

s/ashsec
Red Teaming Services

Vom Gegner lernen

David Wind
Geschäftsführer
slashsec Red Teaming GmbH
slashsec.at

Über slashsec und mich

- **David Wind**, Geschäftsführer der *slashsec Red Teaming GmbH*
- Seit über **zehn Jahren in der IT-Sicherheit**.
- Schwerpunkt auf realistischen Angriffssimulationen und **Red Teaming**.
- Weitere Schwerpunkte sind **Social Engineering**, physische Angriffssimulationen sowie Windows- und Webanwendungssicherheit.

s/ashsec
Red Teaming Services

Die Samples

- **8.200** unterschiedliche Dateien aus *MalwareBazaar*
- **27** Malware-Familien laut vereinheitlichten Feed-Labels
- **5.748** Dateien mit bestätigtem Start in der Sandbox

Analyse-Pipeline

- **Download → Statik → Sandbox → Auswertung**
- *MalwareBazaar* liefert passwortgeschützte ZIPs.
- Die **statische Analyse** untersucht Code, Strings und Dateistruktur.
- Die **Sandbox** versucht den Start und zeichnet die VM-Aktivität auf.
- *DeepSeek* fasst zusammen. Technische Aussagen prüfen wir an Artefakten.

Statische Analysen

- ***capa*** erkennt Code-Muster für Fähigkeiten wie Netzwerkzugriff oder Process Injection.
- ***FLOSS*** rekonstruiert versteckte Strings, etwa URLs und Dateipfade.
- ***YARA*** prüft Text- und Byte-Muster gegen Regeln.
- ***pefile*** liest PE-Header, Imports und Sections.
- Bei gepackten Dateien sehen die Werkzeuge zunächst vor allem **Loader oder Packer**.

Dynamische Analyse

- Jeder Run beginnt mit einem **frischen Klon** des Windows-10-Golden-Images.
- Der **Preflight** prüft die Isolation und stoppt bei Fehlern.
- **Fake-Services** beantworten DNS-Anfragen und Verbindungen im isolierten Netz.
- Der Hypervisor zeichnet Traffic auf. Ein **Host-Diff** erfasst neue Prozesse, Dateien und Tasks.

Analyse-VM | Setup

- Die VM läuft **ohne Guest Additions und Shared Folders**.
- Benutzername und Hardware-Merkmale sollen wie ein normaler Windows-Rechner wirken.
- **Browser-Verlauf und simulierte Interaktion** ergänzen das System.
- DNS und Verbindungen landen bei **Fake-Services**. Reale C2-Server sind nicht erreichbar.

Data, data, data

- **52 USD** für rund **250 Mio. Tokens**
  KI-Analysen mit *DeepSeek V4.1 Flash*
- **40,9 Mio. Pakete** und **1,61 GB TCP-Payload** in den Netzwerk-Summaries
- Zwölf Summaries erreichen 200.000 Pakete.
  Die vollständigen PCAPs gehen darüber hinaus.
- **328.717 Einträge** aus Statik, Runtime und KI-Beschreibungen

C2-Infra

s/ashsec
Red Teaming Services

C2-Infrastruktur | AS214351

- **24 Ziel-IPs** aus *AS214351* in rund **104 Sample-Treffern**.
- Die IPs verteilen sich auf **drei der vier /24-Netze** dieses AS.
- Mehrere Stealer, Loader und RATs verwenden Ziele in dieser Infrastruktur.

C2-Infrastruktur | Geteilte IPs

- ***196.251.107[.]130***
  16 *Stealc* (Infostealer) und **9** *Amadey* (Loader/Bot)
- ***62.60.226[.]159***
  18 *Stealc* (Infostealer) und **16** *Amadey* (Loader/Bot)
- Weitere Überschneidungen betreffen *Quasar* (RAT).

Konkrete Verhaltensbeispiele

s/ashsec
Red Teaming Services

Ant Colony und die DLL im Bild

- **Bitmap → RGB-Bytes → *OptiMax.dll***
- Die echte Ameisensimulation bringt die Bildressource mit.
- Der Decoder startet beim Fensteraufbau in *InitializeComponent*.
- **47.104 Bytes** ergeben eine gültige .NET-Assembly.
- Die Anwendung lädt sie **im eigenen Prozess**.

Nativer AsyncRAT-Client

- **Nativer Client → Bridge / CLR → .NET-Plugins**
- Der native x86-Client nutzt Plugins **im eigenen Prozess**.
- Die Bridge verbindet ihn mit der **.NET-Laufzeit CLR**.
- Über die Bridge lädt der Client .NET-Plugins und führt deren Funktionen im eigenen Prozess aus.
- Vergleichbar mit *SharpInline* (Brute Ratel) oder *InlineExecute-Assembly* (Cobalt Strike) - nur schlechter umgesetzt :D

Ein Launcher mit zwei EXEs

- Der Launcher schreibt **zwei eingebettete EXEs** ins Temp-Verzeichnis und startet sie.
- Ein Helper enthält Funktionen, die Windows Defender beeinträchtigen sollen (vermutlich mittels BYOVD)
- Der **HTTP-Bot** übernimmt die Kommunikation mit dem Server.

PowerShell-Sperre per IFEO

- Über **IFEO (Image File Execution Options)** kann Windows beim Programmstart einen Debugger vorschalten.
- Der Code setzt *svchost.exe* für *PowerShell* und *PowerShell ISE* als Debugger - somit rennt *PowerShell* “ins leere”
- Die beobachtete Persistenz läuft separat über einen **Run-Key**.

Unhooking, Patches und VEH

- **User-Mode-Unhooking** stellt NTDLL-Code im eigenen Prozess wieder her.
- **ETW- und AMSI-Patches** verändern lokale Melde- und Prüfpfade.
- Ein **Vectored Exception Handler** steuert Teile des Systemaufrufpfads.

Fixe IP und fremder TLS-Name

- Die Malware verbindet sich **direkt zur fixen IP aus ihrer Config**.
- Im **SNI-Feld des TLS-ClientHello** nennt sie einen legitimen Azure-Security-Dienst.
- Dieser Hostname ändert das Verbindungsziel nicht. Der Traffic geht weiterhin zur konfigurierten IP.
- Wer nur den TLS-Namen betrachtet, sieht den Namen des legitimen Dienstes.
- Ein öffentlicher *Amatera*-Write-up beschreibt eine sehr ähnliche Umsetzung.

Krypto-Adressen im Browser ersetzen

- Die DLL zielt auf den **Network-Service-Prozess** von Chromium-Browsern (Edge).
- Der Code durchsucht **POST-, PUT- und PATCH-Inhalte** nach Krypto-Adressen.
- Für passende Adresstypen sind **feste Ersatzadressen** hinterlegt.
- Im Run meldet die Komponente *Edge*-Kontext, *hooks=3* und *repl=0*.

BotKiller im Quasar-Fork

- Ein **Serverbefehl** kann die Suche nach anderer Malware starten.
- Der Scanner prüft Prozesse, Mutexe, Verbindungen und **Autostarts**.
- **76 Namenseinträge** und zusätzliche Heuristiken dienen als Suchmuster.
- Für Treffer enthält der Code Beenden- und Löschfunktionen. *Quasar* steht selbst auf der Liste.

Chromium-Testmaterial in Lumma

- Zehn zusätzliche Sections belegen **4,57 MB – rund 89 % der Datei**.
- Darin stehen Strings aus automatisierten **Chromium-Browsertests**, etwa *example.test*, *foo.com* und Dateipfade zu UI-Tests.
- Die String-Auswertung liefert dadurch **635 vermeintliche Netzwerkindikatoren** aus Testdaten.
- Die **neun Domains der Malware-Config** liegen separat vor und lassen sich aus Base64 und XOR rekonstruieren.

SOC-Geschenke

s/ashsec
Red Teaming Services

SOC-Geschenke

- **DNS:** *botnet-panel*, *credential-harvest*, *ddos-controller*, *ransom-note*, *stealer-gate*
- **Datei in AppData:** *njrat 0.7d horror edition.exe*
- **Run-Key:** *RemoteMinerAgent*

Noch mehr SOC-Geschenke

- **DNS:** *destinystealer[.]com*
  Fehlerbericht: *ZeroTrace.exe*
- **Datei im Temp-Verzeichnis:** *fsociety ddos v2.rar*
- **Run-Key und Task:** *XWormClient*
  Prozess und Verknüpfung: *xwormclient.exe*, *xwormclient.lnk*

Was wir als Red Team mitnehmen

s/ashsec
Red Teaming Services

Infrastruktur und Kommunikation

- **Blend in** heißt, die vorhandene Umgebung zu berücksichtigen, etwa die Nutzung von *AWS* oder *Azure*.
- Cloud-Dienste wie *CloudFront* oder *Azure* machen Sinn
- **C2-Traffic verschlüsseln.** Zusätzliche harmlose Daten um Entropie klein zu halten.
- Eine extern geladene Config, etwa über *GitHub* oder *Discord*

Loader und Ausführung

- Der **Loader ist die Grundlage**. Seine Qualität entscheidet, ob die Payload zuverlässig startet.
- **Packing und Anti-Debugging** erschweren die Analyse, garantieren aber keine unauffällige Ausführung.
- **ETW-, AMSI-Patches und Unhooking**
- **In-Memory-Ausführung**. Unnötige Drops und Process Injection vermeiden.
- **OPSEC** gehört in jede Stufe, vom Loader bis zur Kommunikation.

Was wir dem SOC nicht schenken wollen

s/ashsec
Red Teaming Services

Was wir uns sparen können

- **Feste Configs im Klartext** legen die Infrastruktur offen. Ein blockiertes, einziges C2-Ziel kann die Kommunikation lahmlegen.
- **Verräterische Subdomains** wie *c2.slashsec.at*, *botnet.slashsec.at* oder *teamserver.slashsec.at*
- Der **klassische Shellcode-Runner** reserviert RWX-Speicher mit *VirtualAlloc*, kopiert per *memcpy* und startet mit *CreateThread* -> **BIG RED FLAG**

Kontakt

Dipl.-Ing. David Wind, BSc
Geschäftsführer
slashsec Red Teaming GmbH
Kärntner Ring 5-7,
A-1010 Wien

@ dwind@slashsec.at
M +43 681/10 37 69 94
W www.slashsec.at



s/ashsec
Red Teaming Services