Kamasers Analysis: A Multi-Vector DDoS Botnet Targeting Organizations Worldwide

By Achmad Adhikara, 4OURUP and GridGuardGhoul

Published: 2026-03-25 · Archived: 2026-04-06 01:11:04 UTC

DDoS attacks are no longer only an infrastructure problem. They can quickly turn into a **business issue**, affecting uptime, customer experience, and operational stability. Kamasers is a strong example of this new reality, with broad attack capabilities and resilient command-and-control mechanisms that allow it to remain active under pressure.

Let's explore the Kamasers botnet through both **technical and behavioral analysis**, looking at the commands it receives, the geographic distribution of its attacks, and the functions implemented in the malware sample. Together, these elements help reveal how Kamasers operates and why it poses a serious threat to organizations worldwide

Key Takeaways

- Kamasers is a sophisticated **DDoS botnet** that supports both application-layer and transport-layer attacks, including HTTP, TLS, UDP, TCP, and GraphQL-based flooding.
- The malware can also act as a **loader**, downloading and executing additional payloads, which raises the risk of **further compromise, data theft, and ransomware deployment**.
- Its **C2 infrastructure is resilient**, using a Dead Drop Resolver (DDR) through legitimate public services such as GitHub Gist, Telegram, Dropbox, Bitbucket, and even Etherscan to retrieve active C2 addresses.
- Analysis showed that **Railnet ASN** repeatedly appeared in malicious activity tied to multiple malware families, making it a notable infrastructure element in the broader threat landscape.
- Kamasers was observed being distributed through **GCleaner** and **Amadey**, showing that it fits into established malware delivery chains.
- The botnet's activity is **international**, with strong submission visibility in **Germany and the United States**, while targeting extends across sectors including **education, telecom, and technology**.

The Business Risk Behind Kamasers

Kamasers is a flexible attack platform that can turn compromised enterprise systems into operational liabilities, external attack infrastructure, and potential entry points for deeper compromise:

- **Corporate infrastructure can be turned against others:** Infected enterprise systems may be used to launch DDoS attacks on third parties, creating reputational, contractual, and even legal risk for the

organization.

- **A broader incident can follow quickly:** Because Kamasers can function as a loader, a single infection may lead to additional payload delivery, raising the risk of data theft, ransomware, and deeper intrusion.
- **Visibility gaps become harder to defend:** The malware uses legitimate public services to retrieve C2 information, making malicious communication more difficult to detect and increasing the chance of delayed response.
- **Response costs rise fast:** Investigating infected hosts, validating external impact, restoring systems, and handling possible IP blacklisting can create significant operational and financial strain.
- **Business trust can be affected early:** If company infrastructure is linked to malicious traffic, customers, partners, and providers may react before the full incident is even understood.

Kamasers highlights a serious enterprise risk: attackers can use resilient C2 discovery, flexible attack methods, and follow-on payload delivery to turn a single compromise into an incident with operational, financial, compliance, and reputational consequences.

Kamasers Threat Overview

Kamasers is a malware botnet family designed to carry out DDoS attacks using both application-layer and transport-layer vectors. It supports HTTP GET/POST floods, API-targeted attacks, defense evasion techniques, TLS handshake exhaustion, connection-holding methods, as well as UDP and TCP floods. Infected nodes receive commands from the command-and-control infrastructure and generate the corresponding traffic. In addition, Kamasers can also function as a loader, downloading and executing files from the network.

ANY.RUN previously observed activity associated with **Udados**, which is most likely an evolution or updated version of Kamasers. As such, Udados can be considered part of the Kamasers family.

You can find public sandbox analysis sessions related to the Kamasers family with the following Threat Intelligence Lookup query:

threatName:”kamasers”


ANY.RUN’s sandbox sessions related to the Kamasers attacks
ANY.RUN’s sandbox sessions related to the Kamasers attacks displayed inside TI Lookup

If a corporate host becomes part of a botnet and is used to carry out DDoS attacks, the organization may face financial risks related to incident response, system recovery, network costs, and potential contractual penalties, as well as regulatory scrutiny if inadequate security measures are identified, especially in cases involving data compromise.

An additional risk stems from the malware’s ability to act as a loader, downloading and executing third-party payloads. This increases the likelihood of further intrusion, data exfiltration, ransomware deployment, and the resulting operational and reputational damage.

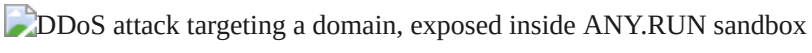
C2 and Infrastructure

As part of the analysis, it was observed that the bot received the !httpbypass control command, which initiates an HTTP flood attack against a specified URL with defined intensity and duration parameters. After completing the attack, the bot reported its status and returned to standby mode.

View analysis session


Communication between the infected host and the C2 server
Communication between the infected host and the C2 server

In the sandbox analysis session, we can see how a DDoS attack targets a domain:


DDoS attack targeting a domain, exposed inside ANY.RUN sandbox
DDoS attack targeting a domain, exposed inside ANY.RUN sandbox

In a number of analysis sessions, the command-and-control server was used not only to coordinate DDoS activity, but also to deliver additional payloads. Specifically, the bot received the *!download*command, after which it downloaded and executed a file from an external domain, then confirmed successful session completion to the C2 server:

View analysis session


Example of a C2 command used to download a malicious file
Example of a C2 command used to download a malicious file

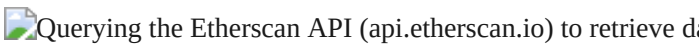
In one observed case, the bot received the *!descargar* command, the Spanish-language equivalent of *!download*, to retrieve an executable file from an external domain.

View analysis session with C2 command in Spanish

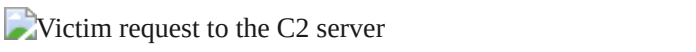

C2 command in Spanish used to download a malicious file
C2 command in Spanish used to download a malicious file observed inside ANY.RUN sandbox

In some cases, the Kamasers botnet was observed using public blockchain infrastructure as an auxiliary mechanism for obtaining the C2 address. Specifically, infected hosts queried the **Etherscan API**(api.etherscan.io) to retrieve data containing the URL of the command-and-control server:

View session querying the Etherscan API


Querying the Etherscan API (api.etherscan.io) to retrieve data
Querying the Etherscan API (api.etherscan.io) to retrieve data

After obtaining the URL, the bot connects to the C2 server and sends information about its ID, command execution status, bot version, privileges on the infected host, C2 discovery source, and system information:


Victim request to the C2 server
Victim request to the C2 server

In a number of cases, Kamasers uses public services, including **GitHub**, as an auxiliary source of configuration:

Check how Kamasers uses public services

Behavioral analysis of Kamasers showed that the botnet frequently establishes connections to IP addresses associated with **Railnet LLC's ASN**.

Railnet is regularly mentioned in public reporting as a legitimate front for the hosting provider **Virtualine**. This provider is known for the absence of KYC procedures, and some research has noted that the associated infrastructure is used to host malicious services and facilitate attacks.

Railnet infrastructure has previously been observed in campaigns targeting both government and private-sector organizations across several European countries, including Switzerland, Germany, Ukraine, Poland, and France.

There are also documented cases of **Railnet** infrastructure being used to distribute other malware families, including **Latrodectus**, which a number of reports link to activity associated with groups such as **TA577**.

At the time of analysis, **ANY.RUN** data showed that **Railnet's ASN** consistently appeared in reports tied to a wide range of malicious activity and was being used by multiple malware families. These were not isolated incidents, but a recurring pattern: the same ASN was repeatedly involved across different campaigns, making it a convenient infrastructure hub for threat actors.

The current picture of **Railnet** activity can be quickly verified using **ANY.RUN's Threat Intelligence Lookup**. Searching by ASN makes it possible to assess how extensively it is involved in malicious chains, which malware families interact with it, and how the nature of that activity changes over time:

destinationIpAsn:"railnet"

Query for RAILNET ASN in ANY.RUN's TI Lookup

In the analyzed sandbox sessions, Kamasers was distributed via **GCleaner** and **Amadey**, a delivery pattern that has also been observed in other DDoS campaigns.

Attack Geography and Targeting

Among the observed **DDoS** targets were companies in the **LATAM** region. However, according to **ANY.RUN's threat intelligence** data, the targeting profile is broader: the education sector is affected most

often, along with telecommunications and technology organizations.

Query in ANY.RUN TI to search for the Kamasers malware family

By geographic distribution of observed submissions, the largest share comes from **Germany** and the **United States**, with separate cases also recorded in **Poland** and other countries. During the analysis, control commands in **Spanish** were also observed. This may indirectly suggest that the botnet may have originated from, or evolved within, a Spanish-speaking operator environment, although its actual activity is clearly international in scope.

It is also important to consider that the botnet uses the infrastructure of infected hosts to carry out attacks. If corporate systems are compromised, the organization may not only become a potential target itself, but also inadvertently serve as a source of attacks against third parties. This creates reputational risks, the possibility of IP address blacklisting, and additional financial costs related to investigation and infrastructure recovery.

Technical Breakdown of Kamasers

To better understand the Kamasers botnet architecture, a detailed sample analysis was conducted. The starting point was the sample from this **ANY.RUN sandbox** session:

Check analysis session

ANY.RUN's analysis session used as a starting point for technical investigation

This was followed by reverse engineering of the binary. The analysis focused primarily on how the malware receives and processes commands from the C2 server, as well as the attack capabilities implemented in the sample.

After launch, the malware begins retrieving commands through a **Dead Drop Resolver** mechanism. It uses public services such as **GitHub Gist, Telegram, Dropbox, and Bitbucket** as intermediary sources. From these sources, the bot extracts the address of the real C2 server and then establishes a connection to it.


The bot validates the format of the command sent by the C2 server

The bot validates the format of the command sent by the C2 server

Command processing takes place in several stages. First, the bot verifies that the command format is valid. All valid commands must begin with the "!" character. If this prefix is missing, the command is rejected and not executed.


Code for the handler caching mechanism

Code for the handler caching mechanism

After validating the prefix, the bot matches the command against an internal handler table. The analysis showed that Kamasers uses a **handler caching mechanism**. If the previously used handler matches the current command index, the bot takes a fast path without performing another lookup. Otherwise, it triggers the dynamic resolution routine.


Pseudocode of the flowchart showing command receipt and handler caching

Pseudocode of the flowchart showing command receipt and handler caching

This mechanism can be briefly described as shown in the pseudocode above.

One of the most illustrative commands is !udppro. It implements a high-speed UDP flood with support for source IP spoofing. Code analysis shows the standard sequence for creating a UDP socket via the **WinSock API** using the AF_INET, SOCK_DGRAM, and IPPROTO_UDP parameters.


Disassembled code for the “!udppro” command
Disassembled code for the “!udppro” command

After initializing the socket, the malware configures the packet transmission parameters. Support for **IP spoofing** enables **reflection** and **amplification** attacks through public **NTP** and **DNS** servers. In such scenarios, the victim receives responses that are significantly larger than the original requests, leading to a sharp increase in load.

The *!download* command is also present, implementing a **Download & Execute** mechanism. The bot retrieves an executable file from the specified URL, checks for the MZ signature, allocates memory, maps the sections, and transfers execution to the entry point. If successful, it sends a task completion message; if an error occurs, it generates a failure notification.


Bot status messages related to the download process
Bot status messages related to the download process

Implementation of Dead Drop Resolver Channels

Kamasers uses four **Dead Drop Resolver** channels: **GitHub Gist**, a **Telegram bot**, a file hosted on **Dropbox**, and a **Bitbucket** repository. Importantly, links to these services are not stored in the sample in plain form. Instead, they are constructed and unpacked dynamically at runtime, which is why such strings do not appear during static analysis of the binary.

The **Dead Drop Resolver (DDR)** mechanism serves as an intermediary layer between the bot and the primary C2 server. After launch, the malware sequentially sends HTTP GET requests to each of the public resources. The content hosted there contains the current address of the command-and-control server. Once a response is received, the bot extracts the C2 address and establishes a direct connection to continue receiving commands.

If the first source returns a valid address, no further requests are made. If the connection fails or the response is invalid, the bot automatically falls back to the next channel: **Telegram**, then **Dropbox**, and finally **Bitbucket**.


DDR links in the Kamasers codebase
DDR links in the Kamasers codebase

All of these resources ultimately point to the same C2 infrastructure:


GitHub Gist content used by Kamasers as DDR
GitHub Gist content used by Kamasers as DDR


Bitbucket content used by Kamasers as DDR
Bitbucket content used by Kamasers as DDR


Fallback domains used if the DDR links are unavailable

Fallback domains used if the DDR links are unavailable

If none of the **DDR channels** responds, the malware falls back to a built-in list of backup domains.

Catching Kamasers Early: A Practical Detection Approach

Kamasers shows how a single malware infection can quickly turn into a broader business problem. Beyond DDoS activity, the botnet can also download and execute additional payloads, increasing the risk of deeper compromise.

For security teams, the challenge is not only spotting the malware itself but also understanding whether an infected host is being used for external attacks, communicating with resilient C2 infrastructure, or pulling in follow-on payloads.

Early detection depends on moving quickly from suspicious network activity to confirmed malicious behavior.

1. Monitoring: Spot Malicious Infrastructure and Unusual Network Behavior Early

Kamasers relies on external infrastructure to receive commands, retrieve C2 addresses, and in some cases download additional payloads. It also uses public services such as GitHub Gist, Telegram, Dropbox, Bitbucket, and even Etherscan as part of its Dead Drop Resolver logic.

Monitoring for suspicious outbound connections, newly observed infrastructure, and repeated communication with known malicious hosting can help teams detect activity before the infection leads to larger operational impact.

Actionable IOCs delivered by TI Feeds to your existing stack

ANY.RUN's Threat Intelligence Feeds help surface suspicious indicators early, giving SOC teams faster visibility into malicious domains, IPs, and infrastructure patterns linked to emerging threats.

2. Triage: Confirm Botnet Activity with Behavior-Based Analysis

With threats like Kamasers, static detection alone may not show the full risk. A suspicious file may appear inconclusive until its real behavior is observed during execution.

Running the sample inside the ANY.RUN interactive sandbox makes it possible to confirm the full execution flow, including:

- retrieval of C2 data through Dead Drop Resolver channels
- connection to the active command-and-control server
- receipt and execution of DDoS commands
- download-and-execute behavior through commands like !download or !descargar
- status reporting back to the C2 infrastructure

Relevant IOCs automatically gathered in one tab inside ANY.RUN sandbox

This helps teams quickly determine whether the malware is only participating in DDoS activity or whether it also creates risk of further payload delivery and deeper compromise.

3. Threat Hunting: Pivot from One Sample to Related Infrastructure

Once Kamasers is confirmed, the next step is understanding how far the activity may extend.

Using ANY.RUN's Threat Intelligence Lookup, teams can pivot from the initial sample to uncover related infrastructure, connected sessions, and recurring patterns across the broader campaign.

This makes it possible to:

- identify other samples tied to the Kamasers family
- trace infrastructure linked to the botnet's C2 activity
- investigate repeated use of ASN-linked hosting such as Railnet
- expand detection based on shared behavior and network indicators

threatName:"kamasers"

ANY.RUN's sandbox sessions related to the Kamasers attacks displayed inside TI Lookup

By pivoting from one confirmed sample, security teams can turn a single investigation into broader visibility across related botnet activity.

Conclusion

Kamasers is a sophisticated **DDoS botnet** with a well-designed architecture. Its use of a **Dead Drop Resolver** through legitimate services makes its C2 infrastructure highly resilient to takedown efforts. The presence of **16 different attack methods**, including modern vectors such as **GraphQL** and **HTTP bypass**, along with advanced implementations of classic techniques, makes **Kamasers** a highly versatile tool for carrying out DDoS attacks.

For business leaders, Kamasers shows that resilient, multi-vector botnets can threaten not only infrastructure, but also uptime, customer experience, and revenue-critical operations.

Power faster, clearer investigations with ANY.RUN ➜

About ANY.RUN

ANY.RUN, a leading provider of interactive malware analysis and threat intelligence solutions, fits naturally into modern SOC workflows and supports investigations from initial alert to final containment.

It allows teams to safely execute suspicious files and URLs, observe real behavior in an interactive environment, enrich indicators with immediate context through TI Lookup, and continuously monitor emerging infrastructure using Threat Intelligence Feeds. Together, these capabilities help reduce uncertainty, accelerate triage, and limit unnecessary escalations across the SOC.

ANY.RUN also meets enterprise security and compliance expectations. The company is SOC 2 Type II certified, reinforcing its commitment to protecting customer data and maintaining strong security controls.

Complete List of Kamasers Commands

Command	Purpose
!stop	Stops the current operation. Closes sockets, terminates attack threads, and clears buffers.
!download	Downloads and executes a file. Retrieves a PE file over HTTP, verifies it, and launches it. Also detects whether the file has been removed by antivirus software.
!visiturl	Sends a basic HTTP GET request to the specified URL to generate traffic or check availability.
!httpget	Basic HTTP GET flood implementation. Spawns several dozen threads with minimal randomization.
!httpgetpro	Advanced HTTP GET flood. Spawns hundreds of threads, randomizes the User-Agent, Referer, URL paths, and parameters. Uses keep-alive connections.
!httppost	HTTP POST flood. Sends POST requests with randomized headers and payloads, creating load on server-side data processing.
!tlsflood	TLS handshake flood. Initiates SSL/TLS handshakes without completing them, creating load on the server’s cryptographic operations.
!httpbypass	HTTP attack with defense evasion. Uses WAF/CDN bypass techniques such as header manipulation, payload encoding, and request fragmentation.
!graphql	GraphQL API flood. Sends deeply nested GraphQL queries that create exponential load on the server parser.
!httphulk	HULK attack (HTTP Unbearable Load King). Applies maximumrandomization to all HTTP request parameters to bypass caching and rate limiting.
!fastflood	Optimized high-speed flood with minimal overhead, designed to saturate available bandwidth.
!proloris	Professional implementation of Slowloris. Slowly sends partial HTTP headers to exhaust the server’s connection pool.
!slowread	Slow Read attack. Requests a large file and reads it very slowly to tie up server resources.
!udppro	Professional UDP flood with support for IP spoofing and NTP/DNS amplification.

Command	Purpose
!tcppro	Advanced TCP flood. Combines SYN flood, ACK flood, and connection reset techniques to exhaust the TCP state table.
!tcphold	TCP connection holding. Establishes the maximum number of connections while maintaining minimal keep-alive traffic to exhaust server limits.

Indicators of Compromise (IOCs)

- F6c6e16a392be4dbf9a3cf1085b4ffc005b0931fc8eeb5fedf1c7561b2e5ad6b
- Dd305f7f1131898c736c97f43c6729bf57d3980fc269400d23412a282ee71a9a
- hxxp://45[.]151[.]91[.]187/pa[.]php
- hxxp://91[.]92[.]240[.]50/pit/wp[.]php
- 071a1960fbd7114ca87d9da138908722d7f1c02af90ea2db1963915fbe234c52
- hxxp://178[.]16[.]54[.]87/uda/ph[.]php

C2 Infrastructure (DDR):

- gist[.]github[.]com/pitybugak/5d16b75e8bd071e15b04cc9c06dcfafa[.]js
- api[.]telegram[.]org/bot8215158687:AAFgSmsaxfsJozcHIIYPv-HytZ3eCEaUrKg
- dl[.]dropboxusercontent[.]com/s/jqvpmc0kwg6ffi1mineh2/fj[.]txt
- Bitbucket[.]org/serky/repyx/raw/main/fq[.]txt

Fallback domains:

- pitybux[.]com
- ryxuz[.]com
- toksm[.]com
- Boskuh[.]com

Yara rules:

```
rule Kamasers {
    meta:
        description = “Detects Kamasers DDoS botnet”
        author = “ANY.RUN”
        date = “2026-02-11”
        threat = “Kamasers”
    strings:
```

$cmd1 = “!stop” ascii fullword

$cmd2 = “!download” ascii fullword

$cmd3 = “!visiturl” ascii fullword

$cmd4 = “!httpget” ascii fullword

$cmd5 = “!httpgetpro” ascii fullword

$cmd6 = “!httppost” ascii fullword

$cmd7 = “!tlsflood” ascii fullword

$cmd8 = “!httpbypass” ascii fullword

$cmd9 = “!graphql” ascii fullword

$cmd10 = “!httphulk” ascii fullword

$cmd11 = “!fastflood” ascii fullword

$cmd12 = “!proloris” ascii fullword

$cmd13 = “!slowread” ascii fullword

$cmd14 = “!udppro” ascii fullword

$cmd15 = “!tcppro” ascii fullword

$cmd16 = “!tcphold” ascii fullword

$msg1 = “Task completed:” ascii fullword

$msg2 = “Task completed: GraphQL Flood on” ascii fullword

$msg3 = “Task completed: HULK on” ascii fullword

$msg4 = “Task completed: UDPPRO Flood on” ascii fullword

$msg5 = “Task completed: TCPPRO Flood on” ascii fullword

$msg6 = “Task completed: TCP HOLD on” ascii fullword

$msg7 = “Task completed: Download & Execute from” ascii fullword

$msg8 = “Task completed: Visit URL” ascii fullword

$msg9 = “Starting GraphQL Flood on” ascii fullword

$msg10 = “Starting HULK on” ascii fullword

```
        $msg11 = “Starting UDP PRO on” ascii fullword
        $msg12 = “Starting TCP PRO on” ascii fullword
        $msg13 = “Starting TCP HOLD on” ascii fullword
        $msg14 = “Starting Visit URL task on” ascii fullword
        $msg15 = “Runtime error in D&E task:” ascii fullword
        $msg16 = “Unknown exception in DownloadAndExecuteTask” ascii fullword
        $msg17 = “Awaiting task” ascii fullword
        $msg18 = “Downloading file from:” ascii fullword
        $msg19 = “Downloaded file disappeared (AV/EDR?)” ascii fullword
        $msg20 = “Download failed with HRESULT:” ascii fullword
        $msg21 = “HTTP GET Flood” ascii fullword
        $msg22 = “HTTP GET PRO” ascii fullword
        $msg23 = “HTTP POST Flood” ascii fullword
        $msg24 = “HULK_POST” ascii fullword
    condition:
        uint16(0) == 0x5A4D and
        (10 of ($cmd*)) and
        (8 of ($msg*))
}
```

Achmad Adhikara

Threat Hunter at ANY.RUN | + posts

Achmad Adhikara is a threat hunter at ANY.RUN. Former red teamer. I chase threats. I prefer to stay below periscope depth. fnord.

4OURUP

I research malicious activity, attack tactics, and techniques. I analyze cyber threats, process data, and help stay one step ahead of adversaries.

GridGuardGhoul

I am a network security researcher and reverse engineer exploring malware, protocols, and exploits.

achmad-adhikara

Achmad Adhikara

Threat Hunter at ANY.RUN

Achmad Adhikara is a threat hunter at ANY.RUN. Former red teamer. I chase threats. I prefer to stay below periscope depth. fnord.

4ourup

4OURUP

I research malicious activity, attack tactics, and techniques. I analyze cyber threats, process data, and help stay one step ahead of adversaries.

gridguardghoul

GridGuardGhoul

I am a network security researcher and reverse engineer exploring malware, protocols, and exploits.

Source: https://any.run/cybersecurity-blog/kamasers-technical-analysis/