Ransomware: Analyse & Simulation

Bedrohungslage, Kill-Chain und technische Implementierung (Rust/Python)

Demo Präsentation

23. Dezember 2025

Teil I: Das Ökosystem Ransomware

Was ist Ransomware?

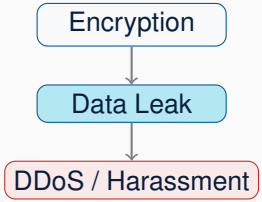
Definition: Malware, die den Zugriff auf Daten oder Systeme durch Verschlüsselung verhindert und für die Freigabe ein Lösegeld fordert.

Die Evolution der Erpressung:

Gen 1: **Single Extortion:** Nur Verschlüsselung.

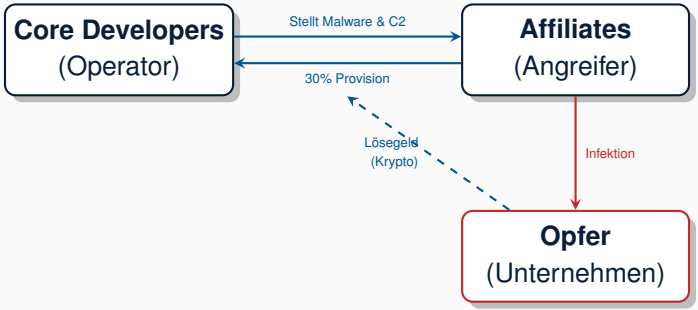
Gen 2: **Double Extortion:** + Diebstahl sensibler Daten (Drohung mit Leak-Seiten).

Gen 3: **Triple Extortion:** + DDoS-Attacken oder Belästigung von Kunden/Partnern des Opfers.



Das Geschäftsmodell: Ransomware-as-a-Service (RaaS)

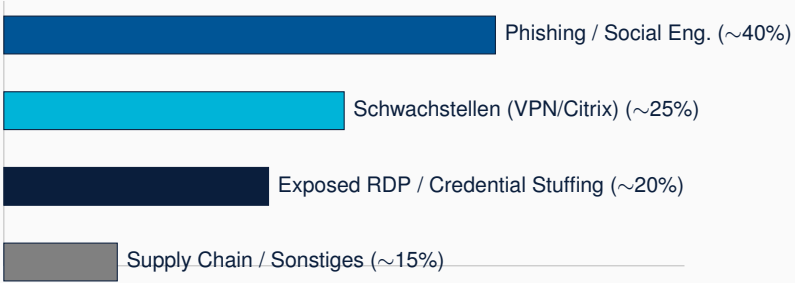
Ransomware ist heute eine hochprofessionelle Industrie mit Arbeitsteilung.



Vorteil für Kriminelle: Die Entwickler müssen sich nicht die Hände schmutzig machen (Einbruch), die Affiliates müssen nicht programmieren können.

Wie kommen sie rein? (Initial Access Vectors)

Die häufigsten Einfallstore für Ransomware-Affiliates:



Datenbasis: Aggregierte Reports 2023/2024 (Mandiant, Sophos)

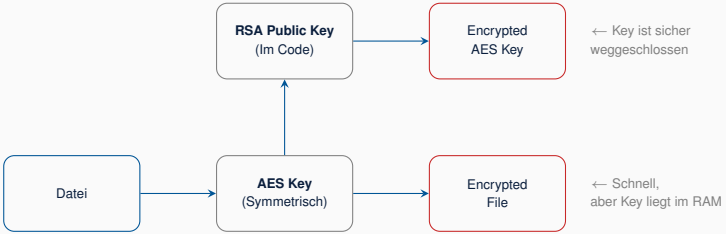
Die technische ”Kill Chain”

Vom ersten Zugriff bis zur Verschlüsselung vergehen oft Wochen (”Dwell Time”).

- **Schritt 2 & 3** sind entscheidend für die Verteidigung. Sobald Schritt 4 beginnt, ist es meist zu spät.
- Unsere Simulation überspringt **Schritt 3 (Recon & lateral Movement)**.

Exkurs: Warum wir den Key nicht einfach ”finden”

Professionelle Ransomware nutzt **hybride Verschlüsselung**, um Geschwindigkeit mit Sicherheit zu kombinieren.



Das Problem: Der AES-Key (zum Entschlüsseln nötig) wird mit dem Public Key des Angreifers verschlüsselt und aus dem Speicher gelöscht. Nur der Angreifer hat den Private Key zum Wiederherstellen.
Hinweis: In unserer Demo vereinfachen wir dies und speichern den Key lokal (‘rescue.key‘).

MITRE ATT&CK Taktiken

Initial Access	Persistence	Command & Control	Impact
• Phishing (T1566)	• Reg Run Keys	• Web Protocols	• Data Encrypted for Impact
• Exploit Public App	• Create Account	• Non-Standard Port	• Service Stop
• Valid Accounts	• Scheduled Task	• Encrypted Channel	

Mapping zur Demo: Unsere Rust-Malware nutzt *Registry Run Keys / Systemd* für Persistence und *Non-Standard Ports* (TCP Socket) für C2.

Command & Control (C2) Architektur

Die ”Lebensader” des Angriffs: C2 steuert Payloads, empfängt Keys und exfiltrierte Daten.

Nutzung unauffälliger Kanäle (Port 443, DNS), um in der Masse des Traffics unterzugehen.

Quellen: Halcyon – C2 in Ransomware; Packetlabs Deep Dive

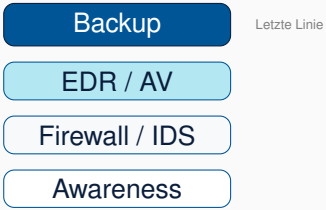
Erkennung & Abwehr (Defense in Depth)

Technische Maßnahmen:

- **C2-Detection:** Blockieren von bekannten Malicious IPs/Domains & Analyse von Beaconing-Mustern.
- **EDR (Endpoint Detection):** Überwachung auf Prozess-Injections und Massen-Dateiänderungen.

Organisatorisch:

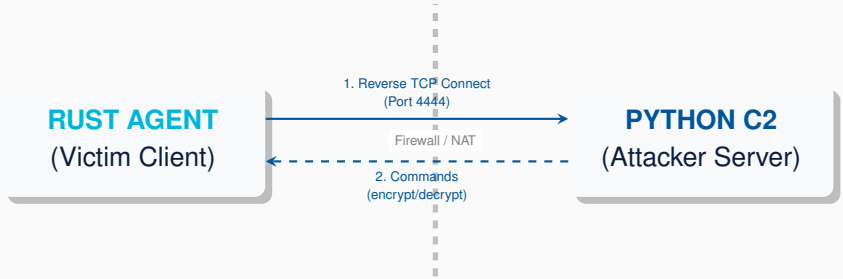
- Offline-Backups (Immutable).
- Netzwerksegmentierung.



Quellen: BSI; Darktrace – Early Signs of Ransomware

Teil II: Technische Simulation

Architektur der Simulation



- **Reverse Shell Prinzip:** Der Agent baut die Verbindung auf (Outbound), um Inbound-Firewall-Regeln zu umgehen.
- **Technologie:** Rust (Client) für Performance und Sicherheit, Python (Server) für einfache Handhabung.

Der Agent: Warum Rust?

Technische Vorteile

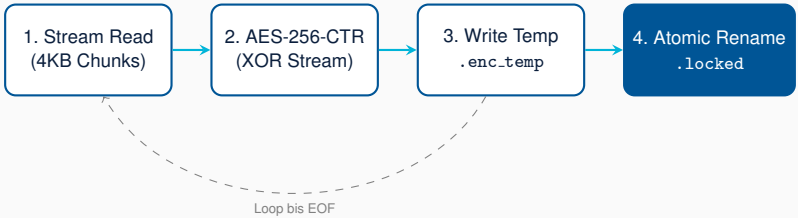
- **Memory Safety:** Keine Buffer Overflows (weniger Abstürze beim Opfer).
- **Zero Runtime:** Keine Installation von Python/Java nötig (Single Binary).
- **Cross-Platform:** Ein Code für Linux, Windows & macOS.

Evasion (Erkennung)

- Komplexer Maschinencode erschwert Reverse Engineering.
- Geringere Erkennungsrate bei klassischen AV-Signaturen im Vergleich zu C/C++.

Kryptographie: Der ”Atomic Lock”

Ein kritischer Fehler bei Malware ist Datenverlust durch Abstürze. Unser Agent nutzt ein atomares Verfahren:



AES-256-CTR: Der Counter-Mode ermöglicht schnelles Verschlüsseln ohne ”Padding” (Dateigröße bleibt identisch).

Persistenz: Das Überleben des Neustarts

Malware muss sich im System verankern. Der Rust-Code unterscheidet zur Kompilierzeit das OS:

Windows (Registry)

```
Command::new("reg")
    .args([
        "add",
        "HKCU\\...\\Run",
        "/v", "Updater",
        "/d", exe_path
    ])
```

Linux (Systemd)

```
[Unit]
Description=SystemSvc

[Service]
ExecStart=/path/to/agent
Restart=always
```

Ablauf der Demonstration

Wie die Komponenten im Test zusammenspielen:

1. **Setup:** Starten des Python C2 Servers (Listener auf Port 4444).
2. **Infektion:** Ausführen der `rust-mw` Binary auf dem Zielsystem.
3. **Handshake:** Client meldet sich beim Server (”New Session”).
4. **Angriff:**
   - Server sendet Befehl: `encrypt`
   - Client generiert Key $\rightarrow$ Verschlüsselt Dateien $\rightarrow$ Zeigt Lösegeldforderung.
5. **Lösung:** Server sendet Befehl `decrypt` (simulierte Zahlung).

Fazit

Bedrohung: Ransomware professionalisiert sich (RaaS, C2).

Technik: Rust + Sockets ermöglichen potente Malware.

Schutz: Defense-in-Depth ist alternativlos.