ERMAC V3.0 Banking Trojan: Full Source Code Leak and Infrastructure Analysis

Published: 2025-08-14 · Archived: 2026-04-06 00:40:59 UTC

In March 2024, Hunt.io discovered and obtained the complete ERMAC V3.0 source code, giving us a rare opportunity to study a live and actively maintained Malware-as-a-Service platform from the inside. Full source code leaks of active and operational threats are uncommon, and this one offers a unique chance for attribution, infrastructure mapping, and identifying exploitable weaknesses.

Its earliest versions were built using the leaked Cerberus source code, and by late 2023, version 2.0 had incorporated large portions of the Hook botnet's codebase. The newly uncovered version 3.0 reveals a significant evolution of the malware, expanding its form injection and data theft capabilities to target more than 700 banking, shopping, and cryptocurrency applications.

This post examines the backend, frontend panel, exfiltration server, and builder included in the leak, detailing ERMAC's infrastructure, operational techniques, and multiple vulnerabilities that could be leveraged to disrupt its activity.

Key Takeaways

Our analysis of the ERMAC V3.0 leak uncovered technical details, operational weaknesses, and active infrastructure that defenders can use to disrupt ongoing campaigns.

- Hunt.io obtained the full ERMAC V3.0 source code, including its PHP and Laravel backend, React-based frontend, Golang exfiltration server, and Android builder panel.
- The leak revealed critical weaknesses, such as a hardcoded JWT secret and a static admin bearer token, default root credentials, and open account registration on the admin panel
- Version 3.0 expands targeting to more than 700 financial, shopping, and cryptocurrency apps, adds new form injection methods, an overhauled C2 panel, a new Android backdoor, and AES-CBC encrypted communications
- Analysis with HuntSQL linked the leaked files to active ERMAC C2 panels, exfiltration servers, and builder deployments still operating online.
- Findings confirm ERMAC's role as a maintained Malware-as-a-Service platform, giving operators flexible control over targeting, encryption keys, and campaign creation

Initial Discovery

The investigation began on March 6th, 2024, when our research team, using the AttackCapture™ tool, identified an open directory on 141[.]164[.]62[.]236:443 containing an archive titled Ermac 3.0.zip. This archive held the

complete ERMAC V3.0 source code.

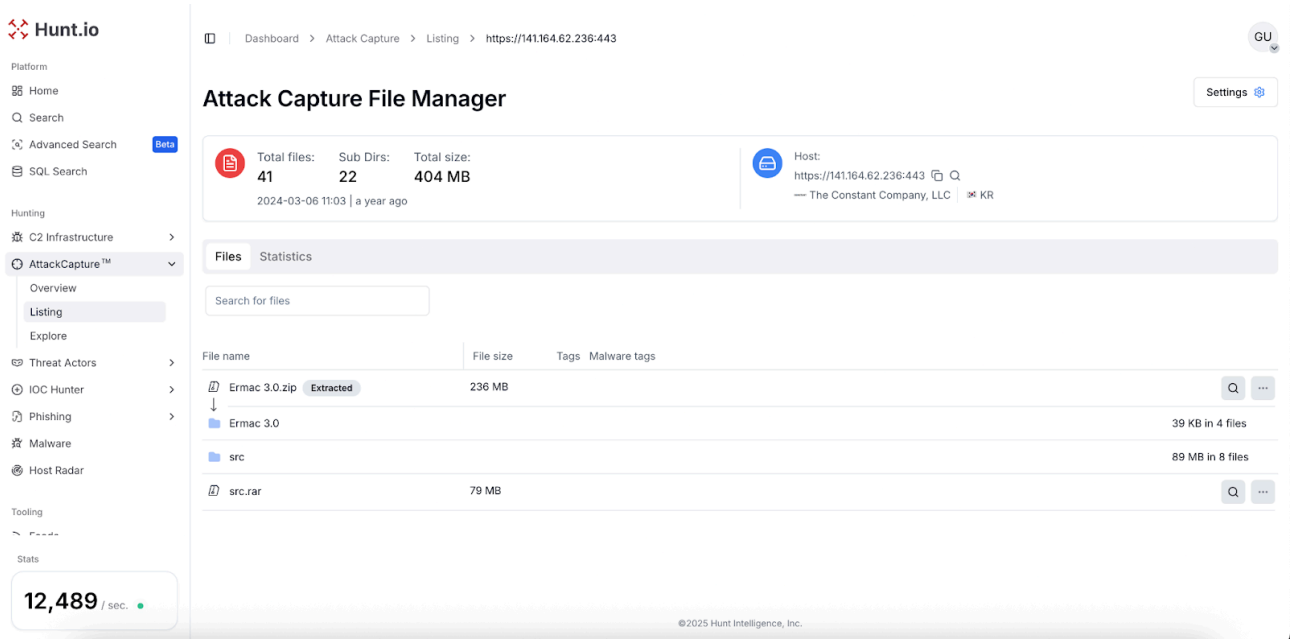


Fig 01: Open directory containing ERMAC's source code, discovered by Hunt.io

Upon further analysis, the Ermac 3.0.zip archive was found to contain 5 different directories:

- **backend**: The ERMAC C2 backend, written in PHP.
- **frontend**: The ERMAC C2 frontend, written in React.
- **golang**: A Go binary that configures an HTTP server used in the exfiltration of data.
- **docker**: Docker configuration files.
- **builder**: Source code to ERMAC 3.0 banking trojan and a panel allowing for compilation and obfuscation of the APK.

Backend C2 Server

Looking into ERMAC's backend source code, we can see it's developed in PHP and uses Laravel as its backend framework. The codebase shares no overlap with the leaked Cerebus source code, with it implementing different routing, infrastructure logic, and features.

Method	Route	Auth Required
POST	/api/v1/sign-in	No
POST	/api/v1/smartInjections/{sessionId}	No
POST	/api/v1/smartInjections/session/list	No
PUT	/api/v1/smartInjections/session/{session}	No

Method	Route	Auth Required
GET	/api/v1/getUserInfo	Yes
POST	/api/v1/injects/getInjectionsList	Yes
POST	/api/v1/injects/createInjection	Yes
DELETE	/api/v1/injects/deleteInjection	Yes
POST	/api/v1/injects/{injection}/editInjection	Yes
POST	/api/v1/sendBotsCommand	Yes
DELETE	/api/v1/deleteBot	Yes
DELETE	/api/v1/deleteAllRemovedApp	Yes
PUT	/api/v1/{bot}/setBotType	Yes
GET	/api/v1/{bot}/commands/getCommandsList	Yes
PUT	/api/v1/{bot}/settings/updateBotSettings	Yes
PUT	/api/v1/{bot}/injects/updateBotInjections	Yes
DELETE	/api/v1/deleteLog	Yes
PUT	/api/v1/editLogComment	Yes
POST	/api/v1/accounts/getAccountsList	Yes
POST	/api/v1/accounts/createAccount	Yes
PUT	/api/v1/accounts/{user}/editAccount	Yes
DELETE	/api/v1/accounts/{user}/deleteAccount	Yes
POST	/api/v1/permissions/getPermissionsList	Yes
PUT	/api/v1/permissions/updatePermission	Yes
POST	/api/v1/counts/getCounts	Yes
POST	/api/v1/counts/getStats	Yes
POST	/api/v1/autoCommands/getAutoCommandsList	Yes
PUT	/api/v1/autoCommands/updateAutoCommand	Yes
POST	/api/v1/search	Yes

Reviewing these routes reveals the scope of control ERMAC operators have over infected devices and stolen data.

The ERMAC backend provides operators the ability to manage victim devices and access compromised data, such as SMS logs, stolen accounts, and device data.

ERMAC primarily leverages form injects for capturing sensitive data, which is done by serving custom form injects through the public/injects directory.

Fig 02: ERMAC's backend directory containing form injects for popular financial apps.

ERMAC targets primarily financial applications with a large focus on mobile banking and cryptocurrency applications with it, capturing sensitive data such as login credentials or credit card data.

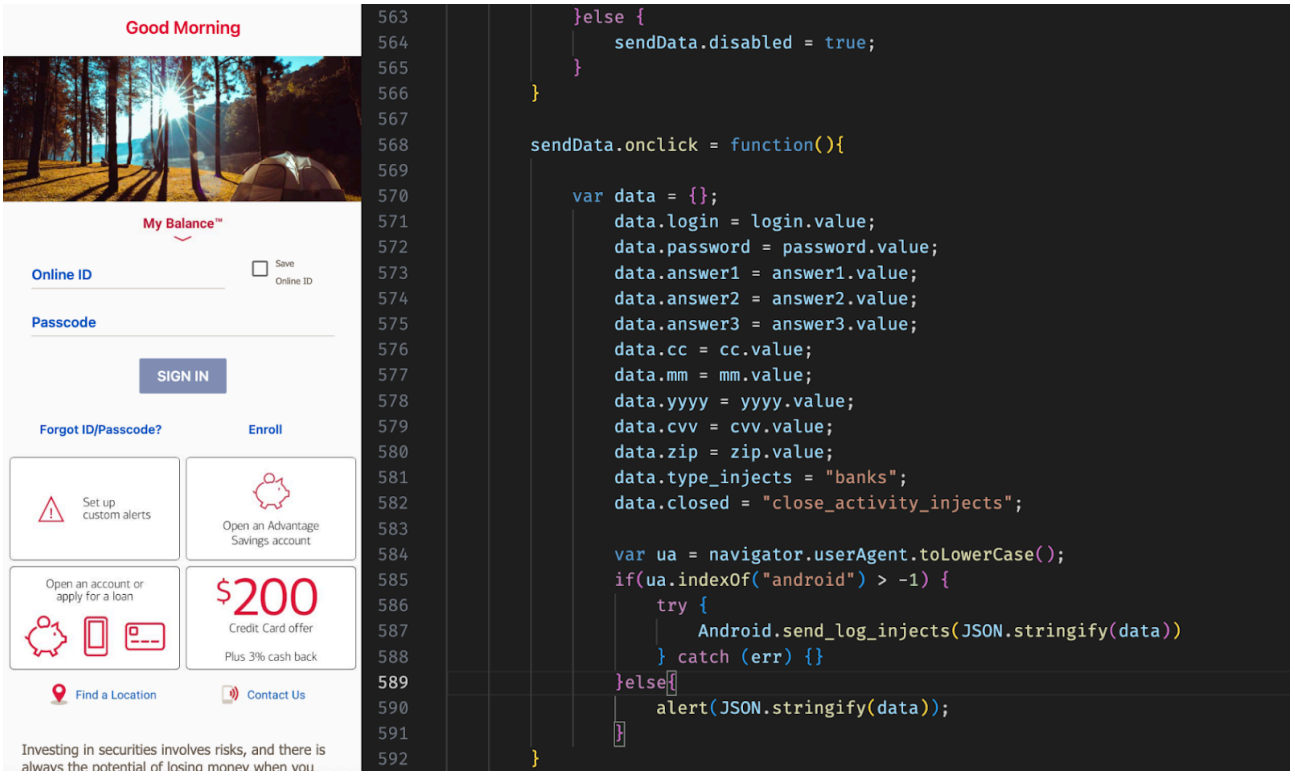


Fig 03: Form Inject mimicking a banking app, and its callback function for exfiltrating form data.

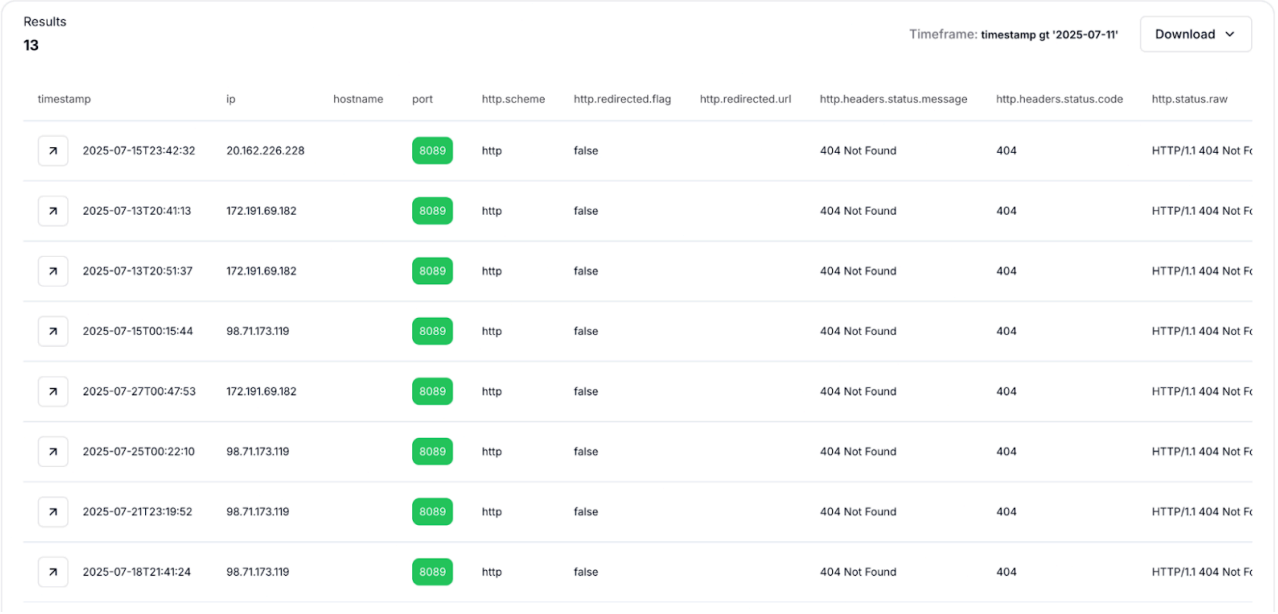
ERMAC will fetch and inject its own custom HTML page into 700+ different applications that uses the callback function `Android.send_log_injects` to exfiltrate form data.

ERMAC's backend uses the `ermac_session` cookie to handle authentication, which allows us to pivot and find other active backends. Using Hunt's SQL search, we are able to find 4 more unique ERMAC C2 servers in the wild:

```
  SELECT * FROM httpv2
WHERE http.headers.bytes.content
LIKE '%ermac_session%'
```

Copy

Output:



Results
13

Timeframe: timestamp gt '2025-07-11' Download

timestamp	ip	hostname	port	http.scheme	http.redirected.flag	http.redirected.url	http.headers.status.message	http.headers.status.code	http.status.raw
2025-07-15T23:42:32	20.162.226.228		8089	http	false		404 Not Found	404	HTTP/1.1 404 Not Fo
2025-07-13T20:41:13	172.191.69.182		8089	http	false		404 Not Found	404	HTTP/1.1 404 Not Fo
2025-07-13T20:51:37	172.191.69.182		8089	http	false		404 Not Found	404	HTTP/1.1 404 Not Fo
2025-07-15T00:15:44	98.71.173.119		8089	http	false		404 Not Found	404	HTTP/1.1 404 Not Fo
2025-07-27T00:47:53	172.191.69.182		8089	http	false		404 Not Found	404	HTTP/1.1 404 Not Fo
2025-07-25T00:22:10	98.71.173.119		8089	http	false		404 Not Found	404	HTTP/1.1 404 Not Fo
2025-07-21T23:19:52	98.71.173.119		8089	http	false		404 Not Found	404	HTTP/1.1 404 Not Fo
2025-07-18T21:41:24	98.71.173.119		8089	http	false		404 Not Found	404	HTTP/1.1 404 Not Fo

Fig 04: ERMAC C2 Servers observed having the ermac_session cookie.

Upon further analysis, ERMAC has three vulnerabilities: a hardcoded JWT token "`h3299xK7gdARLk85rsMyawT7K4yGbxYbkKoJo8gO3lMdl9XwJCKh2tMkdCmeeSeK`", the hardcoded credentials for root with the password changemeplease, and the ability to register an account directly through the API, allowing for full access to the ERMAC admin panel.

Fig 05: Hardcoded default credentials for ERMAC

Frontend Panel

To complement the Backend C2, ERMAC provides a panel for interacting with the backend. ERMAC is observed to have three public versions, with the latest known version being 3.0

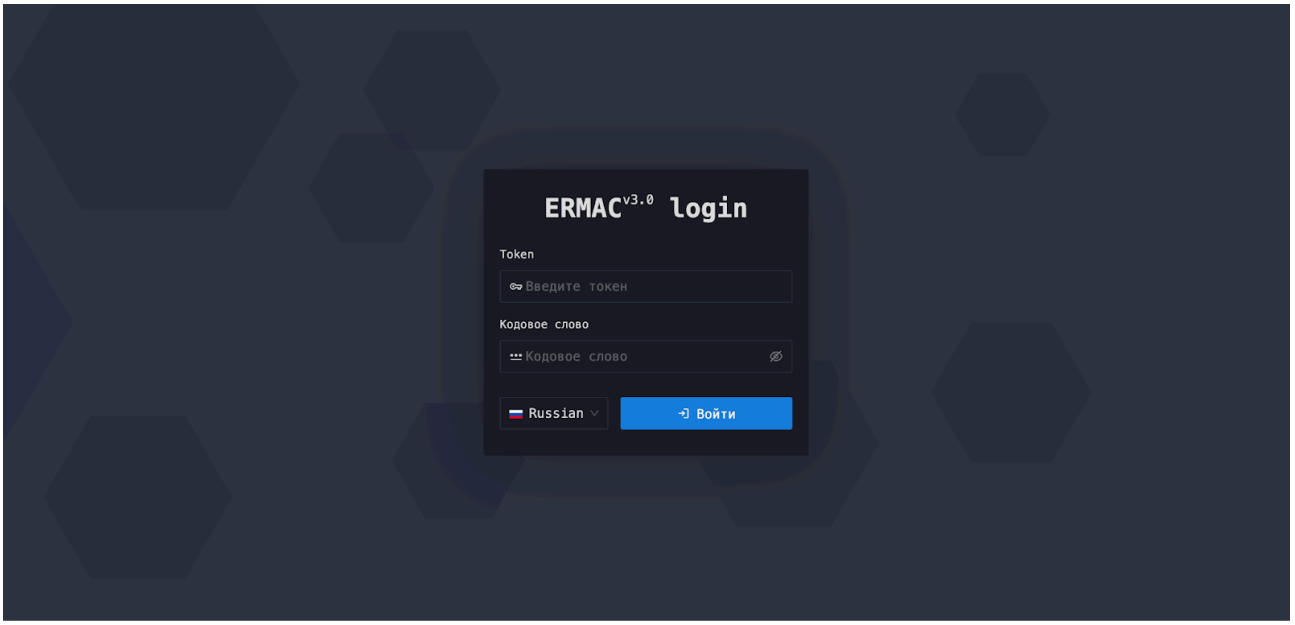


Fig 06: ERMAC V3.0 Panel Login

All ERMAC panels notably have the page title "ERMAC 3.0 PANEL" allowing us to find **43[.]160[.]253[.]145** and **91[.]92[.]46[.]12**. Upon simplifying this query, we can find an additional IP belonging to an older ERMAC panel: **206[.]123[.]128[.]81**

```
  SELECT * from httpv2
WHERE html.head.title = 'ERMAC 3.0 PANEL'
```

Copy

Output:

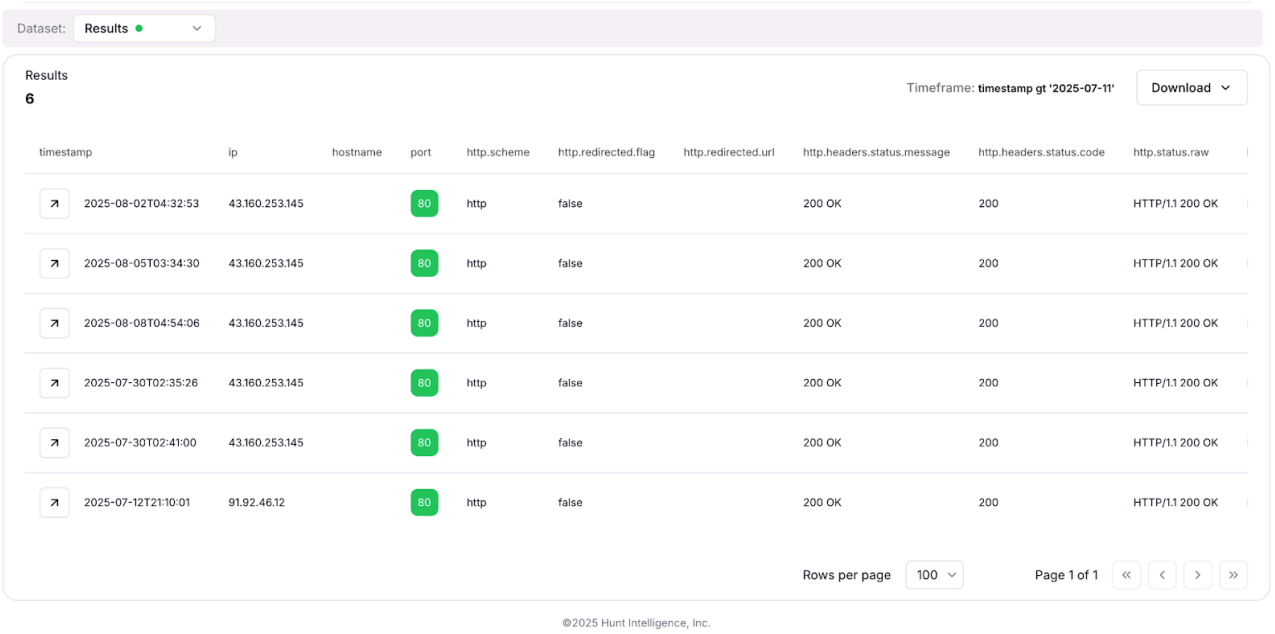


Fig 07: Results from searching for hosts with the ermac_session cookie, observed via HuntSQL by Hunt.io

From within the panel, operators are able to interact with connected devices by issuing commands or accessing stolen data.

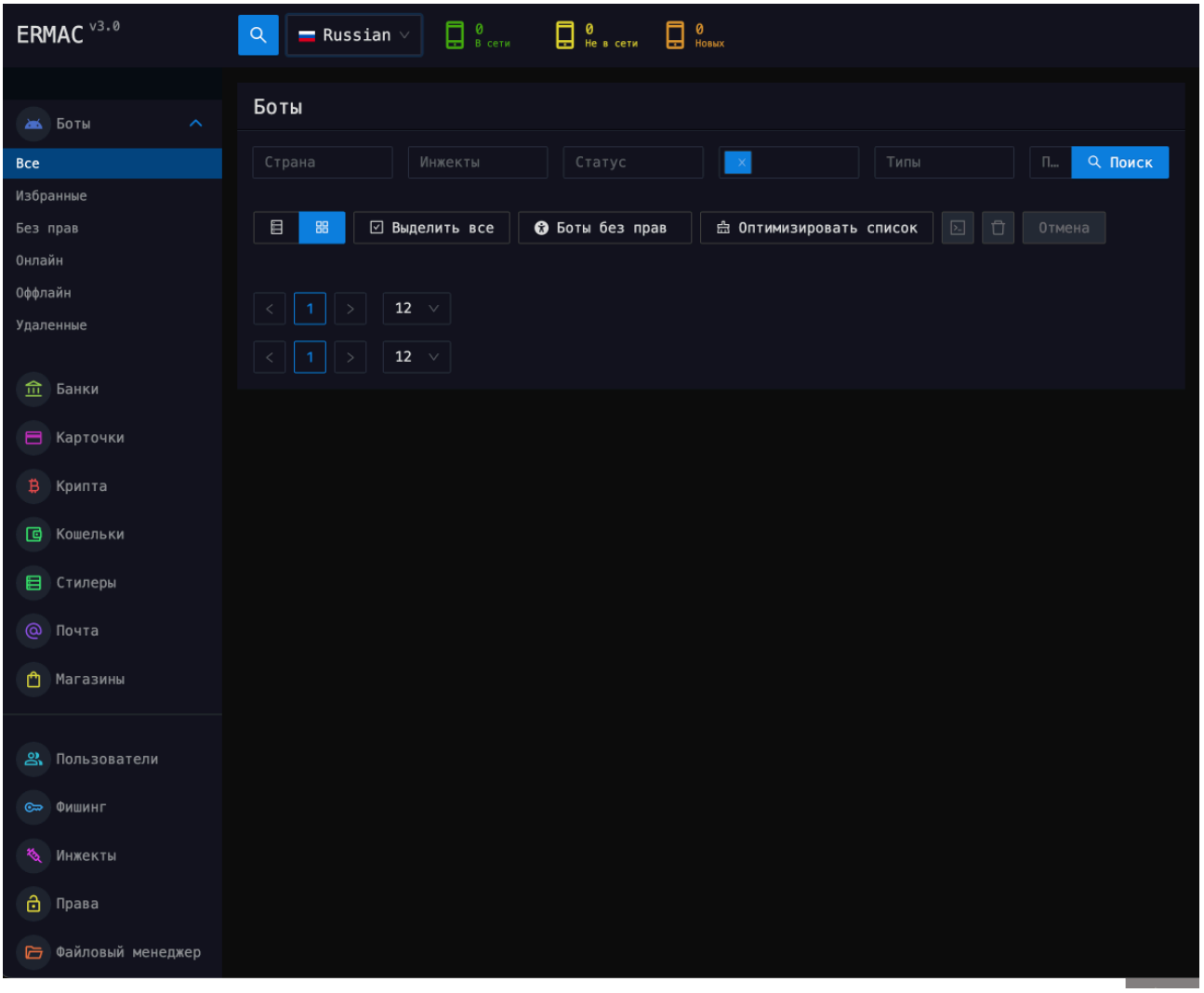


Fig 08: ERMAC Panel UI

This is accompanied by the ability to manage form injects directly through the panel, with the ability to enable or disable targeting of specific applications.

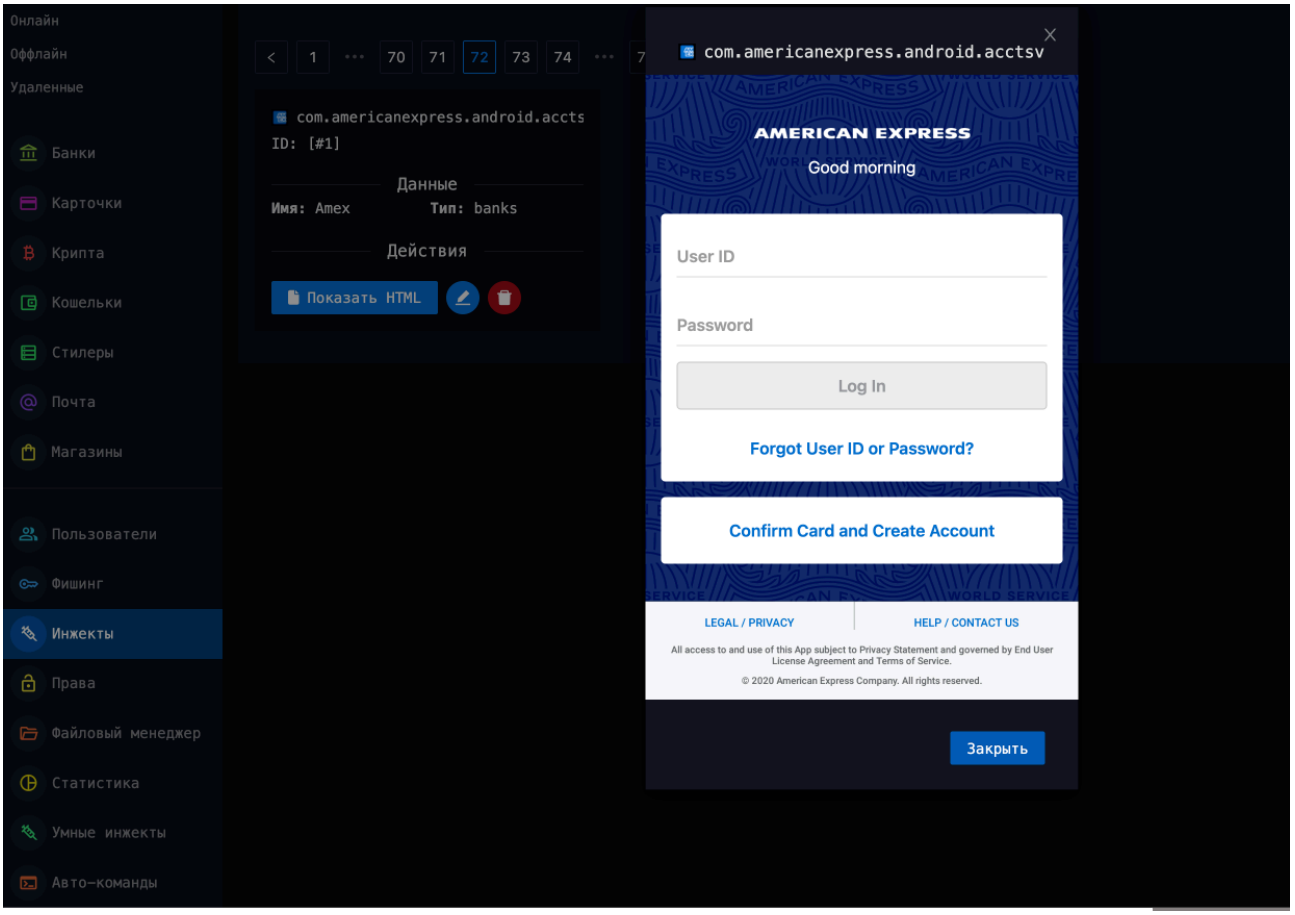


Fig 09: Form inject management system with adversaries able to upload and modify targeted applications.

Exfiltration Server

ERMAC is observed using a second server for exfiltrating stolen data. The exfiltration server is developed in Golang and provides an encrypted HTTP API for receiving exfiltrated data and managing information related to compromised devices.

Location	Name	Function Signature	Function Size
0066f820	github.com/user/server_go/src/aes.Decrypt	undefined gith...	742
0066fb20	github.com/user/server_go/src/aes.PKCS5Padding	undefined gith...	581
0066fd80	github.com/user/server_go/src/db.PutFile	undefined gith...	695
00670060	github.com/user/server_go/src/db.PutFile.func1	undefined gith...	76
006700c0	github.com/user/server_go/src/db.PathInfo	undefined gith...	611
00670340	github.com/user/server_go/src/db.PathInfo.func1	undefined gith...	76
006703a0	github.com/user/server_go/src/db.CheckIDBot	undefined gith...	1115
00670800	github.com/user/server_go/src/db.CheckIDBot.func1	undefined gith...	76
00670860	github.com/user/server_go/src/db.GetGlobalSettings	undefined gith...	1679
00670f00	github.com/user/server_go/src/db.GetGlobalSettings.func2	undefined gith...	76
00670f60	github.com/user/server_go/src/db.GetGlobalSettings.func1	undefined gith...	76
00670fc0	github.com/user/server_go/src/db.GetCommandBot	undefined gith...	1223
006714c0	github.com/user/server_go/src/db.GetCommandBot.func1	undefined gith...	76
00671520	github.com/user/server_go/src/db.UpdateInjection	undefined gith...	2480
00671ee0	github.com/user/server_go/src/db.UpdateInjection.func3	undefined gith...	76
00671f40	github.com/user/server_go/src/db.UpdateInjection.func2	undefined gith...	76
00671fa0	github.com/user/server_go/src/db.UpdateInjection.func1	undefined gith...	76
00672000	github.com/user/server_go/src/db.DownloadInjection	undefined gith...	1702
006726c0	github.com/user/server_go/src/db.DownloadInjection.func2	undefined gith...	76
00672720	github.com/user/server_go/src/db.DownloadInjection.func1	undefined gith...	76
00672780	github.com/user/server_go/src/db.InsertLog	undefined gith...	1080
00672be0	github.com/user/server_go/src/db.InsertLog.func1	undefined gith...	76
00672c40	github.com/user/server_go/src/db.AddUpdateBot	undefined gith...	3954
00673bc0	github.com/user/server_go/src/db.AddUpdateBot.func1	undefined gith...	76
00673c20	github.com/user/server_go/src/db.ExistBotToID	undefined gith...	415
00673de0	github.com/user/server_go/src/db.ExistBotToID.func1	undefined gith...	76
00673e40	github.com/user/server_go/src/db.ScanColumnNames	undefined gith...	452
00674020	github.com/user/server_go/src/db.GetValuesRow	undefined gith...	151

Fig 10: server_go exfiltration binary with the functions and libraries used.

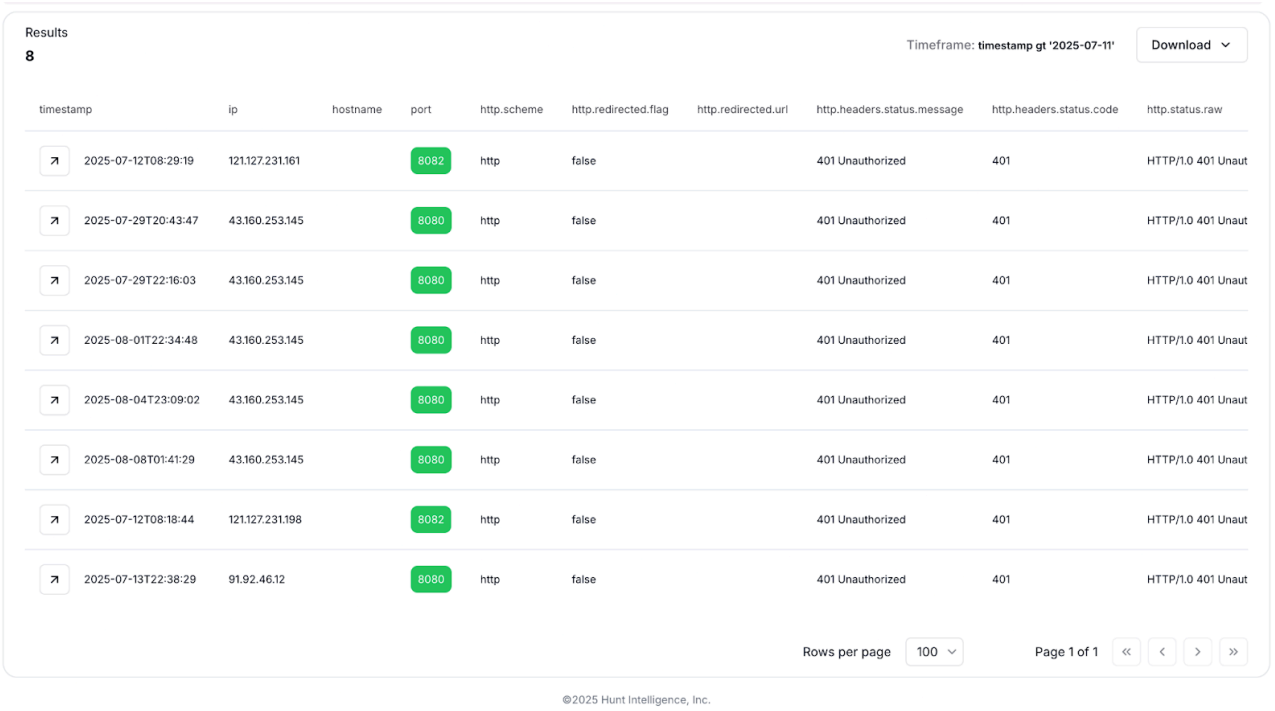
The usage of a separate server is likely to hide the main C2 panel, with operators able to use isolated servers for exfiltration.

ERMAC exfiltration servers are commonly seen using HTTP basic authentication with the following header `Www-Authenticate: Basic realm="LOGIN | ERMAC"`. By querying Hunt's SQL httpv2 dataset, we were able to identify four more unique exfiltration servers used by ERMAC.

```
  SELECT * FROM httpv2
WHERE http.headers.bytes.content like '%LOGIN | ERMAC%'
```

Copy

Output:



Results
8

Timeframe: timestamp gt '2025-07-11'

Download

timestamp	ip	hostname	port	http.scheme	http.redirected.flag	http.redirected.url	http.headers.status.message	http.headers.status.code	http.status.raw
2025-07-12T08:29:19	121.127.231.161		8082	http	false		401 Unauthorized	401	HTTP/1.0 401 Unaut
2025-07-29T20:43:47	43.160.253.145		8080	http	false		401 Unauthorized	401	HTTP/1.0 401 Unaut
2025-07-29T22:16:03	43.160.253.145		8080	http	false		401 Unauthorized	401	HTTP/1.0 401 Unaut
2025-08-01T22:34:48	43.160.253.145		8080	http	false		401 Unauthorized	401	HTTP/1.0 401 Unaut
2025-08-04T23:09:02	43.160.253.145		8080	http	false		401 Unauthorized	401	HTTP/1.0 401 Unaut
2025-08-08T01:41:29	43.160.253.145		8080	http	false		401 Unauthorized	401	HTTP/1.0 401 Unaut
2025-07-12T08:18:44	121.127.231.198		8082	http	false		401 Unauthorized	401	HTTP/1.0 401 Unaut
2025-07-13T22:38:29	91.92.46.12		8080	http	false		401 Unauthorized	401	HTTP/1.0 401 Unaut

Rows per page 100

Page 1 of 1

Fig 11: Public ERMAC Exfiltration Servers found with HuntSQL™

ERMAC Backdoor

ERMAC leverages an Android backdoor developed in Kotlin, which provides control of the compromised device. The Android application supports 71 different languages, with it providing operators the ability to configure encryption keys, C2 servers, and the application settings.

Fig 12: Sample of ERMAC's main configuration file.

ERMAC encrypts all traffic using AES-CBC PKCS5 padding with the hardcoded nonce "0123456789abcdef". All traffic from the C2 server is decrypted using the same server key and nonce.

Before executing ERMAC checks the phone carrier to ensure that it's not executing in a country within the Commonwealth of Independent States (CIS), a common method by malware developers within that region to avoid persecution. This is further followed by a check to determine if ERMAC is being run in an Emulator. If ERMAC detects either of these to be true, it will uninstall itself and quit.

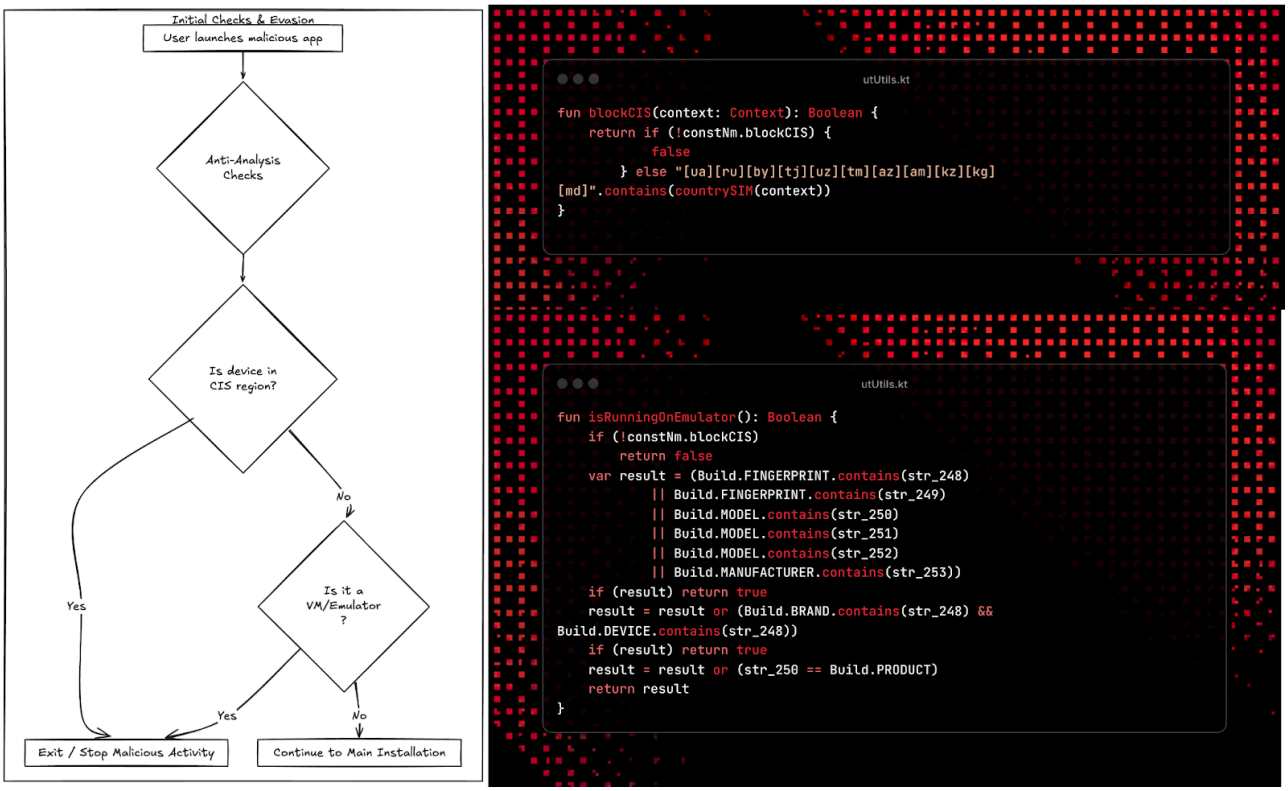
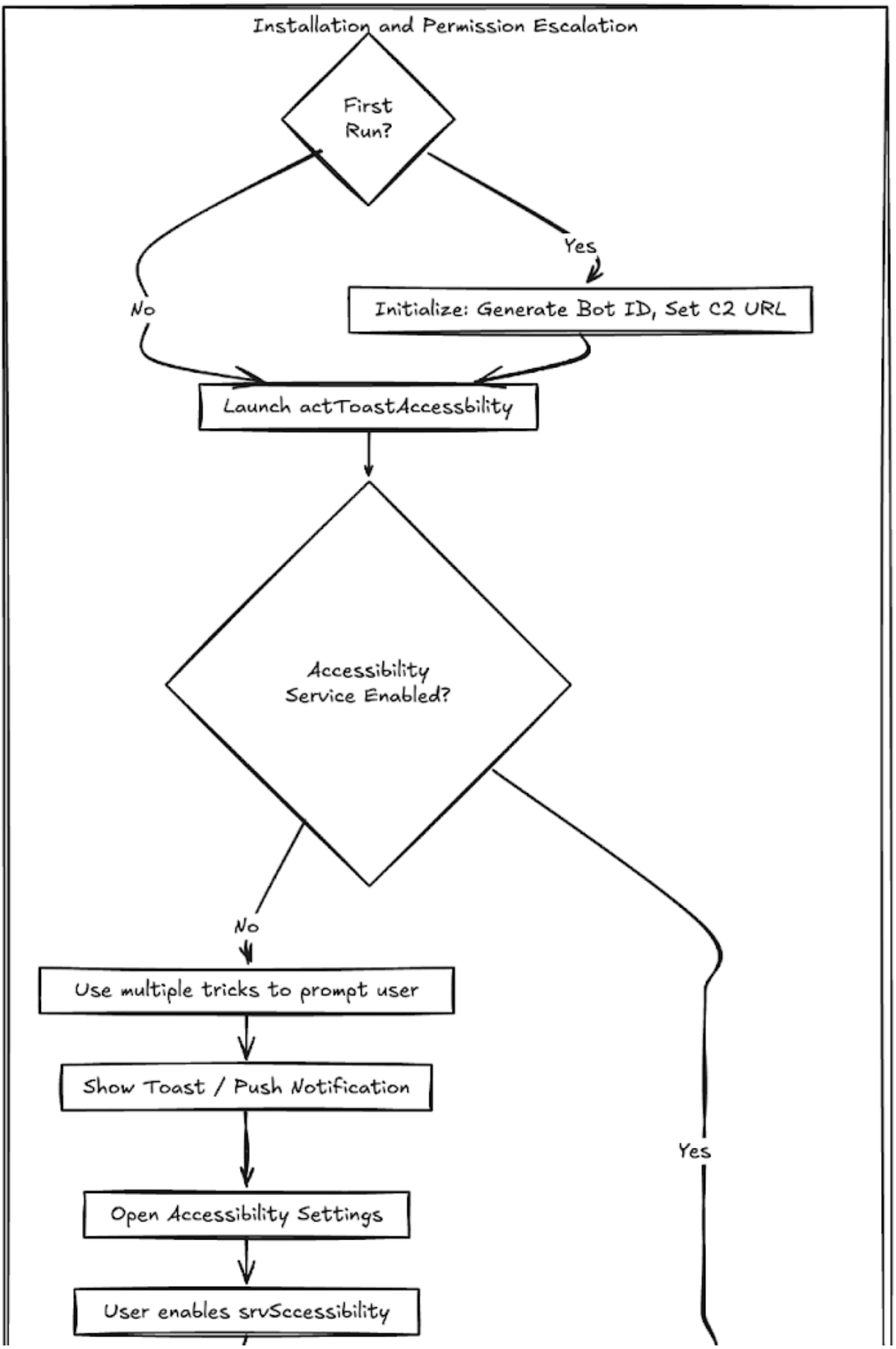


Fig 13: ERMAC Initialization

If the environment is safe, ERMAC will request elevated permissions so that it can run in the background, access SMS logs, and terminate programs. After ERMAC has installed itself properly, it will send all device information to the exfiltration server and listen for incoming commands.


Installation and Permission Escalation
First Run?
Yes
No
Initialize: Generate Bot ID, Set C2 URL
Launch actToastAccessbility
Accessibility Service Enabled?
No
Use multiple tricks to prompt user
Show Toast / Push Notification
Open Accessibility Settings
User enables srvSccessibility
Yes

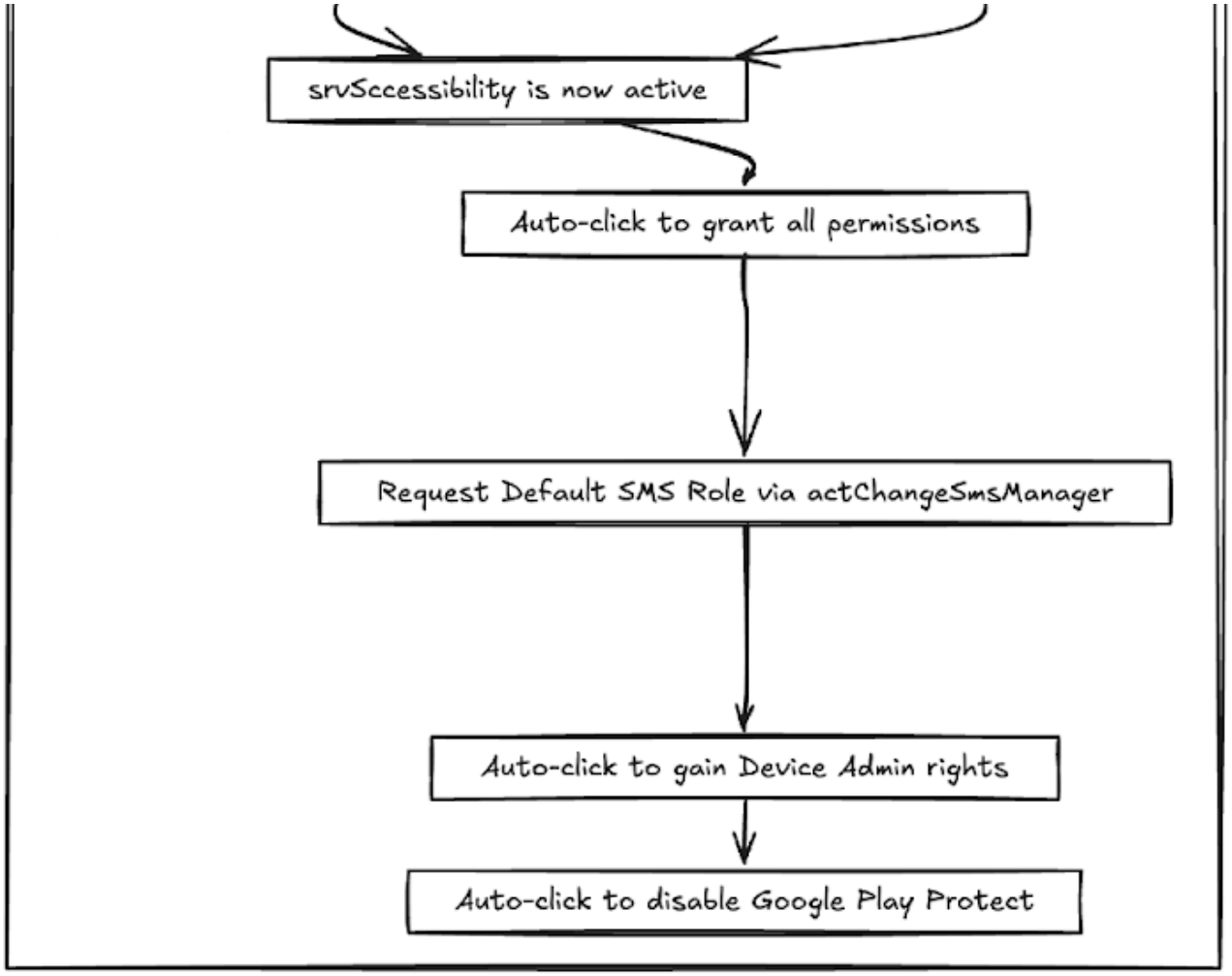


Fig 14: ERMAC Device Provisioning Flow

ERMAC supports the following commands:

Command	**Description**
sendsms	Sends an SMS message to a specified phone number. Can specify the SIM card.
startussd	Executes a USSD request (e.g., *100#). Can specify the SIM card.
forwardcall	Sets up call forwarding to a specified number. Can specify the SIM card.
push	Displays a custom push notification with a specified title, text, and icon.
getcontacts	Steals the user's contact list and sends it to the C2 server.
getaccounts / logaccounts	Steals a list of registered accounts (e.g., Google) and sends it to the C2.
getinstallapps	Gets a list of all installed applications and sends it to the C2.
getsms	Steals all SMS messages from the device and sends them to the C2.

Command	Description
startinject	Launches a fake overlay (injection) for a target application to steal credentials.
openurl	Opens a specified URL in the device's default web browser.
sendsmsall	Sends a specified SMS message to every contact in the user's address book.
startapp	Launches a specified application.
clearcache / clearcash	Attempts to clear the cache of a specified application.
calling	Initiates a phone call to a specified number.
deleteapplication	Uninstalls a specified application from the device.
startadmin	Prompts the user to grant Device Administrator privileges to the malware.
killme	Removes the malware itself from the device.
updateinjectandlistapps	Forces an update of the injection list and the list of installed applications.
gmailtitles	Retrieves the subjects of emails from the Gmail app.
getgmailmessage	Retrieves the content of a specific email from the Gmail app.
fmmanager	With extra: "ls": Lists files/directories. With extra: "dl": Downloads a file to the C2.
takephoto	Takes a picture using the front camera and sends it to the C2 server.

ERMAC Builder

The final component of the ERMAC banking trojan is its web builder, which provides adversaries the ability to configure and create builds for its malware campaigns. The Web Panel allows adversaries to configure the application name, server URL, and a number of other configuration settings for the Android backdoor.

Fig 15: ERMAC Builder Panel UI

On each build, ERMAC uses [obfuscapk] to obfuscate the Android application, giving it a unique hash and obfuscating class names, strings, manifest properties, and more.

Infrastructure Linkages

With each component mapped, we pivoted to identifying how these pieces connect in live environments using HuntSQL™ queries

ERMAC uses a number of identifiable signals for its infrastructure. This equips threat hunters with the ability to quickly find core infrastructure belonging to ERMAC with [Hunt's SQL] search functionality.

ERMAC Panel

HuntSQL™ query for finding ERMAC 3.0 C2 Panels based on the HTML header title.

```
  SELECT * from httpv2
WHERE html.head.title = 'ERMAC 3.0 PANEL'
```

Copy

Why it matters: Panels are the operator's main interface for issuing commands and managing stolen data. Identifying them helps track live infrastructure and block access.

ERMAC C2 API

HuntSQL™ query for finding ERMAC 3.0 C2 APIs based on the ermac_session cookie, which is set by default.

```
  SELECT *
FROM httpv2
WHERE http.headers.bytes.content LIKE '%ermac_session%'
```

Copy

Why it matters: Hosts setting ermac_session are ERMAC backend APIs that handle bot control and data retrieval. Correlating these with panels helps map and confirm active malicious infrastructure.

ERMAC Exfiltration Server

HuntSQL™ query for finding ERMAC exfiltration servers behind basic authentication

```
  SELECT * FROM httpv2
WHERE http.headers.bytes.content LIKE '%LOGIN | ERMAC%'
```

Copy

Why it matters: These servers receive and store stolen data from infected devices. Blocking egress to them can prevent data exfiltration during an active compromise.

ERMAC Builder

HuntSQL™ query for finding ERMAC 3.0 Builder panels based on the HTML header title.

```
  SELECT * from httpv2
WHERE html.head.title = 'ERMAC 3.0 BUILDER'
```

Copy

Why it matters: Builder panels are used to create and configure new ERMAC APKs. Discovering them provides early warning of upcoming campaigns and distribution activity.

The queries above allow defenders to locate and monitor live ERMAC infrastructure. The next step is applying this intelligence to defense and disruption efforts

Mitigation Strategies

ERMAC targets users of banking, shopping, and other financial applications primarily through web injects. It relies on Android's WebView API to place an overlay on top of legitimate apps, capturing credentials and payment information. Implementing secure Android permissions such as FLAG_SECURE and using code to detect or block overlays can reduce exposure to this technique.

Defenders can also focus on identifying and disrupting ERMAC infrastructure. Regularly scan for active C2 and exfiltration servers, and block Android applications that reference known ERMAC IPs or domains.

Additional measures include:

- Monitoring for unique ERMAC traits such as HTTP headers, hardcoded nonces, and panel titles
- Strengthening mobile apps against overlay attacks through runtime detection and code obfuscation
- Running proactive threat hunts with Hunt.io's queries to uncover related infrastructure before it is used in campaigns

Where possible, share indicators with trusted threat intel networks and coordinate with hosting providers to take down exposed panels and exfiltration servers.

ERMAC Android Binary Yara Rule

As part of these mitigation efforts, the following YARA rule can be deployed to detect ERMAC Android binaries.

ERMAC consistently uses the package com.amazon.zzz for its builds. The following YARA rule detects Android APKs containing this string, which serves as a unique identifier for ERMAC. It can be used in malware repositories, sandbox systems, and endpoint scanning solutions to flag potential ERMAC infections.

```
rule Ermac_v3 {

meta:

  author      = "@huntio"

  date        = "2025-08-09"

  description = "ERMAC Android Backdoor"

  version     = "1.0"
```

```
  strings:
    $str0  = "com.amazon.zzz" fullword
  condition:
    uint32be(0) == 0x504B0304
      and filesize < 1MB
      and $str0
}
```

Copy

Conclusion

Hunt.io discovered and obtained the full ERMAC 3.0 source code, giving our team a rare opportunity to examine an active and evolving banking trojan from the inside. The combination of a Laravel-based C2 backend, React control panel, Golang exfiltration service, and obfuscated Android backdoor shows the level of sophistication behind its development. Through this direct access, we identified its expanded targeting across more than 700 financial and cryptocurrency apps, new form injection capabilities, and encryption methods that strengthen its stealth.

Our research also revealed critical weaknesses, including hardcoded credentials, default JWT tokens, and open panel registration. By correlating these flaws with live ERMAC infrastructure, we provide defenders with concrete ways to track, detect, and disrupt active operations. This analysis not only strengthens threat hunting and attribution but also exposes the operational risks of the Malware-as-a-Service model.

MITRE Att&ck Tactics

The following MITRE ATT&CK mapping outlines the techniques ERMAC employs across its lifecycle

Tactic	Technique	Sub-Technique	Description
Execution	User Execution	N/A	ERMAC is executed through user interaction
Execution	Command and Scripting Interpreter	N/A	TERMAC uses WebView components to execute remote JavaScript from its C2 server,

Tactic	Technique	Sub-Technique	Description
			for data collection.
Persistence	Scheduled Task/Job	N/A	Uses StartReceiver to schedule callbacks to maintain persistence.
Persistence	Service	N/A	ERMAC uses ActivityManager to ensure service stays online.
Mobile	SMS Control	N/A	Uses Android intents to intercept and control SMS messages.
Privilege Escalation	Elevated Execution with Prompt	N/A	Requests Device Administrator privileges to lock the screen and prevent uninstallation.
Defense Evasion	Encrypted Channel	N/A	All communication with the C2 server is encrypted using AES, which prevents network-level detection of commands and exfiltrated data.
Defense Evasion	Masquerading	N/A	ERMAC deletes its launcher icon after the first run to hide its presence on the device.
Defense Evasion	Obfuscated Files or Information	N/A	Strings and variable names are obfuscated with Russian text and Base64 encoding to hinder static analysis.
Defense Evasion	Emulator/Sandbox Evasion	N/A	Checks for emulation to prevent analysis.
Defense Evasion	Geofencing	N/A	Blocks [ua][ru][by][tj][uz][tm][az][am][kz][kg][md] from running the malware.
Defense Evasion	Disable or Modify System Firewall	N/A	Using Accessibility Services, ERMAC disables Google Play Protect to reduce the likelihood of being detected and removed.
Credential Access	Input Capture	Keylogging	Includes a keylogger functionality to capture user input across all applications.
Credential Access	Input Capture	GUI Input Capture	Utilizes form injection to capture form data from Android applications.
Credential Access	Access Notifications	N/A	Intercepts incoming SMS messages to steal sensitive information like two-factor authentication (2FA) codes.

Tactic	Technique	Sub-Technique	Description
Discovery	Account Discovery	N/A	Gathers a list of all user accounts on the device.
Discovery	Application Discovery	N/A	Exfiltrates a list of all installed apps.
Discovery	System Information Discovery	N/A	Collects detailed device information such as OS version, device model, phone number, and SIM details.
Collection	Protected User Data	Contacts List	The malware steals the user's entire contact list.
Collection	Protected User Data	SMS Messages	It reads and exfiltrates the user's existing SMS inbox.
Collection	Data from Local System	N/A	A built-in file manager module allows the attacker to browse the device's file system and download files.
Collection	Video Capture	N/A	It can remotely take photos using the front camera without the user's knowledge.
Command and Control	Application Layer Protocol	N/A	Communicates with its C2 server over HTTP/S.
Command and Control	Non-Standard Port	N/A	C2 communication may be obscured by using non-standard ports for exfiltration.
Command and Control	Fallback Channels	N/A	The malware is configured with a list of backup C2 domains to ensure communication persistence if the primary server fails.
Exfiltration	Exfiltration Over C2 Channel	N/A	All collected data, including credentials, contacts, and files, is sent to the attacker using the primary C2 channel.

ERMAC 3.0 Indicators of Compromise

Network Observables

IP Address	ASN	Behavior	Last Seen
43[.]160[.]253[.]145:80	AS132203	ERMAC 3.0 Panel	2025-08-08

IP Address	ASN	Behavior	Last Seen
91[.]92[.]46[.]12:80	AS214196	ERMAC 3.0 Panel	2025-07-17
206[.]123[.]128[.]81:80	AS207184	ERMAC 1.0-2.0 Panel	N/A
43[.]160[.]253[.]145:8080	AS132203	ERMAC Exfiltration Server	2025-08-08
121[.]127[.]231[.]163:8082	AS152194	ERMAC Exfiltration Server	2025-07-11
121[.]127[.]231[.]198:8082	AS152194	ERMAC Exfiltration Server	2025-07-12
121[.]127[.]231[.]161:8082	AS152194	ERMAC Exfiltration Server	2025-07-12
43[.]160[.]253[.]145:8089	AS132203	ERMAC C2 Server	2025-08-08
172[.]191[.]69[.]182:8089	AS8075	ERMAC C2 Server	2025-07-13
98[.]71[.]173[.]119:8089	AS8075	ERMAC C2 Server	2025-07-25
20[.]162[.]226[.]228:8089	AS8075	ERMAC C2 Server	2025-07-25
141[.]164[.]62[.]236:80	AS20473	Open directory with the ERMAC source code.	2024-03-06
5[.]188[.]33[.]192:443	AS202422	Mentioned in the source code, potentially an outdated panel or C2 server.	N/A

Host-Based Observables

Filename	SHA-256 Hash	Behavior
Ermac 3.0.zip	175d4adc5fc0b0d8eb4b7d93b6f9694e4a3089e4ed4c59a2828d0667a9992aaa	ERMAC Source Code
server_go	8c81cebbaff9c9cdad69257f50af0f5208a0d5923659b4e0c3319333f9e8d545	ERMAC compiled

Filename	SHA-256 Hash	Behavior
		exfiltration server

The full list of ERMAC Form Injects is available in **CSV format**.

Source: https://hunt.io/blog/ermac-v3-banking-trojan-source-code-leak