlike typical variants, x86 lacks propagation features and supports only six hard-coded commands, indicating a simplified operational model.

IoT botnet analysis is increasingly difficult due to widespread code reuse and minor modifications by threat actors to evade detection. Manual analysis methods are insufficient to manage the growing number of variants, highlighting the need for automated techniques to track malware lineage and identify relationships between variants.

To meet this challenge, we introduce a hybrid automated analysis framework that combines function-level embedding vector similarity with large language models (LLMs). Decompiled malware functions are transformed into embedding vectors, with similarity measured using cosine and Euclidean distance. This enables efficient clustering, accurate variant detection, and comprehensive malware genealogy tracking.

Using this technique, we analysed the lineage of x86 and correlated it with contextual intelligence derived from open-source social channels commonly used by threat actors, including *YouTube*, *Discord*, *Telegram*, *Instagram* and *Twitch*. Our investigation revealed a clear evolution from the 'qBot' variant, progressing through 'Demon' and 'Rebirth', to the current 'Rebirth Reborn'.

We named the threat actor group behind this lineage 'CTX-5341', comprising actors such as 'Self-Rep-NeTiS'. CTX-5341 has operated through cooperative frameworks and reseller models. One member recently launched a standalone DDoS service, 'Eternal Stresser', indicating further fragmentation.

Although Self-Rep-NeTiS reportedly exited the scene around 2022 after selling all related source code, the reappearance of Rebirth Reborn in late 2024 suggests renewed activity or third-party operators. Our framework proved highly effective in tracing this lineage. It can also be adapted to other malware families depending on the structure of the analysis dataset, making it a valuable tool for modern malware investigations.

1. DETECTION OF NEW IOT BOT

1.1 Background of 'x86' bot discovery

In January 2025, we detected an ELF binary named 'x86' that transmitted the TCP payload string 'Joined Rebirth Reborn'. The command-and-control (C2) server with which the sample communicated also hosted shell scripts designed to execute on systems susceptible to network-exposed remote-code-execution vulnerabilities. Subsequent analysis determined that the file is a variant of the Gafgyt (BASHLITE) family capable of launching distributed-denial-of-service (DDoS) attacks, with portions of its code exhibiting characteristics similar to those found in Qbot.

Figure 1: Shell script obtained from C2 server for bot distribution.

1.2 Key characteristics

The x86 bot supports multi-target attack capabilities via the `listFork()` routine – an archetypal feature of the Gafgyt (BASHLITE) family – and additionally implements VSE (Valve Source engine) flood and UDP HEX flood attacks that are more commonly observed in Qbot variants. Gafgyt (Bashlite/Lizkebab/Torlus) first appeared in August 2014, Qbot emerged in late 2015, and Mirai followed in August 2016. Notably, the C2 server with which this sample communicates (IP 77.90.7.228) is allocated by *GHOSTnet GmbH*, a German internet service provider.

Botnet nomenclature is traditionally derived from the message or socket name transmitted to the C2 server immediately after `InitConnection()`. Since this bot sends the string `Joined RebirthReborn As` to its C2 server, we designate it Rebirth Reborn.

1.3 Differences from existing botnets

A defining hallmark of Gafgyt is the `StartTheLelz()` routine, which propagates the bot by exploiting the ShellShock vulnerability (CVE-2014-6271). In the current Rebirth Reborn bot, however, this infection-and-propagation capability has been excised; only the exploit payload is retained. Moreover, the sample recognizes exactly six hard-coded commands: `UDP`, `XMAS`, `VSE`, `TCP`, `HEX` and `STOP`. These observations suggest that the developer deliberately removed the difficult-to-control spreading mechanism so that the bot would operate exclusively on explicitly designated targets, thereby enhancing operational efficiency – although this absence may simply reflect limited development expertise.

```
POST /ws/v1/cluster/apps HTTP/1.1
Host: x.x.x.x:8088
Content-Length: 261
Accept-Encoding: gzip, deflate
Accept: */*
User-Agent: python-requests/2.6.0 CPython/2.7.5 Linux/3.10.0-
862.14.4.el7.x86_64
Connection: keep-alive
Content-Type: application/json

{"am-container-spec": {"commands": {"command": "cd /tmp; wget
http://167.99.51.231/bash; chmod 777 *; ./bash drone; rm -rf *"}},
"application-id": "application_xxxxxxxxxx_xxxx", "application-type":
"YARN", "application-name": "get-shell"}
```

Figure 2: Example of exploit used to spread Demon bot [1].

2. PROLIFERATION OF IOT BOTNETS

2.1 Widespread adoption of code reuse and incremental modifications

Code reuse is highly prevalent in software development. For adversaries, it simultaneously reduces development costs and enables rapid deployment, thereby serving as a major catalyst for the explosive emergence of countless variants. A quintessential example is the DDoS botnet: once source code was released, innumerable derivatives appeared, and it has since become emblematic of IoT security threats. Most contemporary DDoS botnets appropriate the code base of Mirai or Gafgyt, or at minimum preserve an architectural framework.

Malware source code is actively distributed through internet forums, the dark web, and specific *Telegram* channels, with platforms or users such as 'MIRAILOVERS' representing exemplary cases of collecting, editing, and providing diverse sources for derivative malware based on known DDoS botnets. These sharing activities provide low-barrier attackers with a foundation for easily constructing malicious botnets, while incident response processes face the challenge of repeatedly identifying and addressing similar threats.

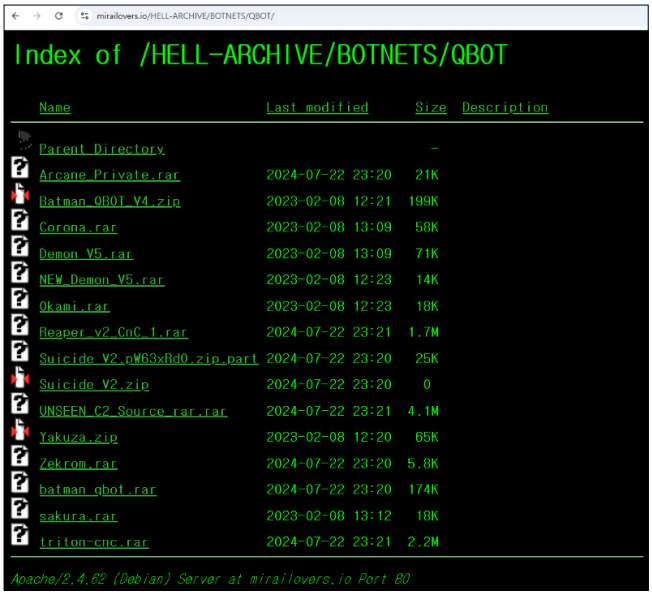

Figure 3: DDoS bot source code being distributed on MIRAILOVERS.IO [2].

In certain online communities, the offerings extend beyond the malware source code itself to include build scripts, detailed guidelines for configuring C2 servers, and curated lists of vulnerable IoT devices. Such comprehensive resource sharing creates an environment in which even actors with limited technical proficiency can readily assemble a fully operational attack infrastructure.

Figure 4: C2 server setup guide for operating Mirai bot.

2.2 Limitations of manual analysis and the necessity for automation

With the rapid proliferation of IoT botnets, the number of malware variants has risen exponentially. Countless samples applying only minor modifications – simple string substitutions, changes in function names, or alternative packing schemes – continue to surface; many of these exhibit functionalities analogous to existing threats yet propagate under new identifiers and appearances. This situation dramatically amplifies the workload of malware analysts. Surveys of individual IoT botnet families often reveal hundreds to thousands of distinct variants, and manually analysing every specimen is not only inefficient in terms of time and personnel but also overly dependent on an analyst's memory and experience, thereby compromising both accuracy and consistency.

To address this challenge, analysis automation is indispensable. Automated systems can execute repetitive, structured tasks rapidly and with high precision, mechanically comparing code similarity, communication patterns, and behaviour-based features to identify variants more accurately than human analysts. Automation further mitigates disparities in analyst skill levels and enables a systematic, threat-intelligence-driven response framework.

In summary, manual analysis alone is inadequate in today's explosive IoT-malware landscape. The adoption and continual refinement of automated analysis technologies are essential for more effective and timely countermeasures; this is not merely a matter of efficiency, but a strategic imperative for building a sustainable security-response posture.

3. LINEAGE ANALYSIS FRAMEWORK

SANDS Lab has developed and now operates an in-house automated malware analysis framework designed to support efficient lineage reconstruction and variant identification across the large volumes of malicious code samples it ingests. The framework automatically decompiles each sample, partitions and extracts it at the function level, and centres its analysis on comparing function similarity.

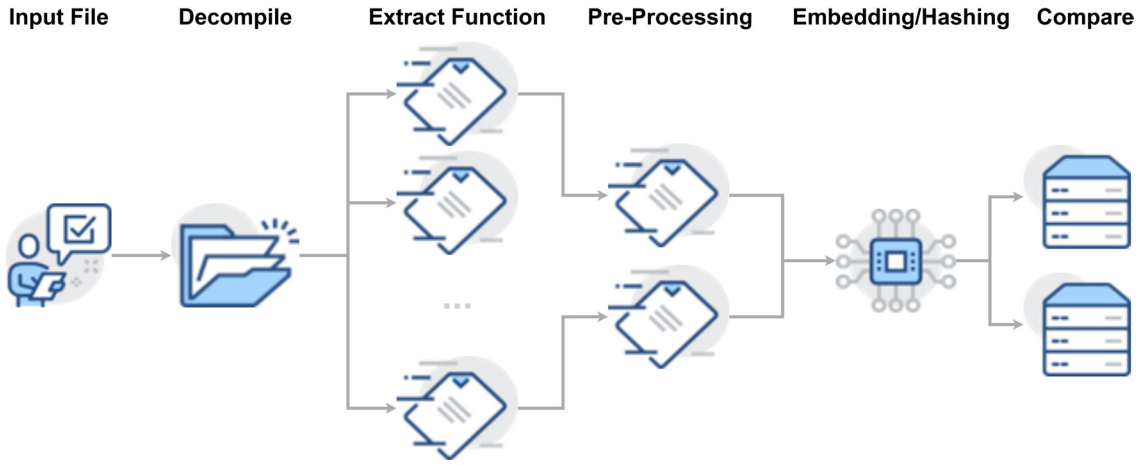

Figure 5: Structure of the lineage analysis framework.

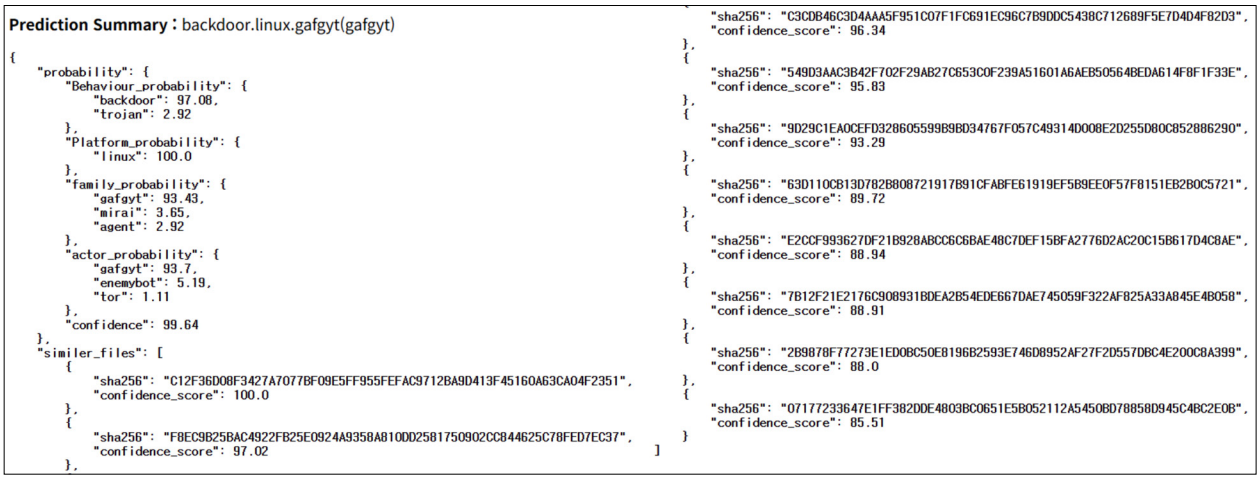

Figure 6: Inference results for the analysed file.

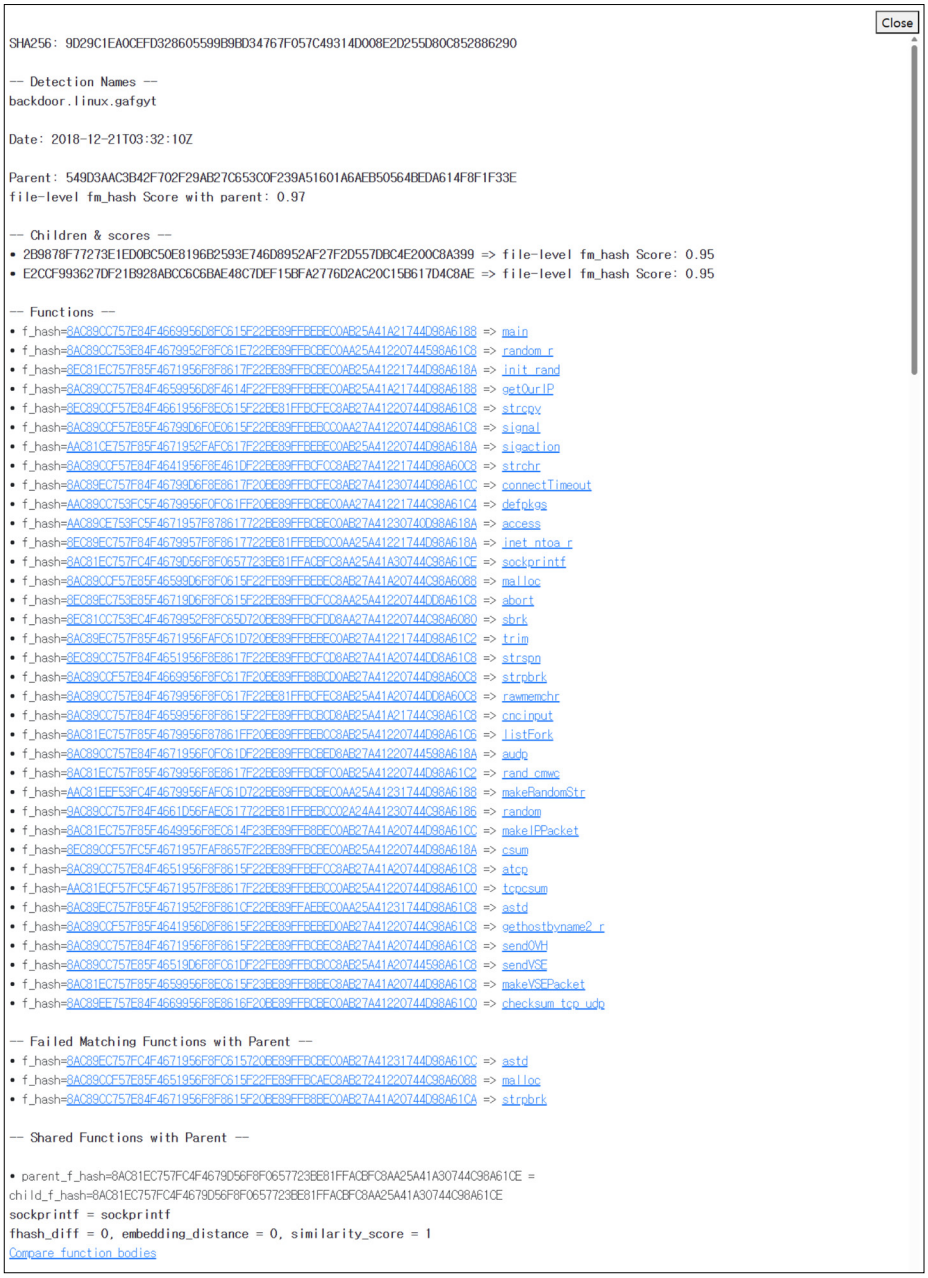

Figure 7: File similarity analysis results and extracted function list.

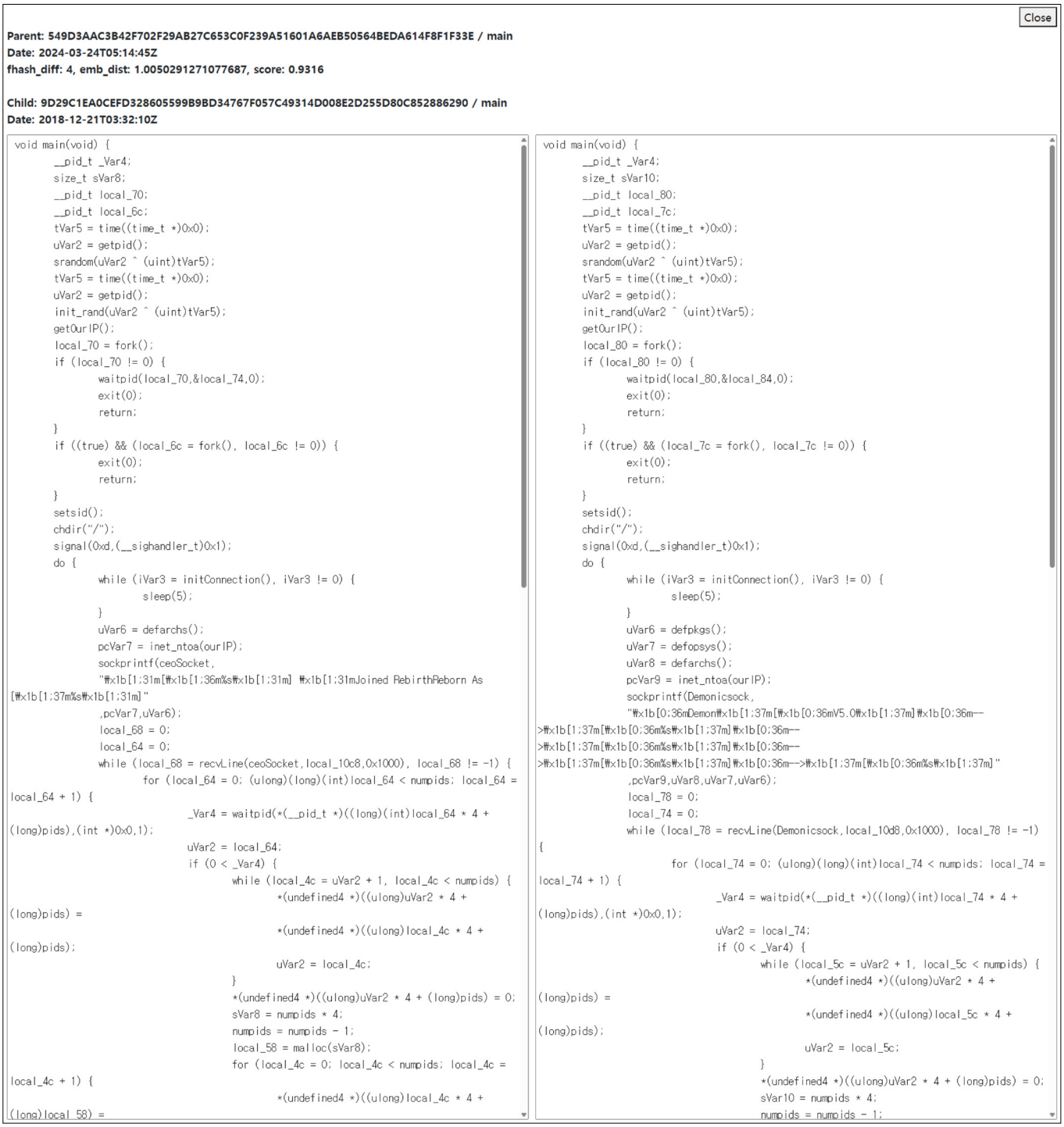

Figure 8: Similarity comparison by function.

A file and function similarity analysis technique offers a highly effective approach to malware lineage reconstruction and static feature examination. Beginning with a single input sample, the method identifies other malicious files that are structurally or behaviourally similar and, on that basis, generates a visual representation of the malware's evolutionary lineage. Similarity is calculated by leveraging embedding-vector information for each extracted function, which enables inference of the malware family, target platform (e.g. *Windows* or *Linux*), principal behavioural traits, and the threat actor group that is likely responsible for the lineage.

In the resulting lineage graph, each node represents an individual malware sample. Selecting a node reveals the file's internal function list, similarity scores for each function, and the corresponding decompiled code. Edges indicate similarity relationships between parent and child samples. The interface permits visual comparison at the function level, allowing rapid and precise identification of code changes such as the addition or removal of command sets in an identical C2 communication routine or the progressive refinement of an encryption module.

This analytical approach moves beyond identifying the functionality of a single specimen. It clarifies the evolutionary path the sample has followed and situates it within the continuum of related campaigns or threat groups, thereby providing substantial value for threat intelligence generation and incident response. Using this framework, we traced the lineage of the Rebirth Reborn malware and quickly determined that the sample descends from a previously disclosed botnet family. The findings aligned closely with external open-source intelligence (OSINT) sources, demonstrating the precision and utility of the *SANDS Lab* framework.

Such automation-centred analysis plays a crucial role in real-time tracking of malware evolution and in the systematic consolidation of threat intelligence. In an environment where large numbers of variants are created rapidly, particularly in IoT malware, this capability is essential for overcoming the inherent limitations of manual analysis.

4. LINEAGE ANALYSIS OF THE NEW BOT REBIRTH REBORN

4.1 Lineage tracking of Rebirth Reborn

By setting the Rebirth Reborn malware sample as the root node of the lineage graph and tracing the ensuing flow, we identified a link to a Rebirth sample that possesses a similar file hash (`fm_hash`). With increasing lineage depth, associations emerge with Demon v5, Demon v1, and early Gafgyt and Qbot variants, allowing the functional evolution among these samples to be visualized. Each node connects samples whose constituent functions exhibit high similarity to those of Rebirth Reborn. The lineage graph is not arranged chronologically; an edge's direction indicates the relationship between the reference and comparison samples rather than temporal order.

Similarity scores are calculated at the file level by aggregating function-level similarity, with each function evaluated according to internal criteria. At the first major branch the Demon v5 node, compared with its parent Rebirth Reborn node, shows modifications in the `main` (score 0.93), `cncinput` (score 0.65), `atcp` (score 0.74), and `audp` (score 0.65) functions involved in malicious behaviour, whereas all other malicious functions remain identical.

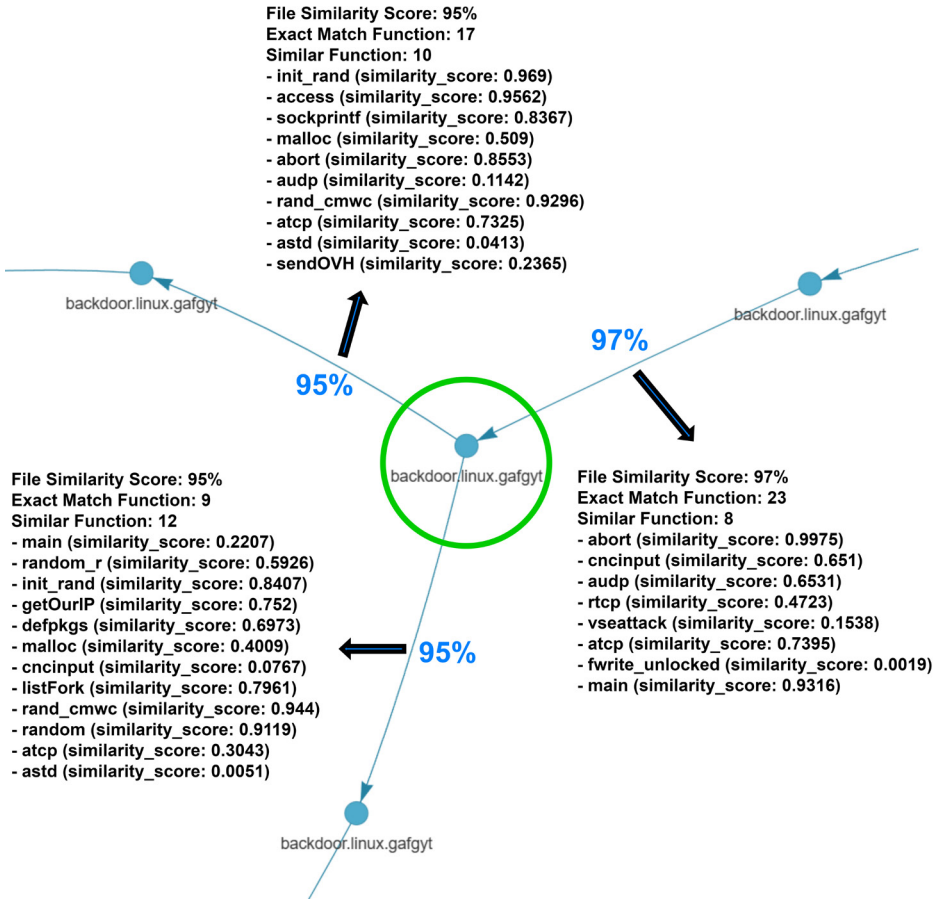

Figure 9: Initial lineage of Rebirth Reborn.

4.2 Evolutionary path from Demon family malware lineage

Demon v1 was an early botnet malware strain with a relatively simple architecture and limited functionality, centred on basic flooding attacks and a small set of control commands. Building on that foundation, Demon v5 diversified its

capabilities and refined its internal structure. Over time, this lineage progressed into the Rebirth series and later re-emerged as Rebirth Reborn. At this stage some legacy functions were removed, new techniques were introduced, and the overall code base tended toward simplification, implying shifts in attacker strategy or operational method. Before detailing the evolutionary path, it is helpful to note that the Rebirth Reborn family comprises four key generations: Demon v1, Demon v5, Rebirth and Rebirth Reborn.

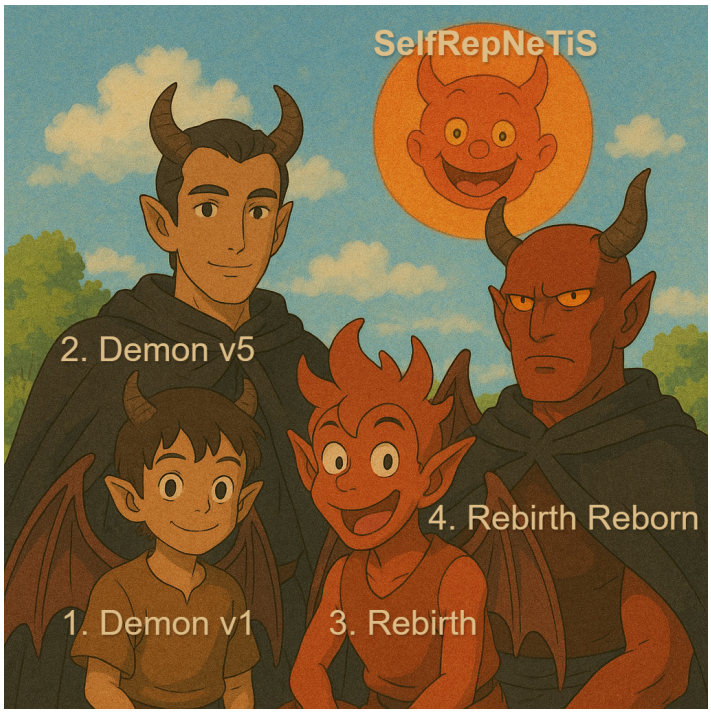

Figure 10: Structure of the Rebirth family.

Evolution of the main() function

In Demon v5, detected in 2018, the `main()` function gathers system details by invoking `defpkgs()`, `defopsys()` and `defarchs()`. In the 2024 Rebirth Reborn sample, this process is reduced to a single call to `defarchs()`, yielding a markedly simplified structure. The socket name used for communication with the C2 server has likewise shifted from `Demonicsock` to `ceoSocket`, a change that may indicate a new operator or rebranding.

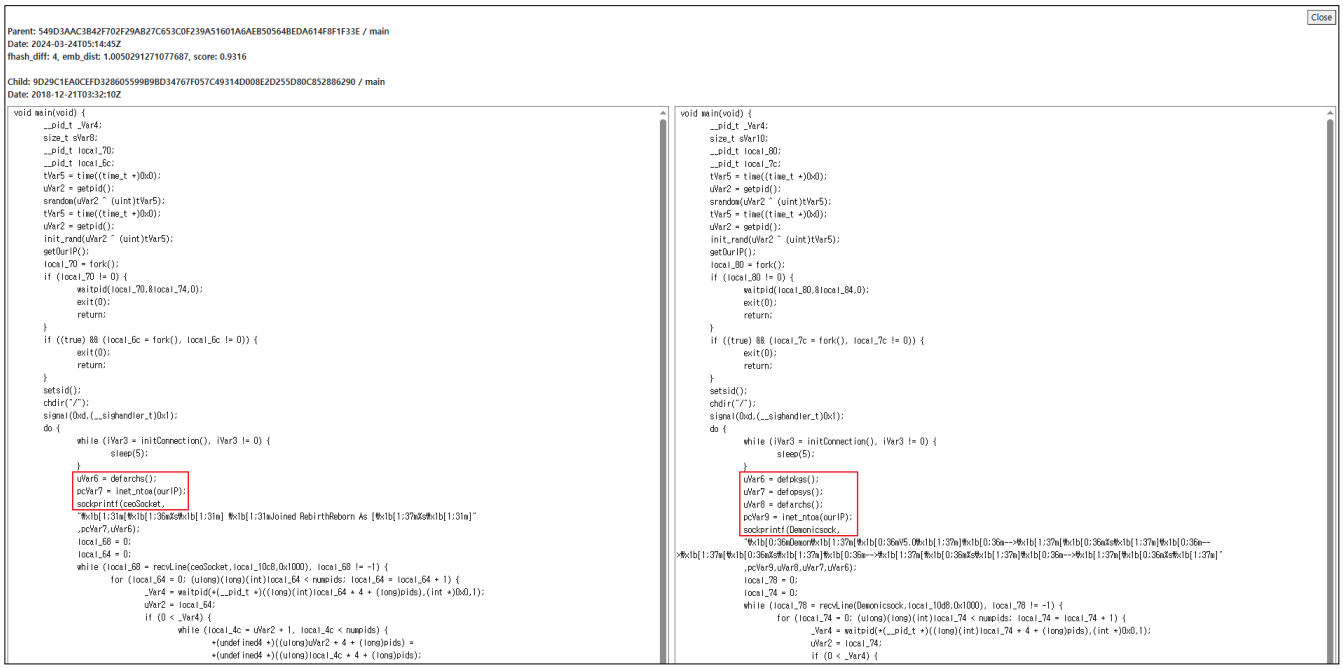

Figure 11: Comparison of main() functions: Rebirth Reborn (parent) vs Demon v5 (child).

Changes in the cncinput() function

Analysis of the `cncinput()` function revealed the addition of a new `XMAS` attack command, which implements a flooding technique that activates all TCP flags simultaneously.

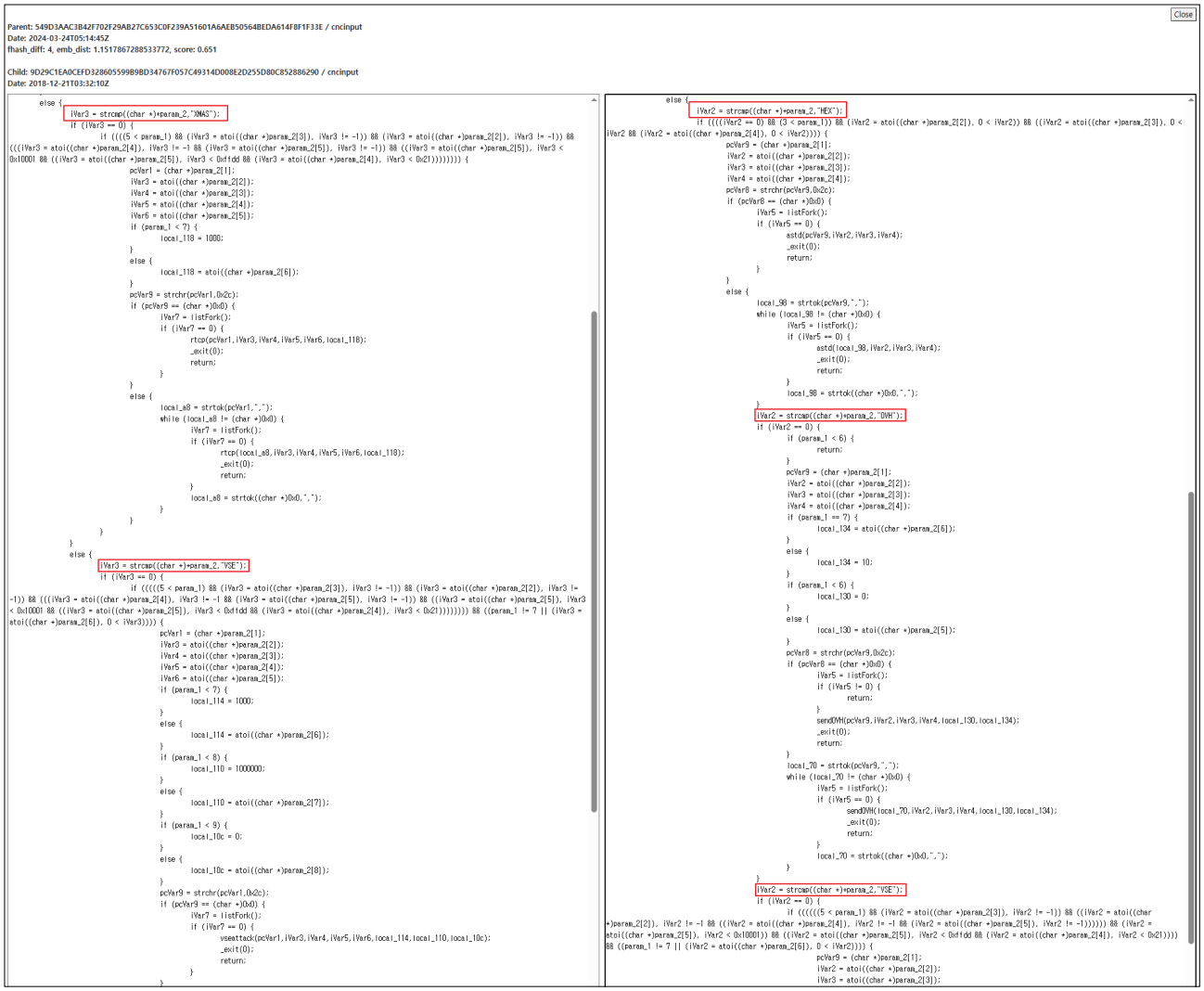

Figure 12: Comparison of cncinput() functions: Rebirth Reborn (parent) vs Demon v5 (child).

Analysis revealed that Demon family malware samples commonly support attacks including UDP flooding, TCP flooding, UDP HEX flooding with fixed payloads, and VSE flooding targeting Valve Source engine (VSE) servers. These constitute fundamental DDoS attack capabilities that have been maintained as core attack methods across most samples.

However, each sample has evolved by adding or removing its own unique attack command configurations built upon this common foundation. For example, Demon v5 incorporated OVH flooding (TCP attacks targeting OVH cloud-based infrastructure) and SERV flooding (attacks targeting specific UDP services), enabling sophisticated attacks tailored to target environments. Conversely, the subsequent sample, Rebirth Reborn, removed these two attack methods and instead added a new XMAS flooding command that sets all TCP flags. This suggests not only changes in attack techniques but also potential modifications in target systems or detection evasion strategies.

Demon v1 included composite attack commands such as `STOMP` in addition to `UDP`, `STD`, `TCP`, `CNC` and `STOP` commands, enabling various flooding and control functions. Samples from this period featured relatively simple functional implementations centred around core control commands.

In comparison, Demon v5 exhibits expanded forms in both command quantity and variety. Various types of attack commands including `UDP`, `TCP`, `HEX`, `OVH`, `VSE`, `SERV` and `STOP` are implemented, with each command having independent processing routines that involve internal argument validation and default value settings. Furthermore, function names such as `sendOVH()` and `sendVSE()` clearly distinguish each attack method within the code, confirming that functional differentiation of commands is well established. These differences demonstrate that each malware version evolves by

adding or removing functions according to target environments, while simultaneously validating the effectiveness of function-level analysis in constructing functional genealogies.

Demon v1	Demon v5	Rebirth	Rebirth Reborn
UDP	UDP	UDP	UDP
TCP	TCP	TCP	TCP
STD	HEX	HEX	HEX
CNC	VSE/OVH	VSE	VSE
STOMP	SERV	XMAS	XMAS
STOP	STOP	STOP	STOP

Table 1: Comparison of commands used in Rebirth Reborn family bots.

Command comparison confirmed that attacks by Demon family malware demonstrate continuous evolution across versions. This extends beyond simple differences in command structure to strategic changes that create substantial impact on victims. The initial version, Demon v1, focused on generic attack capabilities such as UDP, TCP and STD flooding, primarily conducting typical volume-based DDoS attacks that generate indiscriminate traffic. While this could severely impact small- to medium-scale services lacking defence systems, it was relatively easily detected by traffic-blocking equipment above a certain threshold level.

Versions following Demon v5 showed significant improvements in attack command diversity and precision. Attack methods such as OVH flooding and SERV flooding, which directly target specific infrastructure structures (e.g. OVH cloud) or specific UDP services, induce targeted service disruption beyond simple traffic increases. This signifies a transition from simple overload attacks to sophisticated attacks that are difficult to defend against unless victims implement defence strategies specialized for their operational environments.

In Rebirth Reborn, existing OVH and SERV attacks were removed and replaced with anomalous traffic such as XMAS flooding that sets all TCP flags. Consequently, attack traffic evades existing signature-based detection systems or disrupts firewall rule boundaries. Shifts in attack techniques should be construed not as mere code modifications, but as strategic adaptations intended to evade defensive mechanisms and exploit vulnerabilities within victim systems' threat-response processes. Therefore, functional changes in the Rebirth Reborn lineage represent not merely technical advancement, but escalation of threat levels that increase the response burden on victims.

Changes in C2 initial messages

Rebirth Reborn employs highly simplified C2 messages. These messages include only the infected system's IP address and architecture information (e.g. x86, ARM), with an intuitive and concise message format such as '[%s] Joined RebirthReborn As [%s]'. This appears to reflect intentions to enhance rapid propagation and operational efficiency through code simplification. The previous version, Rebirth, has nearly identical message structure, but variant identification is possible through the name being designated as 'Rebirth' within the message. Both samples transmit only minimal system information, which may represent an information exposure minimization strategy for detection evasion.

Conversely, the earlier version, Demon v5, includes more extensive system information in C2 initial messages. Messages include operating system information and installed package information in addition to IP address and architecture, following the structure 'Demon[V5.0]-->[%s]-->[%s]-->[%s]-->[%s]'. This demonstrates the operator's intention to perform more sophisticated system classification at that time. The earliest version, Demon v1, transmitted various information including port, build version, operating system and architecture, with the greatest message length and information volume. The '[Shelling]-->[%s]-->[%s]-->[%s]-->[%s]-->[%s]' format is interpreted as a configuration designed to enhance debugging or operational convenience on the C2 server side.

Overall, C2 initial messages show increasing conciseness over time, indicating that the malware's operational strategies have shifted from precise information collection to a focus on rapid infection and concealment. Changes in message structure, along with code refactoring, represent one of the elements that clearly demonstrate evolution among these lineages.

Demon v1	'[Shelling]-->[%s]-->[%s]-->[%s]-->[%s]-->[%s]'
Demon v5	'Demon[V5.0]-->[%s]-->[%s]-->[%s]-->[%s]'
Rebirth	'[%s] Joined Rebirth As [%s]'
Rebirth Reborn	'[%s] Joined RebirthReborn As [%s]'

Table 2: Initial C2 communication messages used by Rebirth Reborn family bots.

Ultimately, the genealogy of Rebirth Reborn is as shown in Figure 13, and connections to known Gafgyt/Qbot variants can also be confirmed.

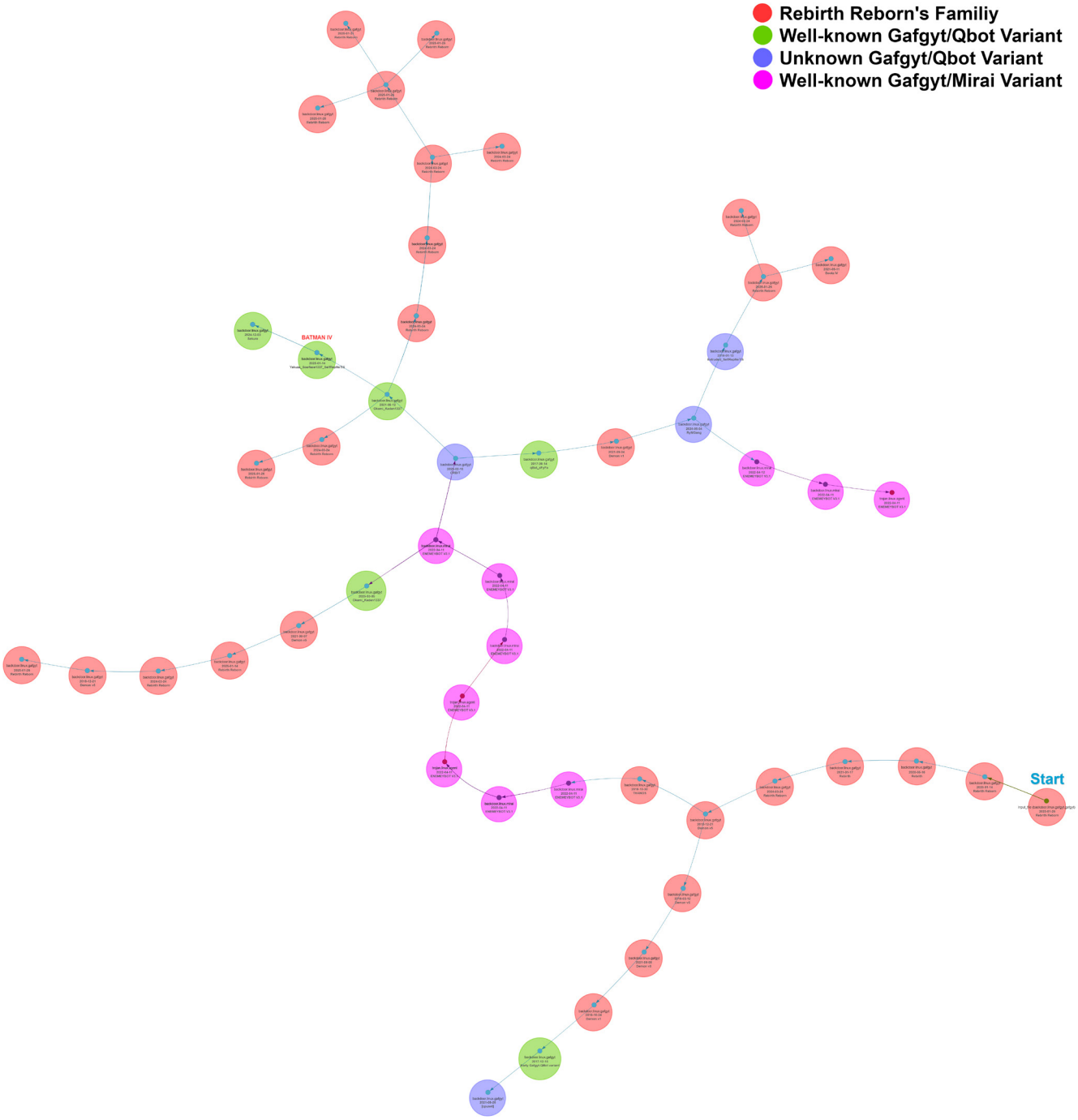

Figure 13: Finalized lineage of Rebirth Reborn.

4.3 OSINT analysis

Activity traces of Self-Rep-NeTiS, known as the creator of Rebirth Reborn, were confirmed across various online platforms including security company blogs, *Pastebin*, *Instagram*, *Telegram*, *YouTube* and *Twitch*. In addition, identical Self-Rep-NeTiS signatures were discovered in all malware samples of the Demon and Rebirth families, and when organized chronologically, their distribution corresponded almost perfectly with the flow of the genealogy graph. Collectively, these circumstances strongly suggest that Self-Rep-NeTiS has directly developed all DDoS bots within this lineage, and this can be considered sufficient evidence to confirm this conclusion with confidence.

Figure 14: Demon v1 (2018) [1].

Figure 15: Demon v5 (2018) [3].

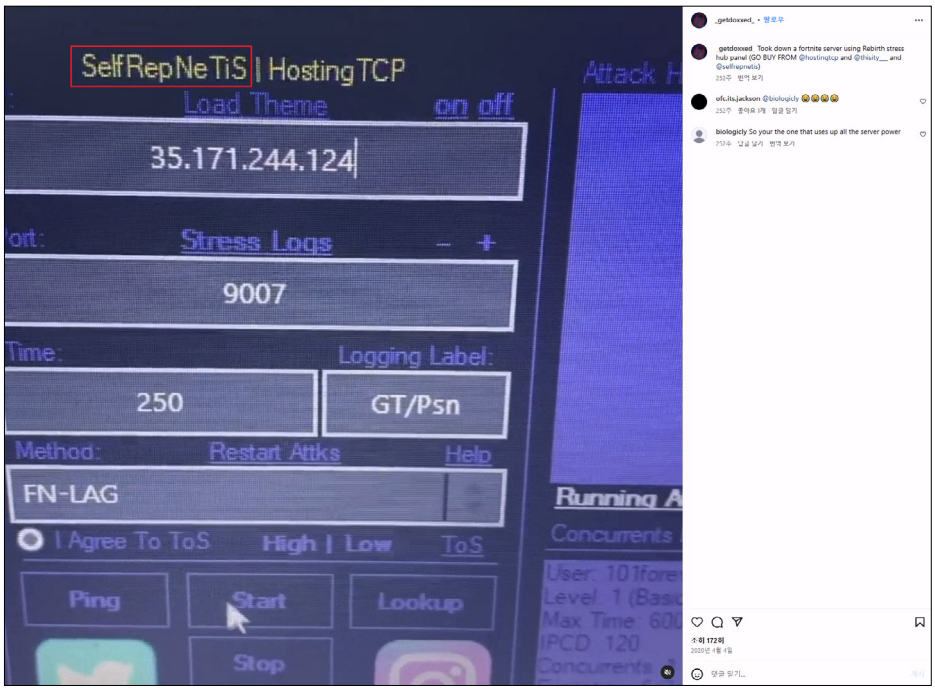

Figure 16: Rebirth (2020) [4].

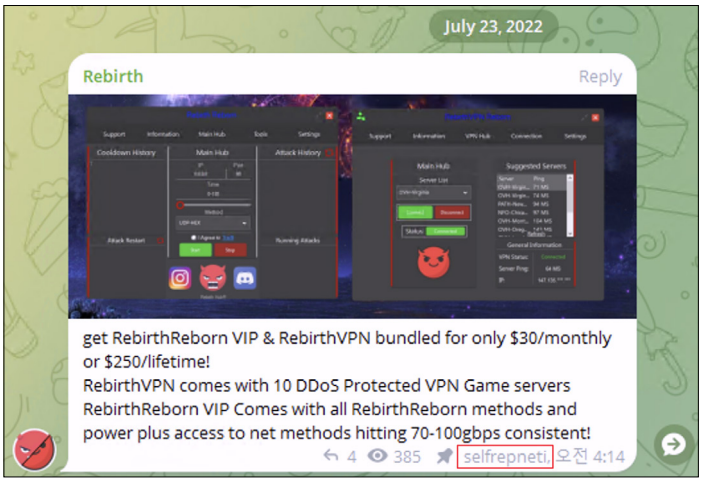

Figure 17: Rebirth Reborn (2021).

4.4 OSINT-based verification of Demon family malware lineage

According to analysis results based on OSINT, a clear technical evolution can be traced from the Demon v1 malware that began distribution in 2018, through subsequent Demon v5 and Rebirth stages, to the currently observed Rebirth Reborn. Analysis results derived from functional changes and code structure differences at each stage demonstrate that malware within this lineage represents evolutionary entities within a continuous genealogy rather than disconnected threats. Particularly, function-level code similarity, consistency in C2 communication methods, and OSINT information regarding chronological distribution pathways cross-validated each other, proving that Rebirth Reborn is not simply a Gafgyt/Qbot derivative but a variant within the Demon lineage continuum. The correlation between Demon and Rebirth is further evidenced by API server banner messages from randle (HostingTCP), who was a core developer of this group.

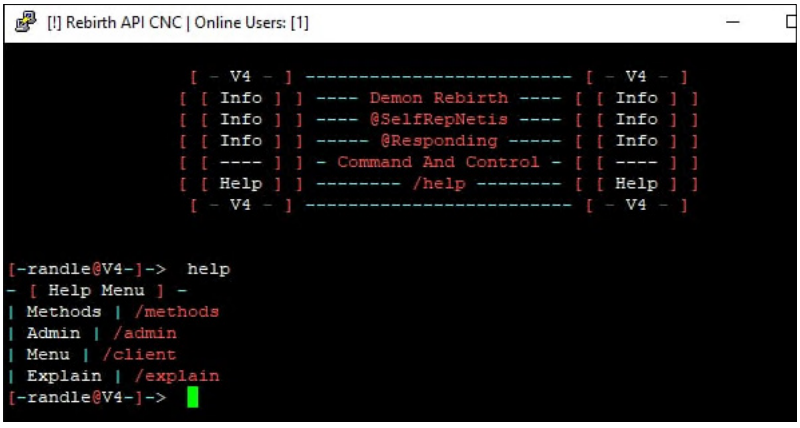

Figure 18: Console screen of randle announcing rebirth of 'Demon'.

5. EMERGENCE OF THREAT ACTOR CTX-5341

5.1 Overview and naming rationale for CTX-5341

SANDS Lab has introduced a proprietary identification framework called CTX-CLIO to ensure consistency in malware family classification and threat actor tracking. The CTX-CLIO system is designed to characterize threat groups using a four-part identifier composed of the following elements: category (type of activity), language/location, intent and order.

The threat group responsible for developing Rebirth Reborn is primarily engaged in botnet-based DDoS attacks, conducts its online operations in English-speaking communities, and is financially motivated. Based on these attributes, we have assigned the following classification:

- 5 (Botnet): Indicates the use of botnets for conducting attacks
- 3 (English): Denotes the language used in community engagement
- 4 (Financial gain): Reflects the group's primary motivation
- 1 (Index): Represents the sequence number assigned to the group

Accordingly, the group is designated as 'CTX-5341'. This identifier is not tied to a single campaign but reflects the group's continuous involvement in the development and deployment of related malware families, including Demon, Rebirth and Rebirth Reborn.

5.2 Principal activities

The CTX-5341 group is believed to have begun its activities around 2018, initially emerging through *Instagram*-based operations. In September 2018, the group – under the alias 'Self-Rep-NeTiS' – publicly released the source code for Demon v1 via *Pastebin* [3]. Shortly thereafter, corresponding malware samples were identified on *VirusTotal*. In January 2019, the group released the source code for Demon v5, indicating ongoing development and refinement.

By 2020, both *Instagram* [4] and *VirusTotal* showed evidence of Rebirth-related malware activity and promotional material, suggesting the onset of more structured commercial operations. Notably, in 2021, a user operating under the alias 'Cash' claimed on the SpyHackerz forum to have introduced the Rebirth service to the Turkish market. That same year, Self-Rep-NeTiS actively promoted the Rebirth brand by launching *Telegram* and *YouTube* channels featuring showcase videos and content related to the service.

In February 2022, the group reappeared under the aliases 'netis', 'tcp', and 'thisity' on *Telegram*, now operating a DDoS-for-hire service under the name 'Rebirth Stresser'. As the user base expanded, the group began selling structured DDoS service plans alongside access to the attack panel, starting from July of that year.

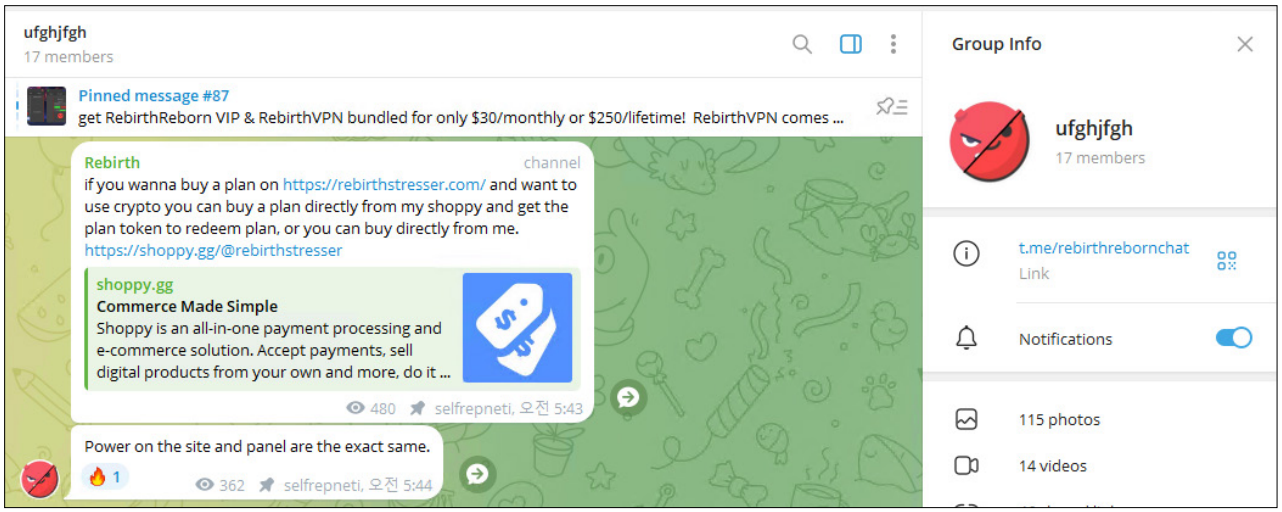

Figure 19: Post advertising Rebirth stresser service.

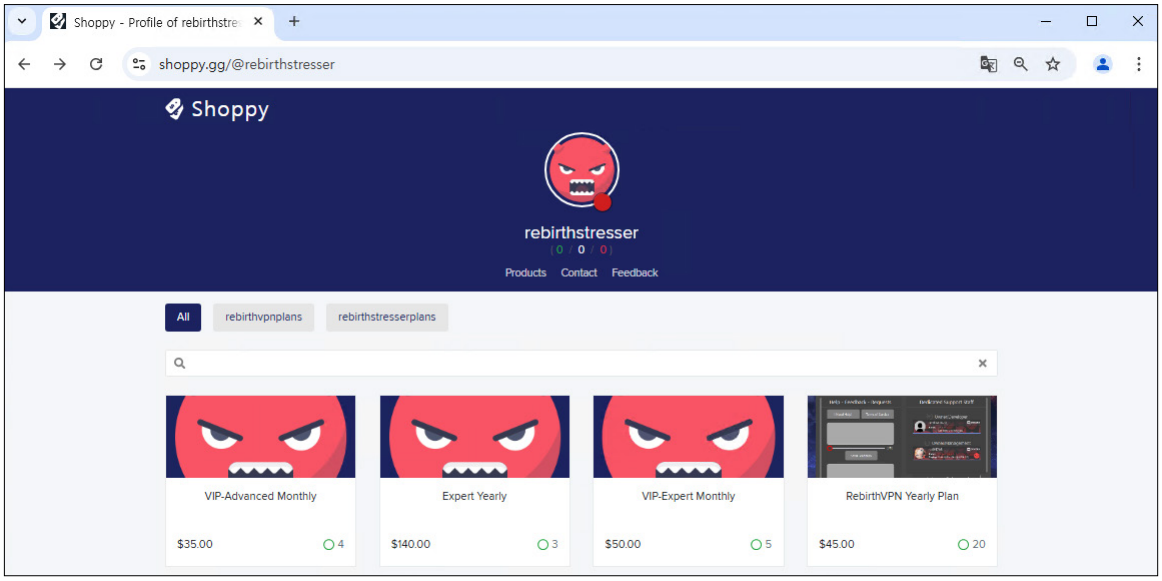

Figure 20: Rebirth Stresser service listed on Shoppy [5].

However, the service began to suffer from quality degradation and operational issues. Beginning in September, problems emerged particularly with the VPN functionality, and tensions within the group became apparent when the developer Self-

Rep-NeTiS publicly announced the sale of the source code. These developments exposed internal conflicts and growing dissatisfaction among users. Subsequently, HostingTCP promoted the Rebirth API and continued service operations.

In 2023, Cash (also known as HostingTCP) launched an independent *Telegram* channel and rebranded the original service as 'Eternal Stresser'. The new channel remained active for some time, although it faced criticism from users due to functional issues with the platform interface. Despite these challenges, Eternal Stresser continued to conduct attacks and offer services commercially until approximately February 2025. Around the time preparations for this paper began, the associated accounts and *Telegram* channels were confirmed to have been deleted.

The CTX-5341 group, therefore, demonstrates a sustained pattern of malicious-service distribution and commercialization through multiple aliases and platforms. Over time, the group has evolved through rebranding, internal fragmentation, and shifts in its user base, reflecting its adaptability and persistence as a threat actor.

Date	Platform	Person/account	Description
2018 (presumed)	*Telegram*	Deleted account	netis, tcp, and thisity begin operating Rebirth Hub
2018.09.29	*Pastebin*	SELF-REP-NETIS	Demon v1 source code uploaded; security firm *Radware* publishes analysis report [1]
2018.10.24	*VirusTotal*	-	Demon v1 sample reported
2018.12.21	*VirusTotal*	-	Demon v5 sample reported
2019.01.26	*Pastebin*	SELF-REP-NETIS	Demon v5 source code uploaded
2020.04.04	*Instagram*	@_getdoxxed_	Proof of purchase for Rebirth Stress Hub via @hostingtcp
2020.05.16	*VirusTotal*	-	Sample titled 'Joined Rebirth As' reported
2021 (presumed)	*Telegram*	@Cash	Claims to have introduced Rebirth to the Turkish market
2021 (presumed)	Eternal Stresser APP	Cash	Claims Eternal Stresser DDoS service launched in 2021
2021.05.27	SpyHackerz	@Cash	Rebirth Stresser Hub sales post, sold via *Discord*
2021.06.17	*Telegram*	@selfrepnetis	Rebirth channel launched (t.me/rebirthreborn)
2021.06.28	*YouTube*	@selfrepnetis	Intro video for RebirthReborn & RebirthVPN uploaded
2021.09.29	SpyHackerz	@Cash	Claims the creator of Demon bot is his friend 'selfrepnetis'
2022.02.15	*Telegram*	@selfrepnetis	Rebirth Chat group created (t.me/rebirthrebornchat)
2022.02.15	*Telegram*	@Cash	Joined the Rebirth Chat group
2022.04.09	*YouTube*	@selfrepnetis	Reseller recruitment video uploaded
2022.07.24	*Instagram*	@selfrepnetis	Promotion of Rebirth services (VPN, Stresser)
2022.08.04	*Telegram*	@selfrepnetis	@selfrepnetis account locked, changed to @selfrepsinatra
2022.09.15	-	-	Service outage begins, increasing user complaints
2022.09.25	*Telegram*	@selfrepnetis	Promotes Rebirth source code sales, growing confusion in chatroom
2022.11.09	*Telegram*	@hostingtcp	Promotes Rebirth API CNC, mentions stomp attack from Demon v1
2022.11.14	*Telegram*	@selfrepnetis	Announces full sale of all Rebirth-related source code

Table 3: Key activities of CTX-5341.

Date	Platform	Person/account	Description
2022.11.16	*Telegram*	@Cash	Launches personal *Telegram* channel
2022.11.26	*Telegram*	@Cash	Requests Rebirth API inquiries via personal DM
2023.01.21	*Telegram*	@Cash	Redirects users to personal bot from Rebirth Chat group
2023.01.24	*Telegram*	@Cash	Mentions Turkish market introduction, requests DM to selfrepneti
2023.03.27	*Telegram*	@Cash	Activates personal channel and rebrands to 'Eternal Stresser'
2023.04.01	*Telegram*	@Cash	First promotion of Eternal Stresser in Eternal channel
2023.07.04	*Telegram*	@Cash	Changes Eternal channel profile picture
2023.07.28	*Telegram*	@Cash	Promotes Eternal Stresser in Rebirth Chat, user complaints arise (UI issues, etc.)
2024.03.24	*VirusTotal*	-	Sample titled 'Joined RebirthReborn As' reported
2023.08.10	*Telegram*	@cashlamo	Promotes sale of Eternal Stresser
2025.02.24	*Telegram*	Eternal	Active demonstrations and sales of Eternal Stresser
2025 (presumed)	*Telegram*	Eternal	Disappears after deleting *Telegram* account and channel

Table 3 contd.: Key activities of CTX-5341.

5.3 Affiliated actors and collaborative structure

Rather than exhibiting a rigid hierarchical structure, the CTX-5341 group operates as a loosely connected network, based primarily on collaborative relationships between technical developers and vendors. Their interactions have been organically shaped according to individual areas of expertise and roles, with cooperation centred around the development and operation of DDoS services such as Rebirth Stresser and Eternal Stresser.

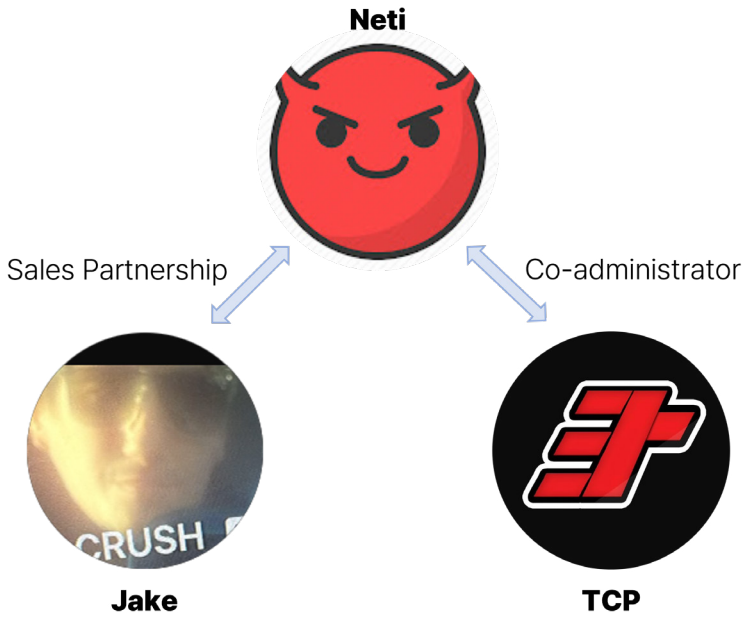

Figure 21: CTX-5341 collaboration structure.

Neti (Self-Rep-NeTiS, Sinatra) – developer and operator

Neti functioned as the technical linchpin for the CTX-5341 campaign, having designed and implemented the architecture of the DDoS attack tool Demon Bot and its derivative service Rebirth Stresser. As both a developer and systems operator, Neti was responsible for building the complex infrastructure supporting VPN services, DDoS hubs, and authentication mechanisms. Notably, he played a leading role in shaping the brand identity and strategic direction of the Rebirth service.

While primarily focused on technical development, Neti also engaged in customer support and sales activities when necessary. He was the principal developer and operator behind both Rebirth and its successor Rebirth Reborn, managing the underlying VPN and DDoS attack platforms. In addition, he is credited with creating the Demon Bot series.

Neti personally designed and implemented a wide array of key components within the Rebirth platform, including the botnet, control panel, API, and VPN infrastructure. However, in late 2022, he ceased his activities after suffering financial losses caused by an attack on the VPN service under his management.

Notably, Neti also maintained a public presence by streaming *Minecraft* gameplay on *Twitch*. He operated under multiple aliases, including @selfrepnetis, @selfrepsinatra, and *Discord* usernames such as ceo#0544 and neti#1768, in addition to using the identifier 'rebirthservices'.

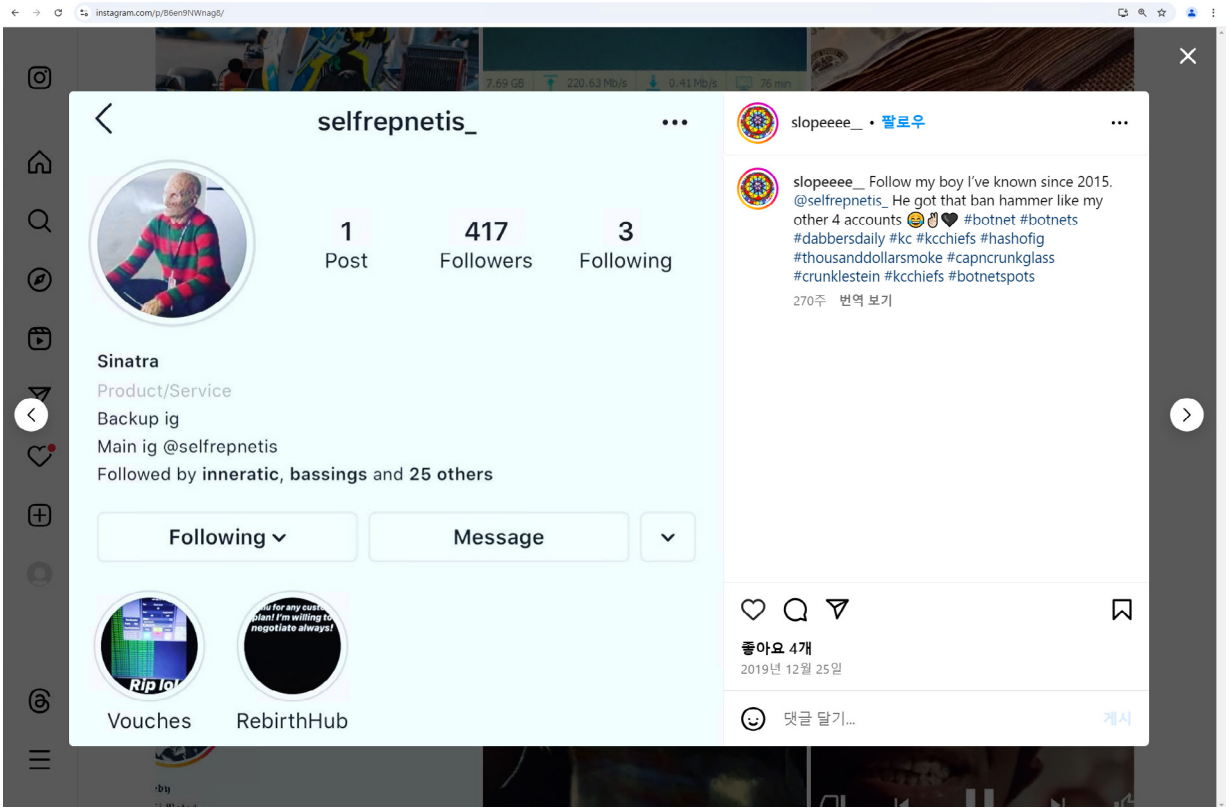

Figure 22: Neti's Instagram profile [6].

TCP (HostingTCP, Randle / Cash) – co-developer and successor operator

TCP is a Turkish-born engineer and operator who collaborated with Neti during the early stages of the Rebirth project, jointly building and managing key portions of its infrastructure. He primarily oversaw the Rebirth API authentication server and played an active role in the platform's sales structure, contributing significantly to its expansion. Leveraging the technical foundation established by Neti, TCP took the lead in distributing the Rebirth service within Turkey and surrounding regional communities, helping to broaden its market presence. He was also responsible for hands-on operational tasks such as customer support and API integration.

Following Neti's withdrawal from the project in late 2022, TCP absorbed the existing user base and infrastructure of Rebirth, launching his own independent DDoS service, Eternal Stresser. He continued to operate the platform autonomously, maintaining it as a spiritual successor to Rebirth, with confirmed activity extending through 2025. However, his whereabouts and involvement have since become unclear, and he is currently considered to be inactive or have gone underground.

In his online activities, TCP used multiple aliases and accounts, including @hostingtcp, @tcpdev, and *Discord* identifiers such as randle#7670, Cash, and @dehatuzcu.

Figure 23: Mention of the connection between Demon and selfrepnetis on SpyHackerz forum [7].

Jake (@slopeeee__) – US-based early distributor

Jake, a Missouri native, was a key early distributor and marketing agent for the Rebirth project. Working in collaboration with Neti, he played a pivotal role in establishing the Rebirth botnet within the market. Up until March 2020, Jake was actively involved as a dedicated vendor, leading marketing initiatives and driving rapid expansion of the customer base. His efforts were instrumental in ensuring the project's successful market entry during its formative phase.

However, after 2020, Jake ceased all sales activities and withdrew from further collaboration with the Rebirth team. Following his departure, distribution and customer management responsibilities were assumed directly by Neti and TCP, who subsequently restructured Rebirth's sales operations around their own efforts.

Jake resided in Missouri and was known to be a fan of the NFL team Kansas City Chiefs. He also reportedly maintained a personal interest in cannabis-related activities. His online presence was primarily associated with the handles @slopeeee__ and @slopeeee___.

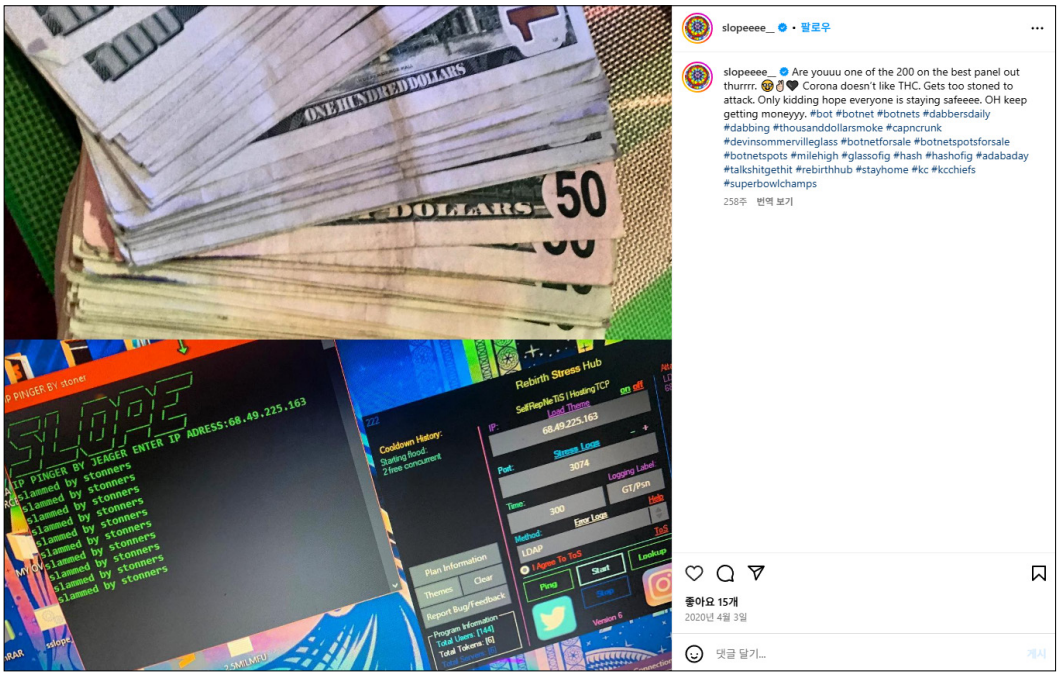

Figure 24: Rebirth Stress Hub sold by Jake on Instagram [8].

6. LAUNCH OF ETERNAL STRESSER

6.1 Transition to an independent DDoS service

The initial Rebirth Hub was not a single-developer service. Instead, it operated through a collaboration between Neti (Self-Rep-NeTiS) and HostingTCP (Cash, Randle). Neti served as the core developer responsible for the DDoS bot, VPN

configuration, and most service-wide components, whereas HostingTCP specialized in building the API authentication server and the DDoS control panel. Their roles were complementary, forming an effective technical partnership.

Although the two actors presented themselves as one team, their cooperation was loose; outside *Telegram* they had no established means of contact. This arrangement proved efficient from a technical perspective but revealed its limits when confronted with risk management. A turning point occurred when the Rebirth Reborn VPN site operated by Neti suffered a severe external attack. Lacking even basic protective measures such as a CAPTCHA, the service was paralysed by a database spam assault and subsequently taken offline. Neti then attempted to purchase a commercial security module but was defrauded, sustaining financial losses, and ultimately lost all motivation to continue the operation. As a result, he effectively disappeared from the online scene.

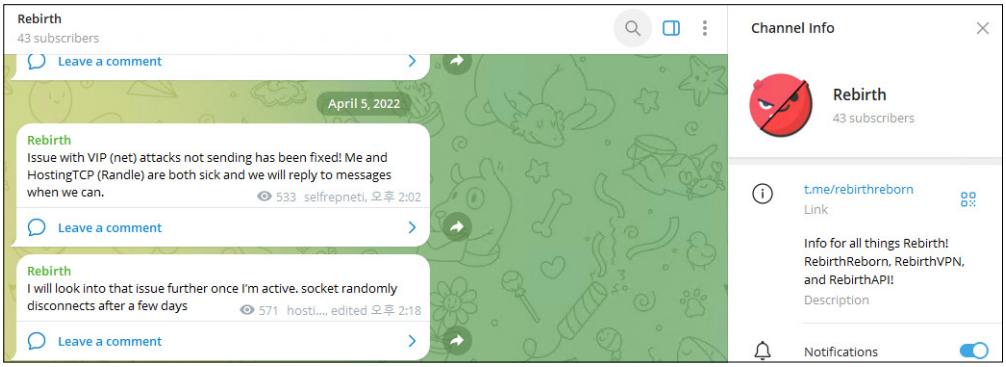

Figure 25: Neti's message notifying temporary suspension of operation.

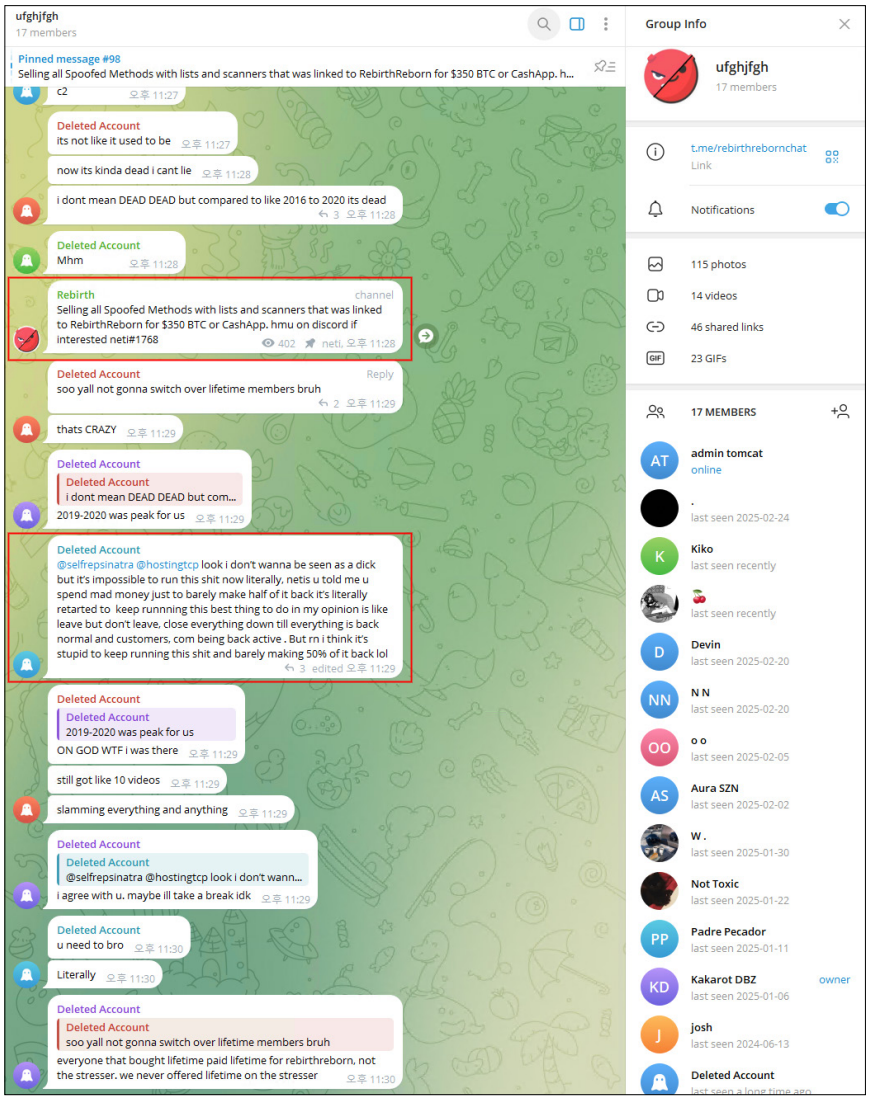

Figure 26: Confusion among buyers due to suspension announcement.

Under these circumstances, HostingTCP saw a need both to preserve the team's legacy and to pursue an independent direction. Leveraging the authentication server and stresser service expertise he had already developed, he single-handedly launched a new DDoS brand called Eternal Stresser in early 2023. While the service inherited critical elements of the Rebirth infrastructure, it operated as a separate platform and quickly built a customer base exceeding 500 users.

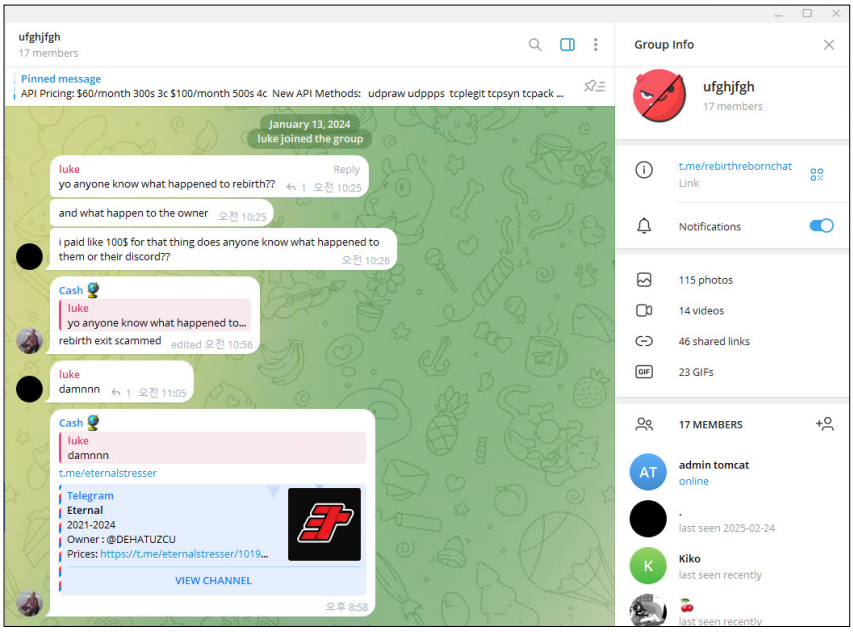

Figure 27: Existing Rebirth Chat users redirected to Eternal Chat.

Consequently, Eternal Stresser emerged as an independent service in which HostingTCP fully took on the technical and operational functions vacated by Neti's sudden departure. This development was not a simple spin-off but rather a reorganized survival strategy arising from the crisis that overtook the original project.

7. CURRENT STATUS OF REBIRTH REBORN

The x86 (a.k.a. Rebirth Reborn) malware sample analysed in this study was first identified on 26 January 2025, a finding that contradicts earlier reports indicating that the Rebirth Reborn service terminated in November 2022. Notably, before the operator Neti abandoned the service, he reportedly sold the complete botnet source code in exchange for cryptocurrency, making it highly likely that the code was later shared among multiple attackers. Detailed examination of the 2025 x86 sample reveals an absence of any self-replication capability; installation is possible only through remote-code-execution exploits, which means the malware cannot propagate autonomously and must be deployed intentionally.

These observations support a single conclusion: independent attackers with no direct affiliation to the CTX-5341 group are distributing malware derived from the Rebirth Reborn DDoS bot source code. The case studied herein accords with this scenario. The x86 sample communicated with a command-and-control server located in Germany, and the server's web interface was presented in Chinese.

```
HTTP/1.0 500 Internal Server Error
Connection: close
Content-Type: text/html; charset=UTF-8
Date: Thu, 06 Feb 2025 13:37:04 GMT
Server: Apache/2.4.62 (Debian)
Content-Length: 0

<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>没有找到站点</title>
<style>
...
</style>
</head>
```

```
<body>
  <div class="main">
    <div class="title">没有找到站点</div>
    <div class="content">
      <p class="t1">您的请求在Web服务器中没有找到对应的站点！</p>
      <p class="t2">可能原因：</p>
      <ol>
        <li>您没有将此域名或IP绑定到对应站点！</li>
        <li>配置文件未生效！</li>
      </ol>
      <p class="t2">如何解决：</p>
      <ol>
        <li>检查是否已经绑定到对应站点，若确认已绑定，请尝试重载Web服务；</li>
        <li>检查端口是否正确；</li>
        <li>若您使用了CDN产品，请尝试清除CDN缓存；</li>
        <li>普通网站访客，请联系网站管理员；</li>
      </ol>
    </div>
  </div>
</body>
```

Table 4: Response from C2 server (77.90.7.228:80).

It is unlikely that either Neti or HostingTCP, the former operators of Rebirth Reborn, established a Chinese-language operating environment. Rather, the evidence suggests that a completely different attacker reconstructed a new threat by reusing the Rebirth Reborn source code. This conclusion is grounded not only in intelligence reporting but also in lineage analysis of the malware code and clearly illustrates how publicly released malware can be re-purposed for new attacks.

8. EFFECTIVENESS OF THE LINEAGE ANALYSIS TECHNOLOGY

8.1. Expansion of botnet tracking

During the lineage analysis of Rebirth Reborn, we identified a new variant, designated 'Batman QBOT', that exhibits characteristic features of the Qbot family. This specimen appears to incorporate several attack modules authored by the actor Self-Rep-NeTiS while retaining the traditional Qbot code base. Notably, the sample employs the signatures 'Yakuza' and 'Scarface1337' and embeds the byte string `Self Rep xxxxing NeTiS and Thisity On Ur xxxxInG FoReHeAd We BiG L33T HaxErS` in its code, thereby confirming the genealogical relationship. The sample was first reported in Germany in early December 2024, and evidence indicates that its source code was distributed through the LeakForum community.

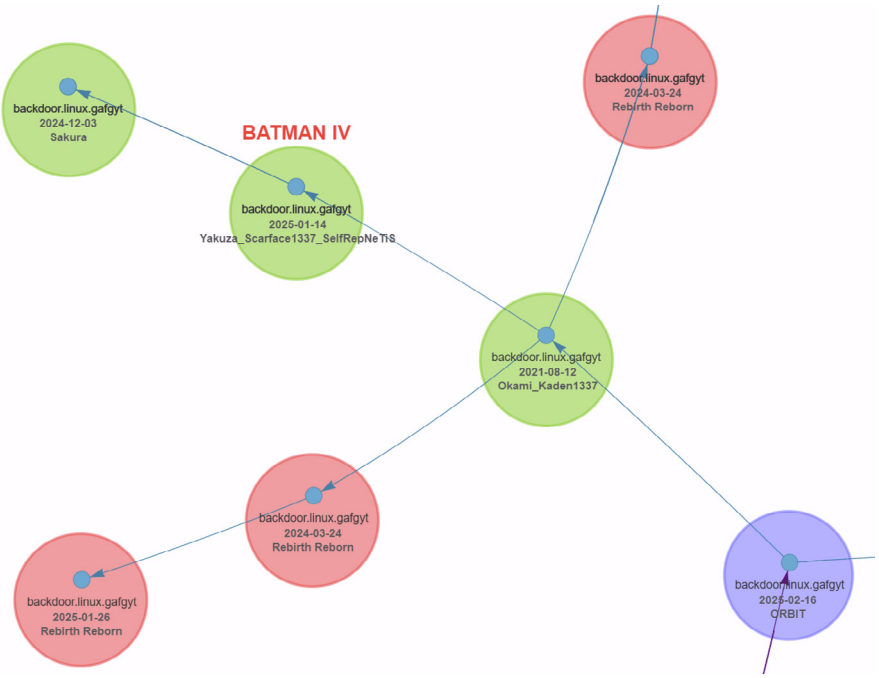

Figure 28: Batman QBOT identified in the tracked lineage graph.

```
void init_rand(int param_1)

{
  uint local_14;

  Q._4_4_ = param_1 + -0x61c88647;
  Q._8_4_ = param_1 + 0x3c6ef372;
  Q._0_4_ = param_1;
  for (local_14 = 3; (int)local_14 < 0x1000; local_14 = local_14 + 1) {
    *(uint *)(Q + local_14 * 4) =
         *(uint *)(Q + (local_14 - 3) * 4) ^ *(uint *)(Q + (local_14 - 2) * 4) ^ local_14 ^
         0x9e3779b9;
  }
  return;
}
```

Figure 29: Qbot-style init_rand function found in Batman.

```
"72qeggInemBIQ5uJc1jQ",
"zwcfbtGDTDBWImROXhdn",
"w70uUC1UJYZoPENznHXB",
"EoXLAf1xXR7j4XSs0JTm",
"lgKjMnqBZFEvPJKpRmMj",
"lSvZgNzxkUyChyxw1nSr",
"VQz4cDTxV8RRrgn00toF",
"YakuzaBotnet",
"Scarface1337",
"\x53\x65\x6c\x66\x20\x52\x65\x70\x20\x46\x75\x63\x6b\x69\x6e\x67\x20\x4e\x65\x54\x69\x53\x20\x61\x6e\x64\x20\x54
//Self Rep    king NeTiS and Thisity 0n Ur    kInG FoReHeAd We BiG L33T HaxErS
"/x38/xFJ/x93/xID/x9A/x38/xFJ/x93/xID/x9A/x38/xFJ/x93/xID/x9A/x38/xFJ/x93/xID/x9A/x38/xFJ/x93/xID/x9A/x38/xFJ/x93
"\x77\x47\x5E\x27\x7A\x4E\x09\xF7\xC7\xC0\xE6\xF5\x9B\xDC\x23\x6E\x12\x29\x25\x1D\x0A\xEF\xFB\xDE\xB6\xB1\x94\xD6
"8d\xc1x\x01\xb8\x9b\xcb\x8f\0\0\0\01k\xc1x\x02\x8b\x9e\xcd\x8e\0\0\0\01k\xc1x\x02\x8b\x9e\xcd\x8e\0\0\0\01
"\x58\x99\x21\x58\x99\x21\x58\x99\x21\x58\x99\x21\x58\x99\x21\x58\x99\x21\x58\x99\x21\x58\x99\x21\x58\x99\x21\x58
                char *STD2_STRING = randstrings[rand() % (sizeof(randstrings) / sizeof(char *))];
            if (a >= 50)
                {
            send(iSTD_Sock, STD2_STRING, STD2_SIZE, 0);
             connect(iSTD_Sock,(struct sockaddr *) &sin, sizeof(sin));
            if (time(NULL) >= start + secs)
            {
            close(iSTD_Sock);
            _exit(0);
            }
            a = 0;
            }
            a++;
            }
            }
```

Figure 30: 'Self Rep NeTiS' string found in Batman's Randhex function.

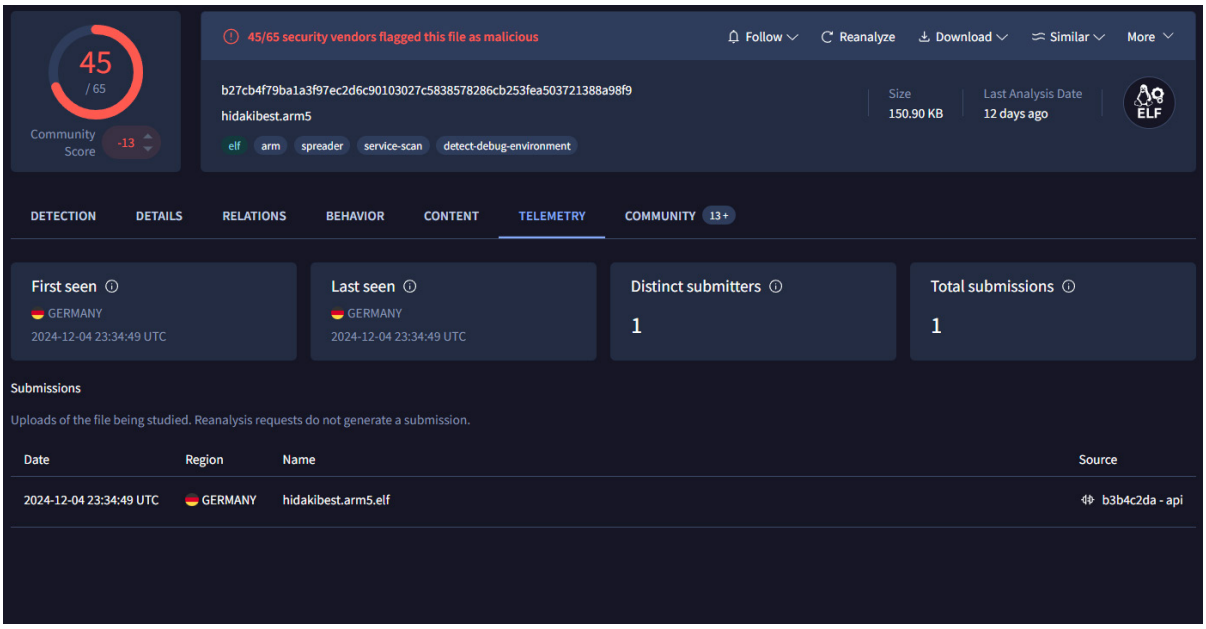

Figure 31: Telemetry logs from VirusTotal [9].

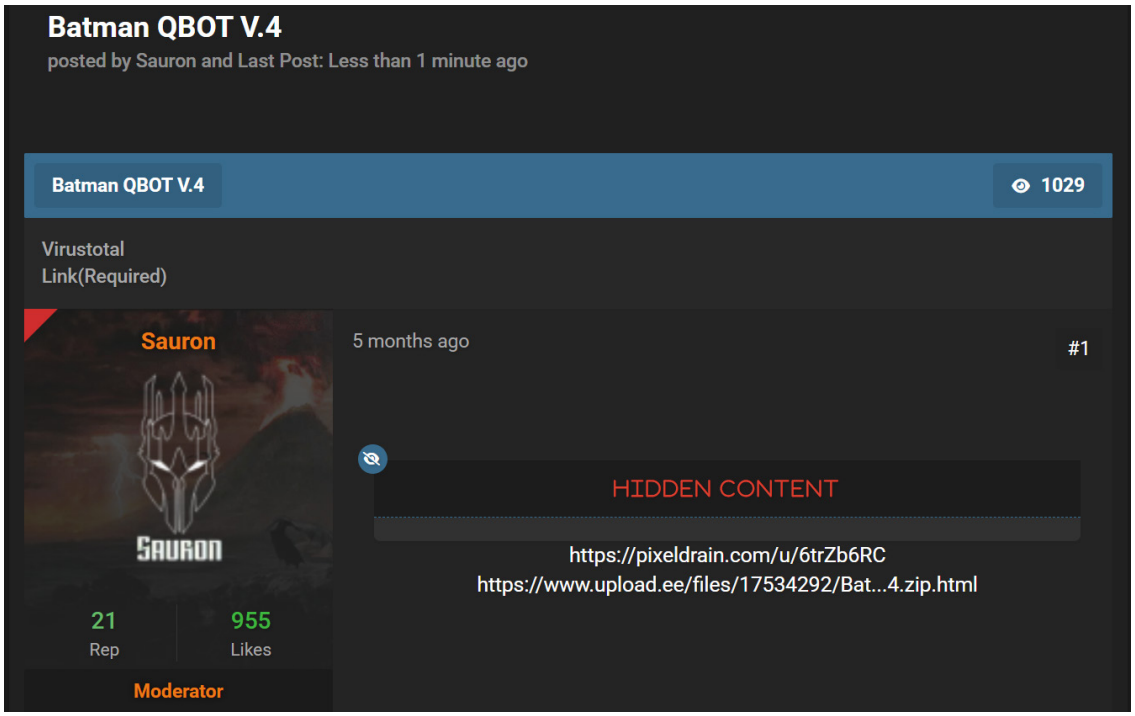

Figure 32: Source code of Batman uploaded to LeakForum [10].

8.2 Potential of the framework

The lineage analysis framework decompiles malware, disassembles it at the function level, and then produces an automated lineage graph by quantifying the similarity among individual functions. This architecture proves highly effective in the IoT botnet ecosystem, where vast numbers of variants coexist, because it greatly reduces repetitive analyst workload and markedly accelerates variant identification.

Rather than merely visualizing code similarity, the framework offers the following advanced analytical capabilities:

- Automatic identification of core functions: the framework pinpoints routines that persist across a family, enabling analysts to focus first on essential logic.
- Family partitioning and parallel tree generation: even within a single family the framework can reveal functional branches and construct parallel trees that reflect internal diversification.
- Signature design support: analysts can transform function-level patterns directly into detection rules that integrate with YARA, Suricata, and related systems.

The framework visualizes code provenance, reuse, and the extent of modification through similarity analysis, bringing objectivity and automation to a task that once relied on analyst intuition.

When these functions are fully integrated into an intuitive user interface and exposed through a RESTful API, the framework can evolve from an internal analysis tool into a software-as-a-service platform for corporate customers. In that role it would deliver lineage-centric threat intelligence, raising both accuracy and efficiency for internet service providers, managed security service providers, and content delivery networks that face constant IoT-based attacks.

8.3 Future work

Although the lineage analysis framework is designed around high analytical precision and automated visualization, several technical and operational challenges must still be addressed for its practical deployment in real-world environments.

Precision enhancement against evasion techniques

The current framework evaluates similarity primarily through function-level embedding vector hash-based comparison. While this approach is effective for lightweight comparisons and fast computation, it suffers a significant drop in precision when attackers apply code obfuscation, virtualization, or control-flow transformation techniques. To overcome this limitation, a shift toward a multi-metric analysis model – rather than reliance on a single metric – is required. By fusing various indicators as outlined below, it becomes possible to detect correlations among seemingly dissimilar variants that originate from the same codebase.

- Comparison based on control-flow graph similarity.
- Behavioural flow analysis through API call sequencing and library dependency extraction.
- Extraction of semantically meaningful features such as hard-coded constants and string literals.
- Enhancement of obfuscation detection sensitivity via code layout and entropy distribution analysis..

Language-agnostic analytical capability

Currently, the framework is built on Ghidra's analysis engine, which provides excellent accuracy for C/C++ binaries and reasonable support for traditional compiled languages. However, it faces challenges with certain modern language ecosystems: IL-based languages, including .NET (CIL), require specialized tooling for optimal analysis, Java (JVM bytecode) benefits from dedicated decompilers, and newer AOT-compiled languages like Go, Rust and Nim present varying levels of complexity for static analysis. While Ghidra has made significant improvements in supporting some of these languages (particularly Go in recent versions), comprehensive analysis of diverse language platforms remains challenging. As attackers increasingly adopt languages that complicate static analysis, for example Go and Rust, the framework would benefit from enhanced language coverage. The following strategic enhancements are required to overcome language-specific barriers in variant analysis and lineage tracking:

- Modularize language-specific decompiler integrations (e.g. integrate with ILSpy, Ghidra-go, Binary Ninja, etc.).
- Customize structural analysers and function extractors to match each language's characteristics.
- Build a unified IR framework to support multiple front ends seamlessly.

Performance and operational scalability for real-time response

Currently, the lineage analysis framework is optimized for variant investigation scenarios. However, operational environments such as managed security service providers (MSSPs), computer security incident response teams (CSIRTs), and security operations centres (SOCs) require seamless integration with real-time detection capabilities. To meet this demand, the following technical improvements are necessary. Furthermore, by extending beyond internal use to provide analysis results externally as lightweight reports or threat feeds, the framework can evolve into a lineage-centric threat intelligence ecosystem.

- Parallel processing optimization for simultaneous analysis of thousands of samples (multi-core and distributed processing).
- Version tracking within malware families via sample-wise lineage change history and re-comparison capabilities.
- Lightweight RESTful/GraphQL API design for enhanced interoperability with external systems.
- Event-driven automated analysis triggers for integration with EDR and SIEM platforms.

These improvements are key to transforming the lineage analysis capability from a mere analytical tool into a field-deployable intelligence platform that simultaneously supports operational response and threat tracking.

REFERENCES

[1] Geenens, P. New DemonBot Discovered. Radware Blog. 25 October 2018. https://www.radware.com/blog/security/new-demonbot-discovered/.

[2] MiraiLovers.io. Index of /HELL-ARCHIVE/BOTNETS/QBOT. 22 July 2024. https://mirailovers.io/HELL-ARCHIVE/BOTNETS/QBOT/.

[3] Self-Rep-NeTiS. Demon V5 CNC. Pastebin. 26 January 2019. https://pastebin.com/GPmBQcX0.

[4] rebirthstresser. Instagram. April 2020. https://www.instagram.com/p/B-hqd-plF5V/.

[5] rebirthstresser. Shoppy.gg. https://shoppy.gg/@rebirthstresser.

[6] rebirthstresser. Instagram. https://www.instagram.com/p/B6en9NWnag8.

[7] SpyHackerz.org. Botnet DDoS Script Demon v5. https://spyhackerz.org/forum/threads/botnet-ddos-script-demon-v5.68742/post-2297078.

[8] rebirthstresser. Instagram. https://www.instagram.com/p/B-gAgj9Fs0v/.

[9] VirusTotal. File b27cb4f79ba1a3f97ec2d6c90103027c5838578286cb253fea503721388a98f9. https://www.virustotal.com/gui/file/b27cb4f79ba1a3f97ec2d6c90103027c5838578286cb253fea503721388a98f9/telemetry.

[10] LeakForum. Batman QBOT V.4. January 2025. https://leakforum.io/Thread-Batman-QBOT-V-4.