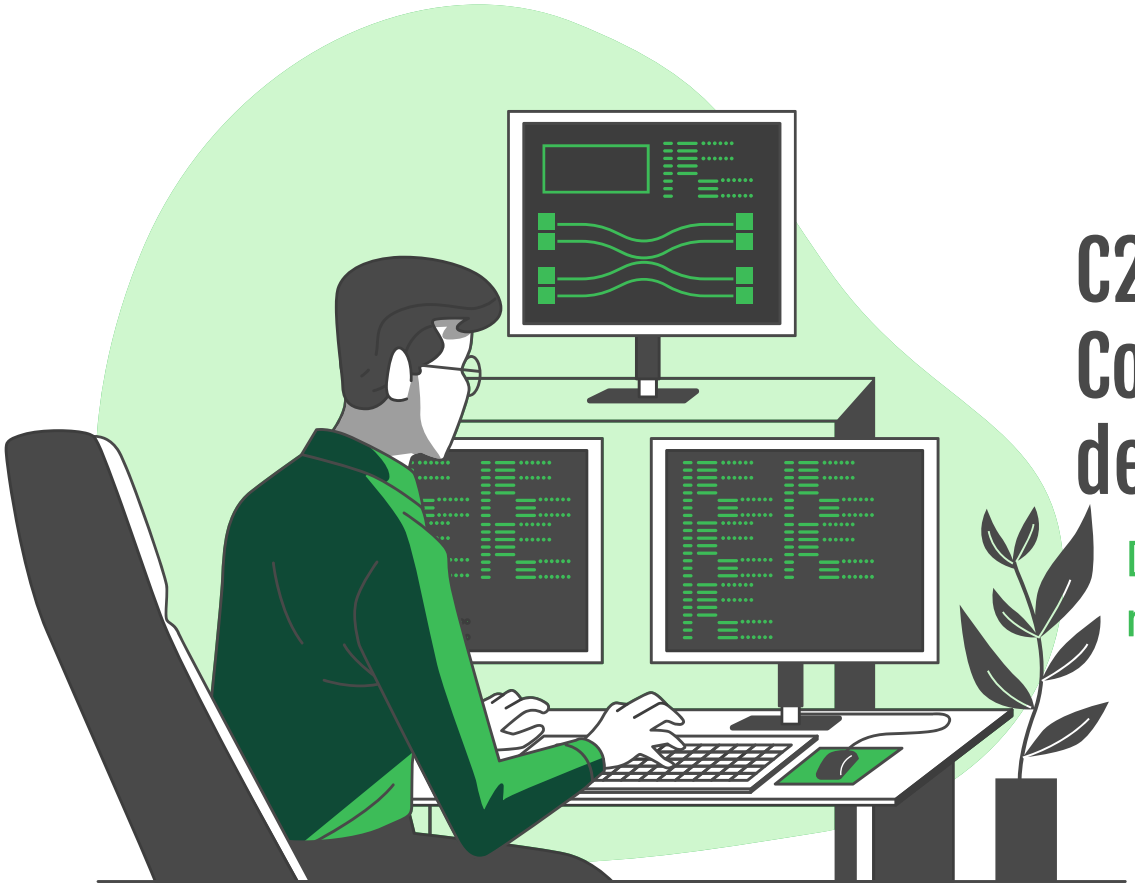


C2-Frameworks: Command & Control in der Netzwerksicherheit

Die verborgene Infrastruktur moderner Cyber Operationen

Sebastian Feustel

Diese Hacking-Tools sind ausschließlich für Bildungszwecke, Schulungen und Penetrationstests bestimmt. Hacking-Versuche auf Computern, die Ihnen nicht gehören (ohne Erlaubnis), sind illegal! Versuchen Sie nicht, Zugriff auf Geräte zu erlangen, die Ihnen nicht gehören.

Ausblick

01
Einleitung
Operation Phantom | Killchain

02
Theoretische Grundlagen & History
Ziel | Geschichte | Architektur

03
Tools & Live Demo
BOF | Empire | Havoc | Silver

04
Kommunikation & Tarnung
Verschleierung von Silver

01

Operation Phantom

Kims Schattenkanal

Überblick Operation Phanton

- Angriffs Start September 2024
- Verantwortlich Lazerus Gruppe APT38 (Nordkorea)
- Zielgruppe: Entwicklerteams im Bereich Kryptowährung und Technologie weltweit
- Nutzung komplexer Infrastruktur mit VPNs, russischer Proxy-Server

Auswirkungen Operation Phantom

Auswirkungen

1.500+

- Mehr als **1.500 Systeme weltweit** kompromittiert über drei Wellen (Nov 2024: 181 Entwickler; Dez 2024: zahlreiche Opfer in Indien/Brasilien; Jan 2025: weitere 233 Opfer)

- Abfluss sensibler Daten: **Entwicklungs-Credentials, Tokens, Browser-Passwörter, Systeminformationen.**

- Ziel im Hintergrund: **Finanzierung nordkoreanischer staatlicher Programme** über Kryptowährungsdiebstähle – Konsistenz mit früheren Aktivitäten.

Warum relevant?

- **Supply-Chain Angriff:** Entwicklungs-Tools kompromittiert und damit nicht nur Endnutzer betroffen

- **Hochkomplexes Setup**: moderner Admin-Panel, persistente RDP-Sessions, **spezialisierte C2-Server**

Quelle: https://securityscorecard.com/blog/operation-phantom-circuit-north-koreas-global-data-exfiltration-campaign

02

Killchain

Nach Lockheed Martin

Killchain

Phases of the Intrusion Kill Chain

Quelle: https://de.wikipedia.org/wiki/Cyber_Kill_Chain

03

Ziele

Aus Red Team Sicht

Ziele

1. Zentrale Kontrolle kompromittierter Systeme
2. Zuverlässige und verdeckte Kommunikation
3. Persistenz & Resilienz
4. Operationssicherheit (OPSEC)
5. Flexibles Tasking & Post-Exploitation
6. Situational Awareness (Entscheidung für weitere Angriffe Schritte)
7. Anpassung an Blue-Team-Maßnahmen

04

Geschichte

Von den 90s bis heute

Geschichte

ca. 1998–2003 | IRC-C2

- Botnets über öffentliche IRC-Server
- Klartext, leicht zu takedownen
- Beispiel: **GTBot, Sdbot**

2004–2010 | HTTP-basierte C2

- Wechsel von IRC zu HTTP
- Tarnung als Web-Traffic
- Beispiel: **Zeus, SpyEye**

2011–2017 | Verschlüsselung & Custom Protokolle

- HTTPS, eigene Binary-Protokolle
- Fokus auf Stealth & Persistenz
- Cobalt Strike entsteht
- **SkyWiper (Flame)** Lua-basierte Module
- Dynamisches Nachladen

Quelle: https://web.archive.org/web/20120528142705/http://www.crysys.hu/skywiper/skywiper.pdf
https://netlas.io/blog/evolution_of_c2_infrastructure/

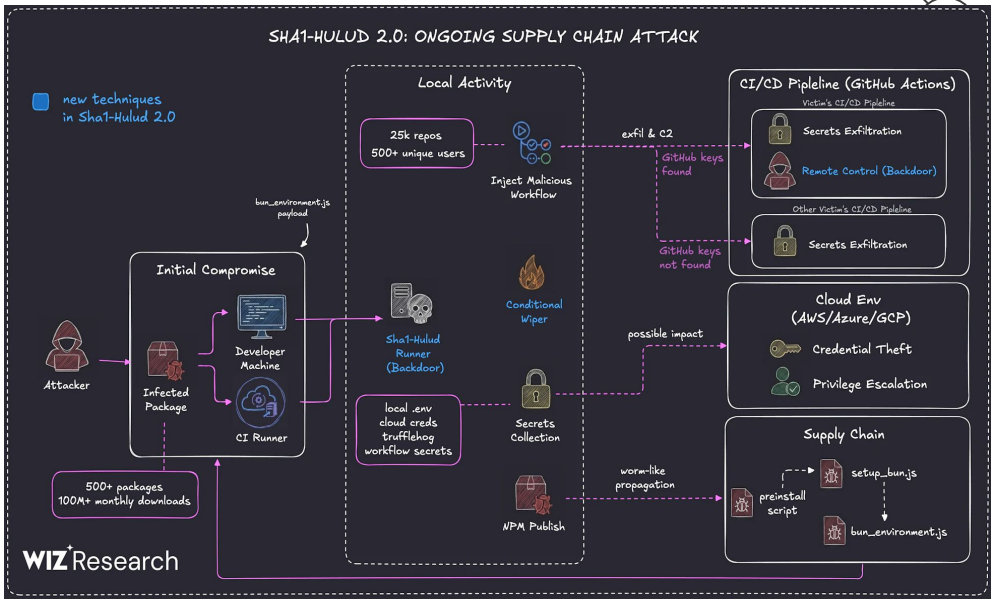
Geschichte

2018–2020 | BOF & In-Memory

- **Cobalt Strike 3.13** (Ende 2019) führte Beacon Object Files (BOF) offiziell ein
- Fileless Post-Exploitation

2021–heute | Cloud & Evasion

- Nutzung legitimer Cloud-Dienste
  a. Meshcentral (Awaken Likho)
  b. Github Action (Shai-Hulud)
  c. AnyDesk (Cozy Bear APT29)
- Kurzlebige Beacons, starke OPSEC



Quelle: https://www.wiz.io/blog/shai-hulud-2-0-ongoing-supply-chain-attack
https://securelist.com/awaken-likho-apt-new-implant-campaign/114101/

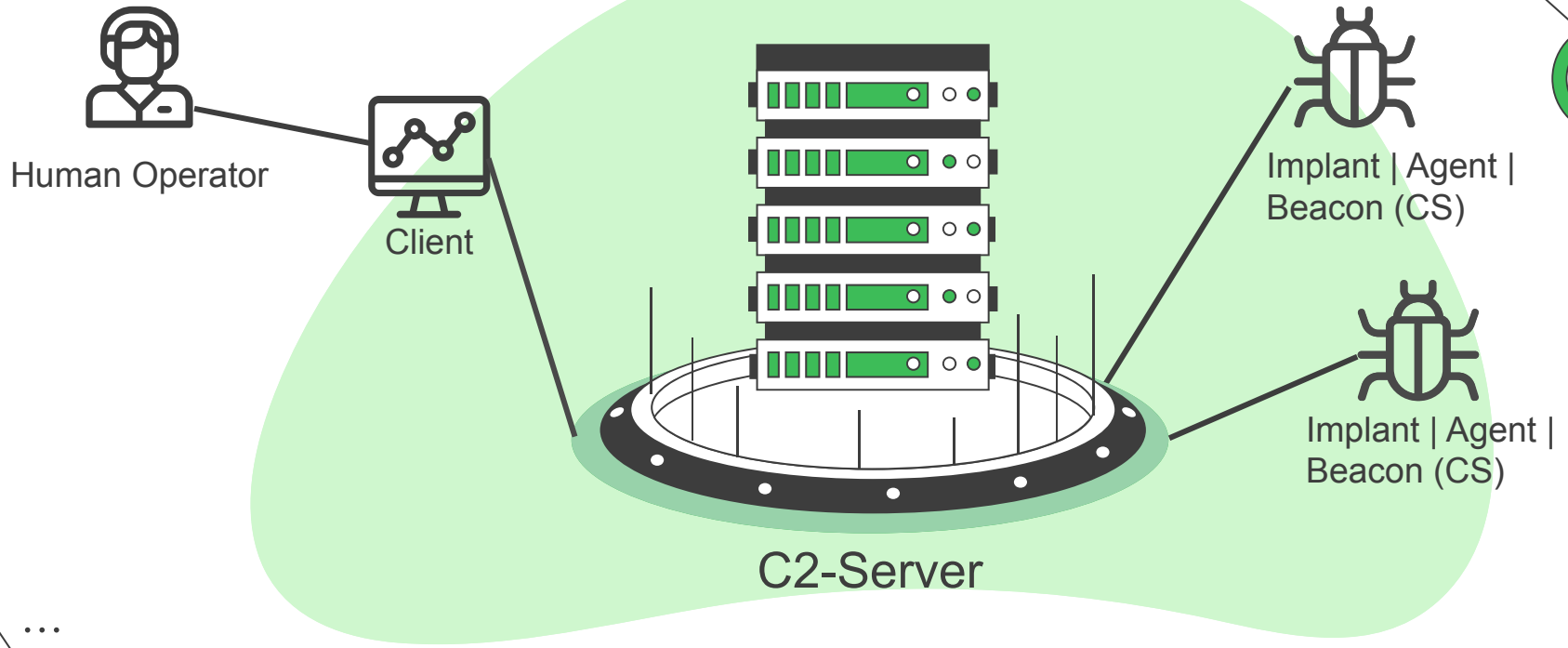
Youtube-Video zu Shai-Hulud

05

Architektur

Aufbau von C2 Frameworks

Architektur



06

BOF

Beacon Object Files

Beacon Object File

- Werden **zur Laufzeit in den Beacon-Prozess geladen** (In-Memory, kein Schreiben auf Disk)
- Nutzen das **COFF-Format** (kein vollständiges PE, kein Linker zur Laufzeit)
- Verwenden eine **eingeschränkte C-Runtime** (keine libc / WinAPI nur über Beacon-APIs)
- **Erweitern Implant-Funktionalität**, ohne neue DLLs/EXEs zu droppen
- Ideal für **Post-Exploitation Tasks** (z. B. Enum, Dump, Token, Lateral Movement)
- Können **opsec-freundlicher** sein als klassische Payloads

Quelle: https://de.wikipedia.org/wiki/Common_Object_File_Format

BOF Beispiele Aufgaben

🧭 Host- & System-Enumeration

- Laufende Prozesse auflisten
- Uptime ermitteln
- Lokale Systemeinstellungen (Sprache | Tastaturlayout | Zeitzone)
- Hostname & lokale Benutzer
- Betriebssystem-Version / Build

🌐 Netzwerk-Enumeration

- IP-Adressen ermitteln
- Routing-Tabelle lesen
- ARP-Cache / ARP-Listen auslesen
- DNS-Cache lesen
- Aktive Netzwerk-Interfaces
- Offene Ports (lokal, rudimentär)

🧱 Security- & Defense-Discovery

- Windows Firewall Rules lesen
- Antivirus / EDR-Erkennung
- AppLocker / Defender-Policies
- Antimalware Scan Interface Status (teilweise)

🗂️ Datei- & Ressourcen-Zugriff

- Verzeichnisse / Dateien auflisten
- Zugriff auf Shares prüfen
- Datei Metadaten lesen (Größe, Zeitstempel)

Quelle: https://github.com/trustedsec/CS-Situational-Awareness-BOF

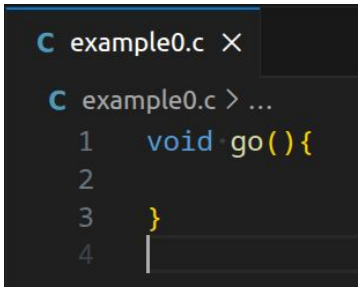
Youtube-Video zu BOFs

Minimal Entwicklung von BOFs

Compiler installieren:

```
Datei  Bearbeiten  Ansicht  Suchen  Terminal  Hilfe
sebastian@aspire5:BOF$ sudo apt install g++-mingw-w64-x86-64 gcc-mingw-w64-x86-6
```

Code:



Zu COFF kompilieren :

```
Datei  Bearbeiten  Ansicht  Suchen  Terminal  Hilfe
sebastian@aspire5:BOF$ x86_64-w64-mingw32-gcc -c -o example0.o example0.c
sebastian@aspire5:BOF$ file example0.o
example0.o: Intel amd64 COFF object file, no line number info, not stripped, 6 sections,
symbol offset=0x166, 16 symbols, 1st section name ".text"
sebastian@aspire5:BOF$
```

Hello World als BOF

- Beacon.h von Cobalt Strike herunterladen
- Nutzen von Funktionen z.B. BeaconPrintf

example1.c 2

example1.c > ...

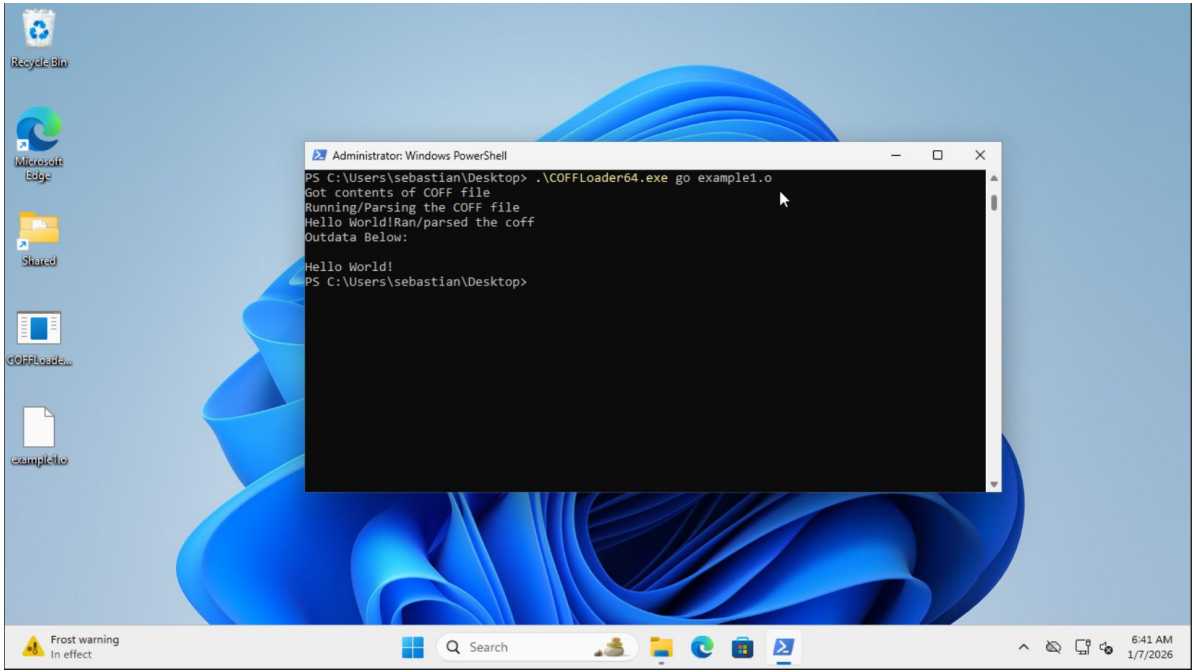
```c
#include "beacon.h"

void go(char * args, int alen) {
    BeaconPrintf(CALLBACK_OUTPUT, "Hello World!");
}
```

Quelle: https://hstechdocs.helpsystems.com/manuals/cobaltstrike/current/userguide/content/topics/beacon-object-files_how-to-develop.htm
https://github.com/Cobalt-Strike/bof_template

Hello World als BOF

- Nutzen eines Loaders der COFF Datei Beispiel: https://github.com/trustedsec/COFFLoader
- Implants der C2 Frameworks können diese dann laden



07

C2 Frameworks

Werkzeuge für die C2 Kommunikation

Cobalt Strike

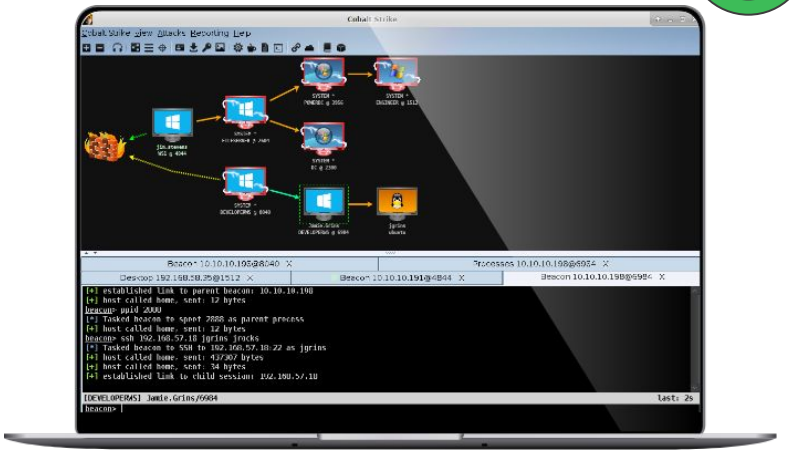
Kommerzielles Red-Team-Framework zur Simulation von Advanced Persistent Threats (APT)

Unterstützt klassische **C2-Kommunikation**:

- HTTP / HTTPS
- DNS
- SMB (Peer-to-Peer)

Post-Exploitation-Funktionen:

- Command Execution
- Credential Dumping
- Lateral Movement
- Privilege Escalation



Gilt heute als **„Goldstandard“ für C2-Frameworks** sowohl bei Red Teams als auch bei realen Threat Actors

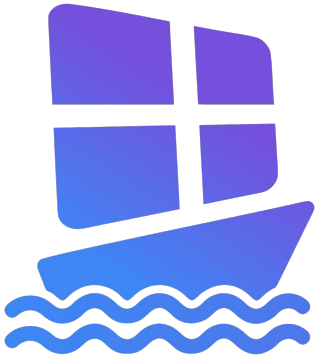
Quelle: https://www.fortra.com/de/product-lines/cobalt-strike

Quellcode offene Frameworks

Nr	Name	URL	Stars
1	Sliver	https://github.com/BishopFox/sliver	10440
2	Havoc	https://github.com/HavocFramework/Havoc	8039
3	Merlin	https://github.com/Ne0nd0g/merlin	5462
4	Empire	https://github.com/BC-SECURITY/Empire	4975
5	Mythic	https://github.com/its-a-feature/Mythic	4208
6	SILENTTRINITY	https://github.com/byt3bl33d3r/SILENTTRINITY	2304
7	PoshC2	https://github.com/nettitude/PoshC2	2075
8	SharpSploit	https://github.com/cobbr/SharpSploit	1853
9	trevorc2	https://github.com/trustedsec/trevorc2	1302
10	Loki	https://github.com/boku7/Loki	1267
11	Malice-network	https://github.com/chainreactors/malice-network	403
12	koadic	https://github.com/offsecginger/koadic	328
13	NamelessC2	https://github.com/trickster0/NamelessC2	284
14	Conquest	https://github.com/jakobfriedl/conquest	249
15	ThunderStorm	https://github.com/iDigitalFlame/ThunderStorm	40

Mein Setup

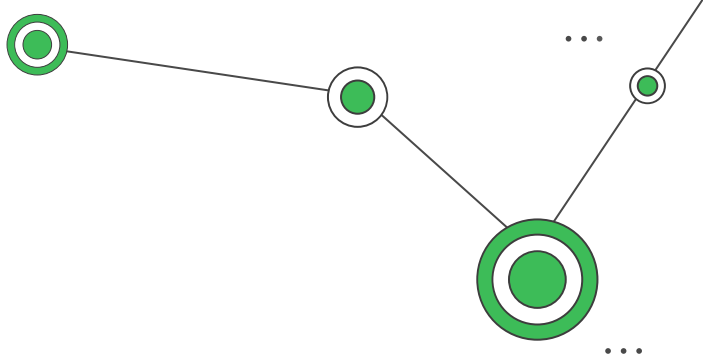
- vServer Hetzner als C2 Server: 46.224.114.83
- Ziel System: Windows im Winboat (Docker)
- Client mein Laptop



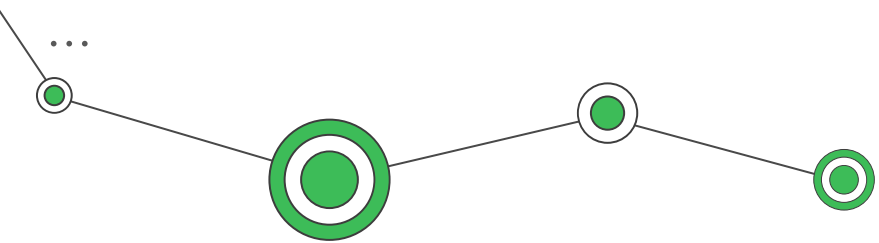
Empire

- Empire ein Framework in Powershell basiert
- Client heißt Starkiller eine WebUI entwickelt mit Vue
- C2-Protokolle (HTTP/S, SMB, TCP)
- Firma dahinter BC Security

Quellen: https://github.com/BC-SECURITY/Empire
https://github.com/BC-SECURITY/Starkiller
https://bc-security.gitbook.io/empire-wiki



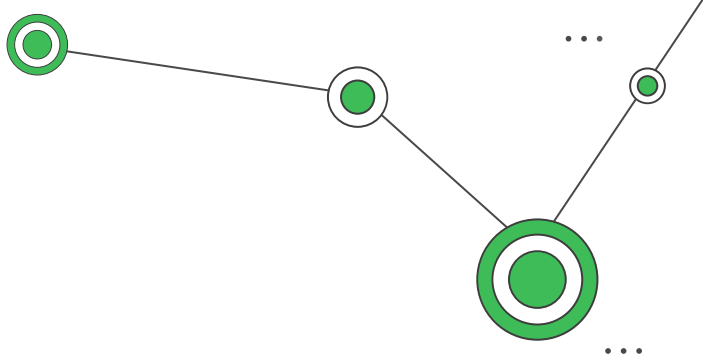
Demo-Video-1



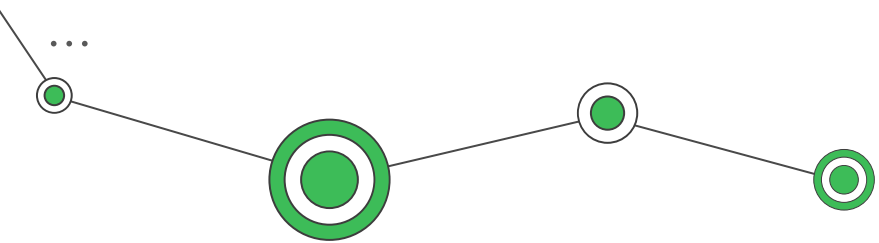
Havoc

- Server in **Golang**
- Demon-Agent als Implant
- GUI-Client (**C++, Qt**)
- HTTPS / WebSocket-basierte C2-Kommunikation
- Unterstützt **BOF-ähnliche Module** (COFF Loader)

Quellen: https://havocframework.com/docs/installation
https://github.com/HavocFramework/Havoc



Demo-Video-2

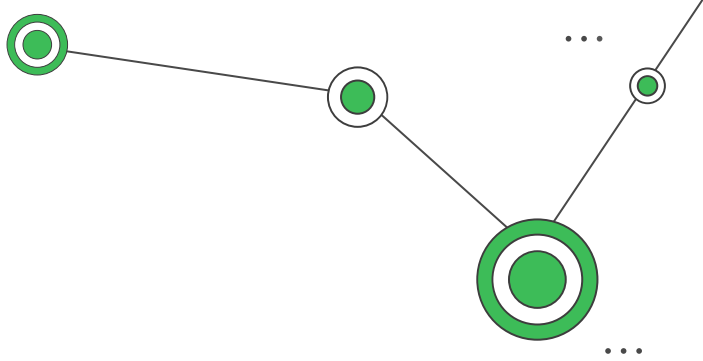


Youtube-Video zu Havoc

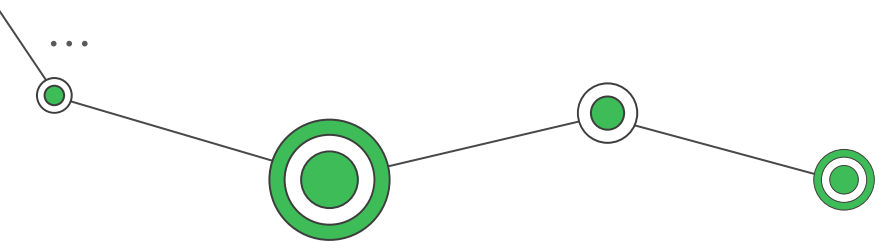
Sliver

- Entwickelt in **Golang** (Server und Implant)
- Unterstützung für **HTTP/S, mTLS, DNS, WireGuard**
- **BOF-Support**
- **CLI-first Workflow**, gut skriptbar
- **Multiplayer / Multi-Operator** fähig
- Sehr beliebt als **Cobalt-Strike-Alternative**
- Große **Community**

Quellen: https://github.com/BishopFox/sliver
https://sliver.sh/



Demo-Video-3



Youtube-Video zu Sliver

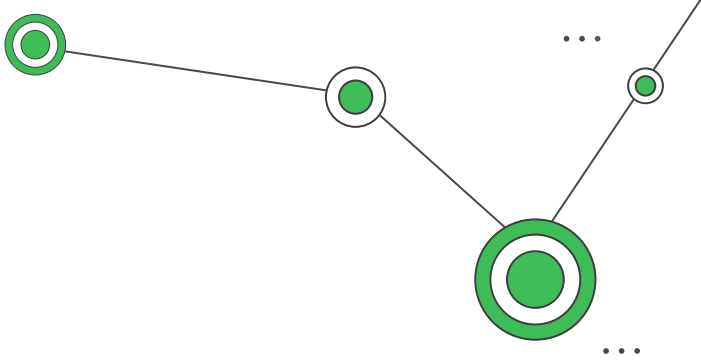
08

Kommunikation & Tarnung

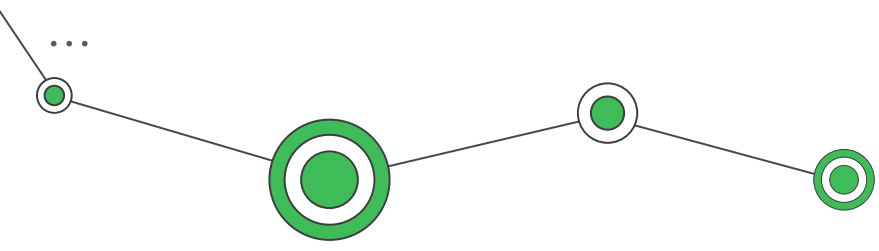
Die Verschleierung der Kommunikation

Sliver Kommunikation

- **Nutzung von Standard-HTTP-Methoden**
  GET für Beaconing, POST für Task-Ergebnisse
- **Realistische URL-Strukturen**
  z. B. /api/v1/status, /assets/js/app.js, /update/check
- **Legitime HTTP-Header**
  User-Agent, Accept, Accept-Language, Referer
- **Browser-ähnliches Verhalten**
  Gleiche Header-Reihenfolge, Case, Default-Werte
- **Antworten sehen aus wie normale Webinhalte**
  JSON, HTML, JavaScript, Bilder (Content-Type passend)
- **Variable Request-Größen**
  Kein gleichbleibendes Beacon-Pattern
- **Timing wie bei Web-Nutzung**
  Unregelmäßige Abstände, Tageszeit-bezogen



Demo-Video-4



Danke!

Habt ihr noch Fragen?

sebastian.feustel@protonmail.com
www.sebastian-feustel.de