

Forum:[Hack Forums](#)

Thread: [\[GUIDE\] Choosing Between RATs, Stealers, Loaders and Botnets](#)

Orvex - July 28, 2025, 6:33 am

[GUIDE] Choosing Between RATs, Stealers, Loaders and Botnets (2025)

Introduction

This thread defines and compares four categories of software used in offensive operations: Stealers, Remote Access Trojans, Loaders and Botnets. Each tool type is explained in terms of operational role, deployment model, infrastructure requirement, detection profile and suitability for different phases of an intrusion campaign.

1. Stealers

Function

Stealers are single stage payloads designed to extract locally stored data from client devices. The target data includes browser credentials, session tokens, autofill forms, cold wallet contents and messaging platform sessions. These payloads do not maintain persistence. Once executed, they perform a short data harvest and immediately exfiltrate results to an external drop.

Use Case

Stealers are designed for people prioritizing credential theft at scale. They are used in phishing campaigns, Discord spam attacks, smishing, and file-based lure vectors. They are best suited for scenarios where you do not intend to retain access after initial compromise.

Exfiltration Method

Collected data is commonly sent to a webhook endpoint, API handler, or bot channel using encrypted or encoded formats. The transmission typically occurs over HTTPS and does not require persistent outbound connectivity.

Detection Profile

Stealers have a relatively small runtime footprint but are highly detectable by static scanners if not modified. Persistence mechanisms are not included, which reduces forensic risk but limits functionality. Rebuilding the binary regularly is necessary in scaled operations.

Advantages

- Fast execution
- Simple exfiltration
- Minimal infrastructure
- No persistent footprint

Limitations

- No ongoing access
- Easily detected if reused
- One-time execution window

- Limited ability to pivot

2. Remote Access Trojans (RATs)

Function

Remote Access Trojans provide full interactive control of a compromised machine. They are designed to maintain long-term access and support a wide range of capabilities including file system browsing, keylogging, screen capture, clipboard monitoring, microphone and camera access, privilege escalation, and lateral movement.

Use Case

RATs are used for manual exploitation of high-value targets, maintaining persistent access to compromised infrastructure, or establishing control points for further deployment of second stage payloads. They are typically used in targeted attacks rather than mass campaigns.

Command and Control Model

Most RATs require a persistent connection to a controller system using TCP, HTTP, WebSocket, or reverse proxy tunnel. Operators must configure backend listeners and ensure the availability of the C2 infrastructure.

Detection Profile

RATs are considered high risk from a detection standpoint due to their constant connection attempts, runtime behavior, and use of persistence mechanisms. Their executables are commonly analyzed for hardcoded strings, mutex values and behavioral signatures.

Advantages

- Long-term access
- Full device control
- Pivoting and lateral movement
- Can chain additional payloads

Limitations

- High detection rate without proper obfuscation
- Requires constant infrastructure uptime
- Vulnerable to traffic analysis and sandboxing
- Not suitable for mass targeting

3. Loaders

Function

Loaders are stub applications that deliver and execute other malware at runtime. They do not contain malicious functionality themselves. Their sole purpose is to retrieve and launch a second-stage payload such as a stealer, RAT, miner or ransomware module. Loaders can be designed to operate in-memory or on-disk and may or may not include persistence.

Use Case

Loaders are the most versatile malware class in use today. They are deployed at the initial infection stage to maintain modularity, reduce detection, and allow operators to swap payloads without changing the infection vector. They are ideal for campaigns requiring flexibility, frequent payload

rotation or segmented targeting.

Payload Handling

The loader fetches its payload from a remotely hosted source. This may be a private CDN, obfuscated domain, content proxy or secured object storage. The download and execution process may involve RunPE injection, dynamic DLL loading, PowerShell stagers or reflective memory loading.

Detection Profile

Well designed loaders have minimal detection risk since the stub contains no static indicators. If executed in memory with encrypted transport, the loader can remain fully undetected. Detection only occurs if payload URLs are known or if the loader is reused without modification.

Advantages

- Payload modularity
- High operational flexibility
- Extremely low detection surface
- Works with all malware types
- Allows campaign adjustment in real time

Limitations

- Requires external payload hosting
- Must handle URL rotation and uptime
- Stub must be rebuilt when exposed
- Requires attention to delivery flow

Operator Preference

Loaders should be considered the default delivery method in most campaigns. They provide separation between delivery and execution, which is essential for long term campaign viability. Payload rotation, modular deployment, and infection chain control are only achievable at scale when a Loader architecture is used.

Whether you are running phishing operations, traffic monetization, or credential collection at scale, Loader first strategies reduce detection, improve delivery rates and provide full control over your infection lifecycle.

4. Botnets

Function

Botnets consist of large networks of infected devices controlled from a centralized command platform. Each device operates as a node that can execute tasks such as distributed DDos attacks, spam relay, click fraud, proxy tunneling, or malware propagation.

Use Case

Botnets are suited for operators who need scale over precision. They are most commonly used for monetizing installs, renting traffic, executing volumetric attacks, or maintaining a fleet of machines to deliver additional payloads. They require backend development, panel protection and traffic management expertise.

Infrastructure Requirements

Botnets require a stable C2 platform with a resilient database, encrypted communication protocols,

bot status tracking and task distribution systems. Load balancing and abuse mitigation must be built into the backend.

Detection Profile

Botnet panels are prime targets for honeypots and sinkholes. Publicly reused frameworks are fingerprinted quickly. Each infected node increases exposure, making compartmentalization and infrastructure segmentation critical.

Advantages

- Scalable control
- Payload delivery at volume
- Monetization through installs or abuse services
- Modular plugin support for different functions

Limitations

- Requires backend protection
- Highly monitored by threat intelligence firms
- Legal and technical risks scale with bot count
- Unsuitable for precise or surgical targeting

Summary Table

Goal	Recommended Category	Rationale
Fast credential and token theft	Stealer	High-volume data collection, no persistence needed
Full interactive control of a victim device	RAT	Live access, manual file retrieval, surveillance
Dynamic payload deployment with modular control and low detection	Loader	Flexible architecture with payload rotation
Long-term monetization of thousands of installs	Botnet	Bulk control, payload chaining, traffic resale potential

Closing Remarks

Each malware type serves a precise tactical role. Stealers provide quick wins. RATs offer manual precision. Botnets deliver scale. But only Loaders provide the structural flexibility to adapt in real time.

If you have infrastructure in place, start with a Loader and build upward from there. All other malware classes should be treated as modular payloads attached to a Loader centric infection chain.