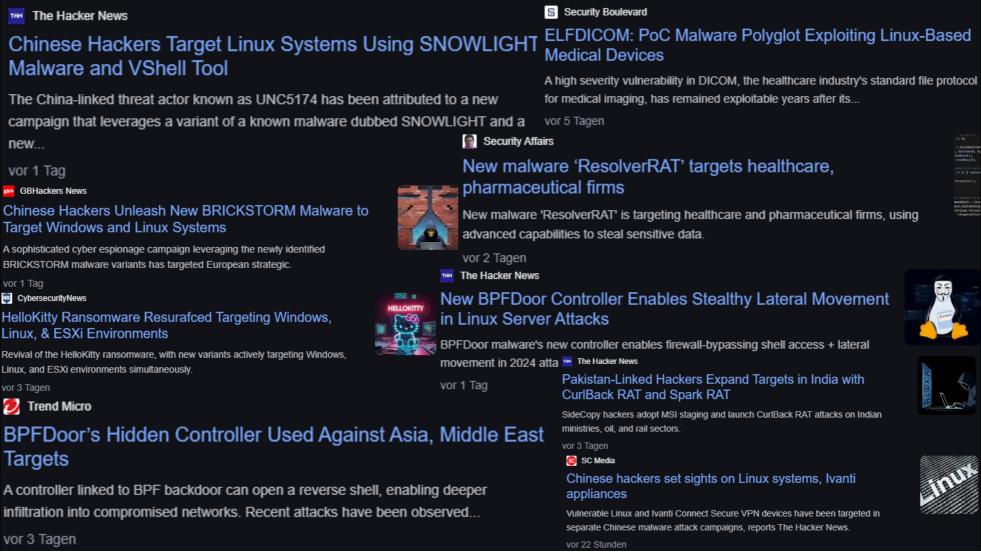


Analyse von Linux Malware

Advance Security Testing 25



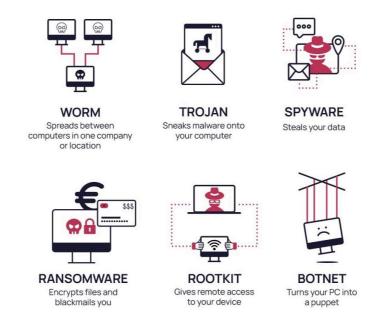
Agenda

- 1. Überblick
- 2. Statische Analyse
- 3. Dynamische Analyse
- 4. Malware Erkennung
- 5. Fazit

Überblick

Typen von Linux-Malware

Überblick



https://sosafe-awareness.com/glossary/malware/

Fallbeispiele: Prometei, Shikitega

Überblick

- Prometei
 - Modular aufgebautes Botnetz mit Fokus auf Kryptomining
 - Verbreitung über schwache Zugangsdaten und bekannte Schwachstellen
 - Persistenz, laterale Bewegung, Root-Zugriff
- Shikitega
 - Nutzt Schwachstellen für Privilege Escalation
 - Polymorph, nutzt legitime Cloud-Dienste für C2
 - Führt Metasploit-Meterpreter aus

Tools

Statische Analyse

Einfache Tools

- strings liest lesbare Zeichen aus Binärdateien
- file erkennt Dateityp und Architektur
- binwalk extrahiert eingebettete Dateien und Header

Reverse Engineering Tools

- objdump disassembliert Maschinencode
- Rizin / Cutter Reverse Engineering mit GUI
- Ghidra High-Level Analyse, Quellcode-Rekonstruktion

Verschleierungs- & Verschlüsselungstechniken

- Packing Code ist gepackt, z. B. via UPX
- Polymorphismus Code verändert sich bei jeder Infektion
- Ziel: Erkennung durch Virenscanner erschweren
- Herausforderung: Erfordert Unpacking oder Laufzeit-Analyse

Beispiel 1: Statische Analyse von Prometei

- 428 KB große statisch gelinkte ELF-Datei
- Tools: file , strings , binwalk , Cutter
- Hinweise auf:
 - systemctl enable → Persistenz
 - UPX-Komprimierung erkannt

Prometei: Entpacken im Debugger

Statische Analyse

- UPX verhindert reguläres Entpacken
- Vorgehen:
 - Ausführung in Cutter + Breakpoints auf mprotect
 - Extraktion des entpackten Codes via gdb :

```
dump memory prometei-unpacked.bin <start> <end>
```

Disassemblierung mit objdump

Prometei: Erkenntnisse

- Malware entpackt sich schrittweise zur Laufzeit
- Nur zweiter entpackter Speicherbereich enthält aktiven Code
- Entpackter Code = ca. 20.000 Zeilen Assembler
- Analyse zu aufwendig für vollständige manuelle Auswertung

Beispiel 2: Shikitega

- Sehr kleine Datei: 4.0 KB, ELF, keine Strings
- binwalk : keine Auffälligkeiten
- Keine sichtbaren syscalls in Cutter

Shikitega: Polymorph verschlüsselt

- Nutzt Encoder "Shikata Ga Nai"
 - Polymorph, schwer entpackbar
 - Teil von Metasploit
- Ergebnis:
 - Statische Analyse kaum möglich
 - Dynamische Analyse notwendig

Sichere Testumgebung

- Notwendig für kontrollierte und reproduzierbare Tests
- Schützt Host-System vor Infektionen
- Testumgebungen:
 - Virtuelle Maschinen (VMs)
 - Sandboxing

Tools für die dynamische Analyse

Dynamische Analyse

GDB: Breakpoints, Variablen, Codefluss

Strace: Verfolgt Systemcalls und Signale

• Ftrace: Kernel-Level-Tracing auf Linux-Systemen

Netzwerküberwachung

- Ziel: C2-Kommunikation & verdächtige Netzwerkaktivitäten erkennen
- Tool: Wireshark
 - Echtzeit-Analyse von Netzwerkpaketen
 - Aufdeckung potenzieller externer Verbindungen

Überwachung des Systemverhaltens

- Wichtig zur Erkennung von Datei-, Prozess- und Systemänderungen
- Tool: Auditd (Linux)
 - Überwachung von Dateioperationen
 - Konfigurierbare Ereignisprotokollierung

Beispiel: Prometei Malware

- Analyse zuerst in Any.run
 - Prozessbaum zeigt Start durch uplugplay
- Persistenz durch systemd-Service
- HTTP-Verbindung zu C2-Server (USA)
- Gesendete Systeminfos:
 - OS, CPU, RAM, Laufzeit, VM-Erkennung

Signaturbasierte Erkennung

- Erkennung durch spezifische Signaturen in Malware-Code
- Tools: YARA, ClamAV
- Stärken:
 - Schnelle Erkennung bekannter Malware
 - Unkompliziert
- Schwächen:
 - Unzureichend bei unbekannten Varianten
 - Manipulierbare Signaturen durch Malware-Entwickler

Beispiel: ClamAV-Erkennung

- Befehl: clamscan -v prometei_sample.elf
- Erkennung durch ClamAV:
 - Einige Varianten korrekt identifiziert
 - Andere Varianten nicht erkannt (z. B. durch einfache Änderungen im Code)
- Ergebnis:
 - "Unix.Trojan.Prometei-10042489-0 FOUND" für erkannte Varianten

Verhaltensbasierte Erkennung

- Fokus auf Systemverhalten, nicht nur Signaturen
- Überwacht System- und Audit-Logs
- Erkennung von Anomalien:
 - Plötzliche Berechtigungsänderungen
 - Ungewöhnlich hoher Netzwerkverkehr
 - Unerwartete Dateioperationen
- Vorteile:
 - Erkennung von unbekannter Malware und Zero-Day-Angriffen
- Nachteile:
 - Hohe Fehlalarme möglich

Beispiel: Prometei

- Prometei: Häufig hochgeladene Malware-Variante
- Erkennungsmethoden:
 - ClamAV: Erkennung durch Signatur
 - YARA-Regeln: Erkennung durch spezifische Merkmale

Erkennung mit YARA-Regeln

Malware Erkennung

Beispiel für YARA-Regel, die "uplugplay" als Merkmal nutzt:

```
rule Prometei
{
    strings:
        $binary = "uplugplay"
        $alt_bin = "Bon=UPlug"
    condition:
        $binary or $alt_bin
}
```

- Vorteile:
 - Flexibel und spezifisch für unterschiedliche Malware-Varianten
 - Erkennt auch angepasste Versionen von Prometei, die nicht mehr auf Standard-Signaturen basieren

Beispiel: Shikitega

- Shikitega nutzt Verschleierungstechniken (z.B. polymorphe Encodierung durch Shikata ga Nai)
- Problem: String-basierte YARA-Regeln schlagen fehl, da der Code verschlüsselt und bei jeder Ausführung verändert wird
- Lösung: Erkennung durch musterbasierte YARA-Regeln, die typische XOR-Schleifen erkennen

Fazit

Fazit

- Linux-Malware ist technisch vielfältig
- Prometei: klassisch, gut analysierbar, oft per Signatur erkennbar
- Shikitega: verschleiert, polymorph, schwer zu erkennen
- Analyse muss auf die Malware-Familie abgestimmt sein
- Keine Einheitslösung Analyse muss flexibel angepasst werden
- Kombination aus statischer & dynamischer Analyse notwendig
- Verhaltensbasierte Methoden gewinnen an Bedeutung

Fragen?