Framework Overview | Veil Framework

Archived: 2026-04-06 00:05:10 UTC

Framework Overview

The Veil Framework is a collection of tools designed for generating payloads, enumerating networks, and testing defensive controls in authorized security assessments. What distinguishes it from a simple script collection is the modular architecture: each component addresses a discrete phase of a penetration test, from initial payload creation through post-exploitation analysis.

This page covers the framework's structure, the thinking behind its design, and how teams put it to practical use in controlled environments. You will find guidance on module selection, safe lab setup, and the common misunderstandings that trip up newcomers.

Architecture and Module Structure

At its core, the framework follows a plugin-based architecture. Each module — Evasion, PowerView, Catapult, Pyherion, Pillage — operates independently but shares a common configuration layer and output format. This means you can swap modules in and out depending on the engagement scope without rearchitecting your workflow.

The modular approach was a deliberate design choice. Early monolithic tools often coupled payload generation with delivery, making it difficult to isolate which phase was detected by endpoint controls. By separating concerns, each module can be tested and updated on its own cadence.

How Teams Use This in Labs

Security teams typically deploy the framework within isolated virtual networks — think VMware or VirtualBox segments with no route to production infrastructure. A common setup looks like this:

- **Attack host:** Kali or Parrot OS running the framework
- **Target range:** Windows Server, Windows 10/11 workstations, sometimes a domain controller
- **Monitoring stack:** Sysmon, Windows Event Forwarding, a SIEM collector, and ideally an EDR agent in audit mode

The value is not in the payloads themselves but in the telemetry they produce. When you fire a Veil-generated payload against a monitored target, you learn what your detection stack catches, what it misses, and where your logging has blind spots. That feedback loop is the entire point.

Purple teams — where offensive and defensive operators collaborate in real time — have found the framework particularly useful for structured exercises. The offensive side generates payloads with specific characteristics (encoded, obfuscated, staged), and the defensive side tunes rules and signatures against them.

Common Misunderstandings

"It bypasses all antivirus." This has never been accurate. Detection technology evolves continuously — what evades signature-based scanning today may be flagged by behavioral analysis tomorrow. The framework's evasion capabilities are useful for testing whether your specific controls detect specific techniques, not for making universally undetectable binaries.

"It's only for red teamers." Defensive teams arguably benefit more. If you operate a SOC, understanding how evasion payloads are structured helps you write better detection logic, tune your EDR policies, and communicate risk to stakeholders with concrete examples rather than abstract threat models.

"Running the tool equals a pentest." The framework is one component of a much larger methodology. A proper security assessment involves scoping, rules of engagement, systematic testing, evidence collection, and reporting. Simply generating a payload proves nothing on its own.

Ethical and Legal Boundaries

Every use of this framework must occur within environments where you have explicit written authorization. This is not a gray area — unauthorized access to computer systems is a criminal offense in virtually every jurisdiction. Testing against systems you do not own or do not have permission to test is illegal, regardless of your intent.

For educational purposes, set up your own lab. Cloud platforms like AWS, Azure, and GCP all offer free-tier resources sufficient for building a basic range. Alternatively, projects like DVWA, Metasploitable, and HackTheBox provide purpose-built targets for practice.

Framework Context

The concept of security testing frameworks has a long history in the industry. For background on how software frameworks in general are structured and why modular architecture matters in security tooling, Wikipedia's overview of software frameworks provides useful context on the design principles at work here.

Where to Go Next

If you are new to the framework, the Veil Tutorial walks through initial setup and your first payload generation in a safe lab. For module-specific documentation, the Modules hub links to each component's dedicated page. And if you want to understand the detection side, Veil-Evasion covers how to think about evasion from a defender's perspective.
