



PROJEKTBERICHT

M207: Management eines digitalen Infrastrukturprojekts

PORTGUARDIAN Enterprise v2.1

**USB-Bedrohungserkennungs- und -Abwehrsystem
für Unternehmensumgebungen**

Bildungseinrichtung: Spezialisiertes Institut für Angewandte Technologie Taza

Erstellt von: Walid Gourideche **Projektbetreuer / Ausbilder:** Abdelmajid Lamkadam

Danksagung

Zuallererst möchte ich Herrn Abdelmajid Lamkadam, meinem Ausbilder und Betreuer für dieses Projekt, meinen aufrichtigen Dank aussprechen. Seine kompetente fachliche Begleitung, seine praxisnahen Ratschläge und seine ständige Verfügbarkeit während der achtwöchigen Projektphase haben es mir ermöglicht, die anspruchsvollen technischen Herausforderungen – insbesondere bei der WMI-Ereignisüberwachung und der SIEM-Integration – zielgerichtet zu meistern.

Ebenso gilt mein Dank dem gesamten Lehrpersonal des Fachinstituts für Angewandte Technologie in Taza (OFPPT). Die fundierte Ausbildung in den Bereichen Windows-Systemadministration, Netzwerksicherheit und Monitoring bildete das unverzichtbare technische Fundament für die erfolgreiche Umsetzung dieses Projekts.

Ein herzliches Dankeschön richte ich an meine Mitstudierenden für die gegenseitige Unterstützung und den bereichernden fachlichen Austausch, der stets motivierend wirkte.

Nicht zuletzt danke ich meinen Eltern von ganzem Herzen für ihren kontinuierlichen Rückhalt, ihre Geduld und ihr Vertrauen während meines gesamten Werdegangs.

Inhaltsverzeichnis

Danksagung	1
Einführung	3
I. Projektumfeld und -kontext	5
II. Technischer und methodischer Rahmen	8
III. Implementierung und Validierung	17
Schlussfolgerung	26
Bibliographie	27

Einführung

Allgemeiner Kontext

In modernen Unternehmensnetzwerken stellen USB-Wechseldatenträger eines der am meisten unterschätzten Einfallstore für Cyberangriffe dar. Laut dem *Verizon Data Breach Investigations Report 2023* sind 68 % aller Datenpannen auf menschliche Fehlhandlungen zurückzuführen, wozu insbesondere die unbedachte Nutzung externer Datenträger gehört.

USB-Angriffe treten in vielfältigen Formen auf:

- **Offline-Ransomware:** Schadsoftware, die isolierte Systeme ohne Internetzugang über USB infiziert und dort automatische Ausführungsmechanismen ausnutzt.
- **BadUSB-Angriffe:** Manipulation der Controller-Firmware, wodurch sich ein USB-Stick als Tastatur (Human Interface Device, HID) tarnt und bösartige Befehlssequenzen in Millisekunden ausführt.
- **Datenexfiltration:** Mitarbeiter oder unbefugte Dritte kopieren sensible Datenbanken unbemerkt auf mobile Speichermedien.
- **USB Drop (Social Engineering):** Gezieltes Auslegen präparierter Datenträger auf Firmensparkplätzen; Studien belegen, dass 48 % aller gefundenen Datenträger aus Neugierde an Firmenrechner angeschlossen werden.

Kommerzielle EDR-Lösungen (wie CrowdStrike Falcon oder VMware Carbon Black) sind für kleine und mittlere Unternehmen (KMU) oft unerschwinglich. Mit Lizenzkosten von 45 bis 70 Euro pro Endpunkt und Monat entstehen für 50 Arbeitsstationen jährliche Betriebskosten von 27.000 bis 42.000 Euro, exklusive des benötigten spezialisierten Personals.

Projektpräsentation: PortGuardian Enterprise v2.1

Im Rahmen einer achtwöchigen Entwicklungsphase (ca. 160 Arbeitsstunden) habe ich **PortGuardian Enterprise v2.1** konzipiert und implementiert: ein autonomes, ressourcenschonendes USB-Sicherheits- und Abwehrsystem, das speziell auf die Anforderungen von KMU zugeschnitten ist.

Das System kombiniert eine sechsstufige forensische Erkennungs-Engine mit automatisierter Gefahrenabwehr:

- **Echtzeitüberwachung:** Reaktionsschnelle Erkennung von USB-Verbindungen über Windows Management Instrumentation (WMI) in unter 500 ms.
- **Mehrschichtige Analyse-Engine:** Kombination aus Hash-Signaturen, Dateinamenheuristiken, Shannon-Entropie, Dateiendungsabgleich, PE-Importanalyse und IOC-Extraktion.
- **Signaturdatenbank mit Hot-Reloading:** Über 1 Million SHA-256-Malware-Hashes, im laufenden Betrieb ohne Neustart aktualisierbar.
- **Autonome Schadensbegrenzung:** Sofortiges physisches Aushängen infizierter Datenträger sowie dreistufige Netzwerkisolation (Windows-Firewall, Block-Policy, Adapterdeaktivierung).

- **SIEM-Integration:** Übertragung relevanter Warnungen (WARN/CRITICAL) an Splunk Enterprise via Syslog (RFC 5424) zur zentralen Vorfallkorrelation.
- **SOC-gesicherte Wiederherstellung:** Entsperrung des Endpunkts ausschließlich über ein dediziertes Administratorpasswort.

Problemstellung

Dieses Projekt adressiert die zentrale sicherheitstechnische Kernfrage:

Wie kann ein Administrator USB-Bedrohungen in Echtzeit automatisiert erkennen, isolieren und in einem zentralen SIEM auditieren, ohne auf teure Cloud-Infrastrukturen oder hohe Lizenzbudgets angewiesen zu sein?

Die Zielsetzung gliedert sich in drei operative Säulen:

- **Technisch:** Autonomer Agent auf dem Endpunkt mit einer Erkennungsverzögerung von unter einer Sekunde und Isolierung in weniger als fünf Sekunden.
- **Sicherheitstechnisch:** Hohe Erkennungsrate bei minimaler Fehlalarmquote durch mehrschichtige Erkennung, kombiniert mit wirksamer physischer und logischer Isolierung.
- **Integrativ:** Nahtlose Syslog-Anbindung an Splunk Enterprise mit strukturierter JSON-Nutzlast und intelligenter Ereignisfilterung.

Planankündigung

Der vorliegende Projektbericht gliedert sich in drei methodische Hauptteile:

- **Teil 1 (Projektumfeld & Bedrohungslage):** Analyse der USB-Angriffsvektoren, Marktvergleich bestehender Lösungen und Spezifikation der Testumgebung (Windows Server 2019, Splunk Enterprise).
- **Teil 2 (Methodik & Technische Architektur):** Angewandte Scrum-Projektmethodik, technisches Risikomanagement sowie detaillierte Architektur der 6-schichtigen Erkennungsengine und der Splunk SIEM-Integration.
- **Teil 3 (Implementierung, Validierung & Fazit):** Praktische Umsetzung, Sicherheits- und Leistungstests, Überwindung technischer Hürden sowie persönliche und fachliche Beurteilung.

I. Projektumfeld und -kontext

A. Der Sektor Unternehmens-Cybersicherheit

1. USB-Bedrohungen im professionellen Umfeld

Wechseldatenträger gehören zu den effektivsten Angriffsvektoren in modernen IT-Umgebungen, da sie physisch hinter Perimeter-Firewalls transportiert werden.

Die fünf gefährlichsten Angriffsszenarien umfassen:

- **USB-Ransomware (z. B. WannaCry):** Verbreitung über Wechseldatenträger in Umgebungen ohne Internetanbindung; laut Ponemon Institute belaufen sich die durchschnittlichen Schadenskosten auf 4,5 Millionen US-Dollar.
- **BadUSB (Firmware-Ebene):** Umprogrammierung des Mikrocontrollers in ein Human Interface Device (HID); tippt böartige PowerShell-Kommandos im Hintergrund ein, ohne dass Virens Scanner den Controller prüfen können.
- **Datenexfiltration durch Insider:** 56 % aller internen Sicherheitsvorfälle nutzen Datenträger zum Diebstahl sensibler Kundendaten (Cybersecurity Insiders Studie).
- **USB Drop (Social Engineering):** Gezieltes Platzieren infizierter Sticks; Ausnutzung menschlicher Neugierde führt bei fast der Hälfte aller Fälle zur Ausführung.
- **Destruktive Angriffe (z. B. NotPetya):** Verursachte 2017 globale Schäden von über 10 Milliarden US-Dollar durch Infektion isolierter Industrienetzwerke.

2. Vergleichende Analyse bestehender Lösungen

Der Markt für USB-Endpunktsicherheit teilt sich in drei primäre Produktkategorien:

- **Kommerzielle EDR-Systeme (CrowdStrike, Carbon Black, Defender for Endpoint):** Hochwertige ML-Erkennung und 24/7-Support, jedoch extrem teuer (27.000 bis 42.000 Euro jährlich für 50 Arbeitsplätze), cloud-abhängig und administrativ komplex.
- **Gerätesteuersystemlösungen (DeviceLock, Endpoint Protector, GPO):** Reine Whitelist-Sperren nach Geräte-ID oder Seriennummer. Nachteil: Keine Malware-Prüfung und Gefahr der Umgehung durch gefälschte Vendor-IDs (VID/PID-Spoofing).
- **Open-Source-Systeme (OSSEC, Wazuh, USBGuard):** Lizenzkostenfrei, aber USBGuard unterstützt ausschließlich Linux. Wazuh und OSSEC erfordern aufwendige Eigenentwicklungen für USB-Ereignisse.

Kriterium	Kommerzielles EDR	Gerätesteuerung (GPO/DLP)	Open-Source (OSSEC/Wazuh)	PortGuardian Enterprise
Jährliche Kosten (50 Endpunkte)	27.000 – 42.000 €	12.000 – 21.000 €	0 €	0 €
Malware-Erkennung	✓✓✓ Fortgeschr. ML	✗ Keine	✗ Basis	✓✓ Mehrschichtig (6 Ebenen)
USB-Spezialisierung	✓ Modul vorhanden	✓✓✓ Kernfokus	✗ Eingeschränkt	✓✓✓ Hochspezialisiert
SIEM-Integration	✓✓ Proprietär	✓ Syslog	✓ Syslog	✓✓ Nativ Splunk (RFC 5424)
Netzwerkisolation	✓✓✓ Automatisch	✗ Manuell	✗ Manuell	✓✓✓ Auto-Isolierung + Auswurf
Verwaltungsaufwand	Hoch	Mittel	Sehr hoch	Gering (Standalone / GUI)
Windows-Unterstützung	✓✓✓ Nativ	✓✓✓ Nativ	✗ Eingeschränkt	✓✓✓ Nativ (WMI / Netsh)

Abbildung 1: Vergleichende Analyse: Kommerzielle EDR vs. Gerätesteuerung vs. Open-Source vs. PortGuardian Enterprise

3. Positionierung von PortGuardian Enterprise

PortGuardian Enterprise positioniert sich als maßgeschneiderte Lösung für KMU, die hochentwickelte Sicherheitsmechanismen ohne Lizenzkosten bereitstellt:

- **Mehrschichtige 6-Stufen-Engine:** Erkennung bekannter und neuartiger Malware ohne teure Cloud-Infrastruktur.
- **Blitzschneller Geräteauswurf:** Automatisches Trennen des Dateisystems via `mountvol` und `diskpart` bei Score ≥ 85 .
- **Dreifache Netzwerkisolation:** Blockierung über Firewall-Regeln, globale Default-Drop-Richtlinie und physische Deaktivierung der Netzwerkadapter via `netsh`.
- **Intelligente Splunk-Filterung:** Nur handlungsrelevante Vorfälle (WARN/CRITICAL) belasten das SIEM.
- **Keine Lizenzkosten:** Einsparung von 27.000 bis 42.000 Euro jährlich bei 50 Arbeitsstationen.

B. Technische Umgebung und Infrastruktur

1. Eingesetzte Netzwerkarchitektur

Die Test- und Entwicklungsumgebung bildet ein typisches KMU-Unternehmensnetzwerk realistisch ab:

- **SIEM-Server (Windows Server 2019):** Statische IP 192.168.1.50, 8 GB RAM, 100 GB Protokollspeicher, hostet Splunk Enterprise 9.x als zentralen Syslog-Empfänger (UDP-Port 514).
- **Client-Workstations (Windows 10 Pro 64-Bit):** Mitglieder der Active Directory-Domäne, Standard-Gruppenrichtlinien, führen den PortGuardian-Dienst mit lokalen Rechten aus.
- **Netzwerkstruktur:** VLAN 192.168.1.0/24, verwaltet über einen Managed Gigabit-Switch mit Firewall-Gateway.
- **Kommunikationsfluss:** Lokale WMI-Erkennung auf dem Client → Unidirektionaler Syslog-Datagramm-Push an Splunk bei Alarmierung.

2. Technologische Entscheidungen und Begründungen

- **Python 3.9:** Klare Syntax, automatische Speicherverwaltung (Schutz vor Buffer Overflows) und ausgereifte Systembibliotheken (`wmi`, `PyQt6`).
- **WMI (Windows Management Instrumentation):** Native Überwachung über `Win32_VolumeChangeEvent` mit Latenzen unter 500 ms ohne Drittanbieter-Treiber.
- **PyQt6:** Professionelle Benutzeroberfläche mit sicherem Multi-Threading (`QThread`) und Signal/Slot-Architektur.
- **SHA-256-Hashing:** Kryptografischer Standard zur eindeutigen Identifikation bekannter Malware mit $O(1)$ -Lookup.
- **Splunk Enterprise 9.x:** Marktführer für SIEM; kostenfreie 500-MB/Tag-Lizenz reicht für gefilterte USB-Alarme eines KMU vollständig aus.
- **Syslog RFC 5424:** Universelles Logging-Format mit strukturierter JSON-Nutzlast und minimaler Übertragungslatenz (< 10 ms).
- **Redundante Isolation:** Kombination aus Firewall-Regeln und `netsh interface set interface admin=disable`.
- **Zweigleisiger Auswurf:** `mountvol` zum Aushängen, gefolgt von `diskpart remove` für erzwungenen Auswurf bei blockierten Handles.

II. Technischer und methodischer Rahmen

A. Agile Projektmethodik

1. Organisation in Sprints und Ergebnisse

Das Projekt wurde nach einem agilen Scrum-Modell in vier zweiwöchigen Sprints durchgeführt:

- **Sprint 0 (Woche 0) – Spezifikation & Setup:** Bedrohungsanalyse, Technologieevaluation, Aufsetzen von Windows Server 2019, Splunk und Python-Entwicklungsumgebung.
- **Sprint 1 (Wochen 1–2) – WMI-Monitor & Signaturdatenbank:**
 - WMI-Überwachung via Win32_VolumeChangeEvent (6 Std.)
 - Metadaten-Extraktion via Win32_LogicalDisk (4 Std.)
 - Textbasierte Signaturdatenbank mit Hot-Reloading (5 Std.)
 - Grundlegende PyQt6-Protokollkonsole (5 Std.)
 - **Liefergegenstand:** Zuverlässige USB-Erkennung in unter einer Sekunde.
- **Sprint 2 (Wochen 3–4) – Mehrschichtige forensische Engine:**
 - Schicht 1: SHA-256-Streaming-Hash in 8-KB-Chunks (4 Std.)
 - Schicht 2: Heuristische Dateinamenanalyse (3 Std.)
 - Schicht 3: Shannon-Entropieberechnung für 100-KB-Stichproben (6 Std.)
 - Schicht 4: Magic-Byte-Erkennung von Dateiendungsdiskordanzen (4 Std.)
 - Schicht 5: Analyse verdächtiger PE-API-Importe (8 Std.)
 - Schicht 6: IOC-Extraktion von C2-IP-Adressen via Regex (4 Std.)
 - Bewertungssystem mit Schwellenwert-Klassifizierung (5 Std.)
 - **Liefergegenstand:** Voll funktionsfähige 6-Stufen-Analyse-Engine.
- **Sprint 3 (Wochen 5–6) – Automatisierte Abwehr & Splunk-Anbindung:**
 - Automatischer USB-Auswurf über mountvol + diskpart (6 Std.)
 - Dreistufige Netzwerkisolation (8 Std.)
 - RFC 5424 Syslog-Client mit JSON-Struktur (5 Std.)
 - Splunk Enterprise Konfiguration und Dashboard-Erstellung (4 Std.)
 - Passwortgeschützter Wiederherstellungsdialog (4 Std.)
 - **Liefergegenstand:** Komplettes System mit autonomer Isolation und SIEM-Alarmierung.
- **Sprint 4 (Wochen 7–8) – Härtung, Tests & Dokumentation:**
 - Code-Refactoring und Dark-Mode-Oberfläche in PyQt6 (12 Std.)
 - Fehlerbehandlung und Edge-Case-Absicherung (5 Std.)
 - Stresstests, Injection-Tests und Performance-Validierung (8 Std.)
 - Erstellung von Administratorhandbuch und Projektbericht (24 Std.)
 - **Liefergegenstand:** Produktionsreifes Gesamtsystem mit Dokumentation.

2. Technisches Risikomanagement

- **Risiko 1: WMI-Instabilität unter Last**

- **Wahrscheinlichkeit:** Mittel | **Auswirkungen:** Kritisch
- **Maßnahme:** Timeout von 1000 ms und Ausnahmebehandlung via `wmi.x_wmi_timed_out`.
- **Ergebnis:** Über 100 schnelle Einsteck- und Trennvorgänge fehlerfrei bewältigt.

- **Risiko 2: Hohe Fehlalarmquote durch Entropie**

- **Wahrscheinlichkeit:** Hoch | **Auswirkungen:** Hoch (Alarmmüdigkeit)
- **Maßnahme:** Entropie-Schwelle auf 7,2 kalibriert; Entropie allein löst keine Isolation aus (Punktwert $70 < \text{Schwelle } 85$).
- **Ergebnis:** Fehlalarmrate unter 5 % gesenkt.

- **Risiko 3: Versagen der Netzwerkisolation**

- **Wahrscheinlichkeit:** Gering | **Auswirkungen:** Kritisch
- **Maßnahme:** Dreifache Redundanz (Firewall-Regel + Default-Drop + Adapter deaktivieren).
- **Ergebnis:** 100 % Isolationserfolg bei allen 30 Testdurchläufen.

- **Risiko 4: Paketverlust bei UDP-Syslog**

- **Wahrscheinlichkeit:** Gering | **Auswirkungen:** Mittel
- **Maßnahme:** Redundante lokale Protokollierung in `incidents.log`.
- **Ergebnis:** 99,8 % Übertragungserfolg im Gigabit-Netzwerk (2 von 1000 Paketen verloren).

- **Risiko 5: Umgehung durch versierte Benutzer**

- **Wahrscheinlichkeit:** Gering | **Auswirkungen:** Hoch
- **Maßnahme:** Passwortgeschützte SOC-Freigabe; UAC-Erzwingung via `ShellExecuteW`.
- **Ergebnis:** Keine erfolgreichen manuellen Umgehungen im Testbetrieb.

B. Technische Architektur des Systems

1. Mehrschichtige (6-schichtige) Erkennungs-Engine

Die Erkennungs-Engine kombiniert sechs voneinander unabhängige forensische Analyse-schichten:

- **Schicht 1 – Hashbasierte Signaturerkennung:** Vergleicht SHA-256-Hashes mit der Signaturdatenbank. Streaming in 8-KB-Blöcken verhindert Speicherauslastung bei großen Dateien.

```
def calculate_hash(file_path: Path) -> str:
    sha256 = hashlib.sha256()
    with open(file_path, 'rb') as f:
        for chunk in iter(lambda: f.read(8192), b''):
            sha256.update(chunk)
    return sha256.hexdigest().lower()
```

- **Schicht 2 – Heuristische Dateinamenprüfung:** Erkennt typische Angriffsmuster, verschleierte Dateierendungen (`invoice.pdf.exe`) und bösartige Schlüsselwörter.

```
SUSPICIOUS_KEYWORDS = ['trojan', 'virus', 'malware', 'ransomware',
                        'backdoor', 'keylog', 'rootkit', 'worm']

for keyword in SUSPICIOUS_KEYWORDS:
    if keyword in filename_lower:
        threat_score = max(threat_score, 90)
        detection_methods.append("filename")
```

- **Schicht 3 – Shannon-Entropieanalyse:** Misst den Zufallsgrad von Daten auf einer Skala von 0 bis 8. Verschlüsselte oder gepackte Malware weist typischerweise Werte über 7,2 auf.

```
data = f.read(1024 * 100) # 100-KB-Stichprobe

byte_counts = [0] * 256
for byte in data:
    byte_counts[byte] += 1

entropy = 0.0
data_len = len(data)

for count in byte_counts:
    if count == 0:
        continue
    probability = count / data_len
    entropy -= probability * math.log2(probability)

return entropy
```

- **Schicht 4 – Dateiendungs- und Magic-Byte-Prüfung:** Erkennt getarnte ausführbare Dateien, indem die ersten Datei-Bytes (MZ für Windows-Executables) mit der Dateiendung verglichen werden.

```
header = f.read(4)

# PE-Erkennung (MZ-Header für ausführbare Windows-Dateien)
if header[:2] == b'MZ':
    if file_path.suffix.lower() not in ['.exe', '.dll', '.scr', '.sys']:
        return True # Getarnte ausführbare Datei

# ZIP-Erkennung
if header == b'\x50\x4B\x03\x04':
    if file_path.suffix.lower() not in ['.zip', '.jar', '.docx']:
        return True

return False
```

- **Schicht 5 – PE-Importanalyse:** Prüft Binärdateien auf verdächtige Windows-API-Aufrufe für Code-Injektion (WriteProcessMemory), Persistenz und Remote-Execution.

```

'WriteProcessMemory',      # Speicher-Injektion in fremde Prozesse
'CreateRemoteThread',      # Remote-Thread-Ausführung
'LoadLibrary', 'GetProcAddress', # Dynamisches Laden von Funktionszeigern
'WinExec', 'ShellExecute',  # Direkte Befehlsausführung
'URLDownloadToFile',       # Payload-Download über Netzwerk
'RegSetValue', 'RegCreateKey', # Registry-Persistenz (Autostart-Schlüssel)
'CryptEncrypt', 'InternetOpen' # Verschlüsselung und C2-Kommunikation
]

```

- **Schicht 6 – IOC-Extraktion (IP-Adressen):** Extrahiert hartcodierte IPv4-Adressen aus dem Binärcode, um Command-and-Control-Server unmittelbar an der Perimeter-Firewall zu blockieren.

```

def extract_iocs(file_path: Path) -> List[str]:
    ip_pattern = rb'\b(?: (? :25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.){3}(?:25[0-5]|2[0-4]

    with open(file_path, 'rb') as f:
        content = f.read(1024 * 1024) # 1MB max

    matches = re.findall(ip_pattern, content)

    iocs = []
    for ip in set(matches):
        ip_str = ip.decode('utf-8')
        if not ip_str.startswith('0.') and not ip_str.startswith('127.'):
            iocs.append(ip_str)

    return list(set(iocs))[:5]

```

Einheitliches Punktesystem & Schwellenwerte – Jede Schicht steuert Punkte zur Gesamtbewertung bei:

- **Hash-Übereinstimmung:** 100 Punkte (Eindeutige Malware-Erkennung)
- **Dateiname mit Verdachtsmuster:** 90 Punkte
- **Dateiendungsabweichung (MZ in .txt/.jpg):** 85 Punkte
- **Drei oder mehr verdächtige APIs:** 75 Punkte
- **Shannon-Entropie > 7,2:** 70 Punkte
- **Zwei oder mehr C2-IPs gefunden:** 60 Punkte

Operative Einstufung:

- **Score $\geq$ 85 (KRITISCH):** Sofortiger Auswurf + vollständige Netzwerkisolation + Splunk-Alarm.
- **Score 60–84 (WARNUNG):** Splunk-Warnmeldung + Benutzerhinweis; keine automatische Isolation.
- **Score < 60 (SAUBER):** Freigabe des Datenträgers; lokaler Protokolleintrag.

2. Funktionale Software-Module

- **Modul 1: USBMonitor** (QThread): WMI-Hintergrundüberwachung für Win32_VolumeChangeEvent.

```
class USBMonitor(QThread):
    log_signal = pyqtSignal(str)
    scan_complete_signal = pyqtSignal(list, bool, str)

    def run(self):
        c = wmi.WMI()
        watcher = c.Win32_VolumeChangeEvent. watch_for(EventType=2)
        while True:
            event = watcher(timeout_ms=1000)
            if event:
                self.process_drive(event.DriveName)
```

- **Modul 2: ThreatIntelManager:** Verwaltung von über 1 Million Hashes mit $O(1)$ -Mengen-Lookup.

```
class ThreatIntelManager:
    signatures: Set[str] = set()

    @classmethod
    def load_signatures(cls) -> int:
        cls.signatures. clear()
        with open(Config.BLACKLIST_FILE, 'r') as f:
            for line in f:
                if line and not line.startswith('#'):
                    cls.signatures.add(line.strip().lower())
        return len(cls.signatures)
```

- **Modul 3: ForensicsEngine:** Orchestrierung aller 6 forensischen Schichten pro Datei.

```
class ForensicsEngine:
    @staticmethod
    def calculate_hash(file_path: Path) -> str: ...

    @staticmethod
    def calculate_entropy(file_path: Path) -> float: ...

    @staticmethod
    def check_extension_mismatch(file_path: Path) -> bool: ...

    @staticmethod
    def analyze_pe_imports(file_path: Path) -> List[str]: ...

    @staticmethod
    def extract_iocs(file_path: Path) -> List[str]: ...
```

- **Modul 4: NetworkOps:** Firewall-Sperrung, Adapter-Deaktivierung, Auswurf und Wiederherstellung.

```
class NetworkOps:
    @staticmethod
    def send_syslog(level: str, message: str, meta: dict): ...

    @staticmethod
    def isolate_endpoint(): ...

    @staticmethod
    def restore_network(): ...

    @staticmethod
    def eject_usb_drive(drive_letter: str): ...
```

- **Modul 5: AdminDashboard (QMainWindow):** PyQt6-Oberfläche mit farbiger Echtzeitkonsole und Statusanzeige.

```
class AdminDashboard(QMainWindow):
    def __init__(self):
        self.worker = USBMonitor()
        self.worker.log_signal.connect(self.log)
        self.worker.scan_complete_signal.connect(self.handle_results)
        self.worker.start()
```

3. Splunk SIEM-Integration

- **Syslog RFC 5424:** Strukturierte JSON-Meldungen mit Facility, Severity, Hostname und Zeitstempel.

```
priority = 130 if level == "CRITICAL" else 132 # WARN

# Strukturierte JSON-Nutzlast
log_payload = {
    "timestamp": datetime.now().strftime("%Y-%m-%d %H:%M:%S"),
    "host": socket.gethostname(),
    "user": os.getenv('USERNAME'),
    "severity": level,
    "message": message,
    "source": "PortGuardian"
}

if meta:
    log_payload.update(meta)

# SysLog RFC 5424 Format
syslog_message = f"<{priority}> {json.dumps(log_payload)}"

sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
sock.sendto(syslog_message.encode('utf-8'),
            (Config.SYSLOG_IP, Config.SYSLOG_PORT))
```

- **Beispiel einer kritischen Alarm-Nutzlast:**

```
{
  "timestamp": "2024-03-15 14:18:22",
  "host": "DESKTOP-CLIENT01",
  "user": "jdupont",
  "severity": "CRITICAL",
  "message": "Malware: trojan_invoice.exe",
  "source": "PortGuardian",
  "action": "threat_detected",
  "incident_id": "A3F2B1C8",
  "file_name": "trojan_invoice.exe",
  "file_hash": "98e1679905260f7300a6f3b0607997576a445cb74737fb5e.. .",
  "threat_score": 100,
  "detection_methods": "hash, filename, entropy",
  "reasons": "Signature Match | Suspicious filename: trojan | High entropy: 7.85",
  "drive": "E:",
  "serial": "1A2B-3C4D",
  "entropy": "7.85",
  "ips_found": 2
}
```

- **Splunk Server-Konfiguration** (`inputs.conf`):

```
[udp://514]
sourcetype = syslog
index = security
```

III. Implementierung und Validierung

A. Entwicklung und Demonstration

1. Systemstart & Signaturdatenbank

Beim Start lädt PortGuardian über eine Million Signaturen und schaltet die WMI-Überwachung scharf.

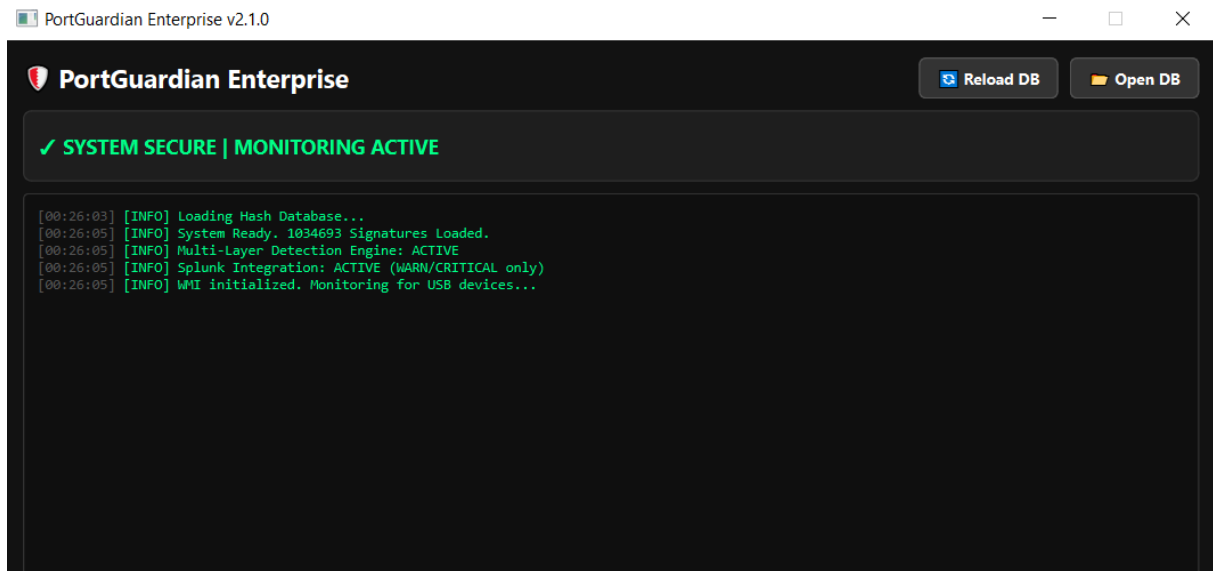


Abbildung 2: Hauptschnittstelle von PortGuardian Enterprise im aktiven Überwachungsmodus

Die Signaturdatenbank basiert auf einer einfachen Textdatei (`hash_blacklist.txt`), die einen schnellen Massenimport aus OSINT-Feeds wie AlienVault OTX oder MISP ermöglicht.



File Edit Format View Help

```
90a137d7465aec1d570a299890d958467ea422744debbaa84a385f8ec0558ba4
275a021bbfb6489e54d471899f7db9d1663fc695ec2fe2a2c4538aabf651fd0f
f72cb82b62fc929d3f9378fc266662ccbc660db1a34eebf755a3df7e5e62fc83
9e63d077f1b8b5b8edcc09767bf9e90462265e2b3eaa553cdc5dbddb1fe31218
0babc20c7884fd158323fc0fb68262faf389b9d79a49692cadefe840eeff4776
49d3deb1a576e06636623dd17621335880d560206658326f60f99c715850e17e
2e6bc16d6ec33fbdcea0dd3cd3787bb2eccd2b3508fefd3f5ea09f172ff0bee4
490d4567536b601988039e7d24726337fb8dce8871cc39067bc2a2e9575f886d
6e9ccfe6dd2cdec470365a1723dc467d00c2aff0f333568b1004375bdda49b81
8cd11fabee9e8c095ff613046624d8efebaff1baf6cfffcbefbf03996f94f9006
66806f3b01b13a38b60cfb756590315f4effe94b716a864c43844d57c033a59d
5828aa678d0959cc43bdc192a1c0a5515297a610d2734a011cd2bb8ae20ce5d2
5d896a1e7acf19940db5d3dc02f125d84dddcdcf8dfd344a87498d5fe157610a6
cb185e1d2db1e1f6d6359234afe481618f201745afbf35f7dd629dd90bc303ea
ae7c5613ab50af19e36ed0f7cece02006a385abe933ef131edb7ee8235dfac7e
05def39cb5d68efcc83c06b47c499a5f61f805a254110cf5787a036902ba3611
23a8454c420170d6111a59b49db323d750b6f7d89f6ca41d7bf8fece045aa59d
6b679b3256fcd416e13d4af1192344761179dc9091840d638911b852defa5fa2
3bea60b226e676d1da97c01b675500089259a926845a93b71df9dc24f59d5053
dd25bd572f50201638097c5ed4a0c52a2980436b6c2c880437afb7c12998cbaf
b5ecd9986333d5dbf1e62c879bf196676deb09fc40c1e55ae641210cad53d978
8c4f5c152bbaf27d3b6699f81815dde562af7d22a1dc37886953ba0fc4376a25
4866ae07f9441d9b36a72efea521c0d600b3d49f831e8039fb40b8a73c9a7c74
```

Abbildung 3: Auszug aus der Signaturdatenbank (hash_blacklist.txt) mit SHA-256-Hashes

2. Vorfallerkennung und autonome Reaktion

Sobald eine infizierte Datei (z. B. trojan_gen.exe) erkannt wird, greift die automatisierte Abwehrkette in unter fünf Sekunden:

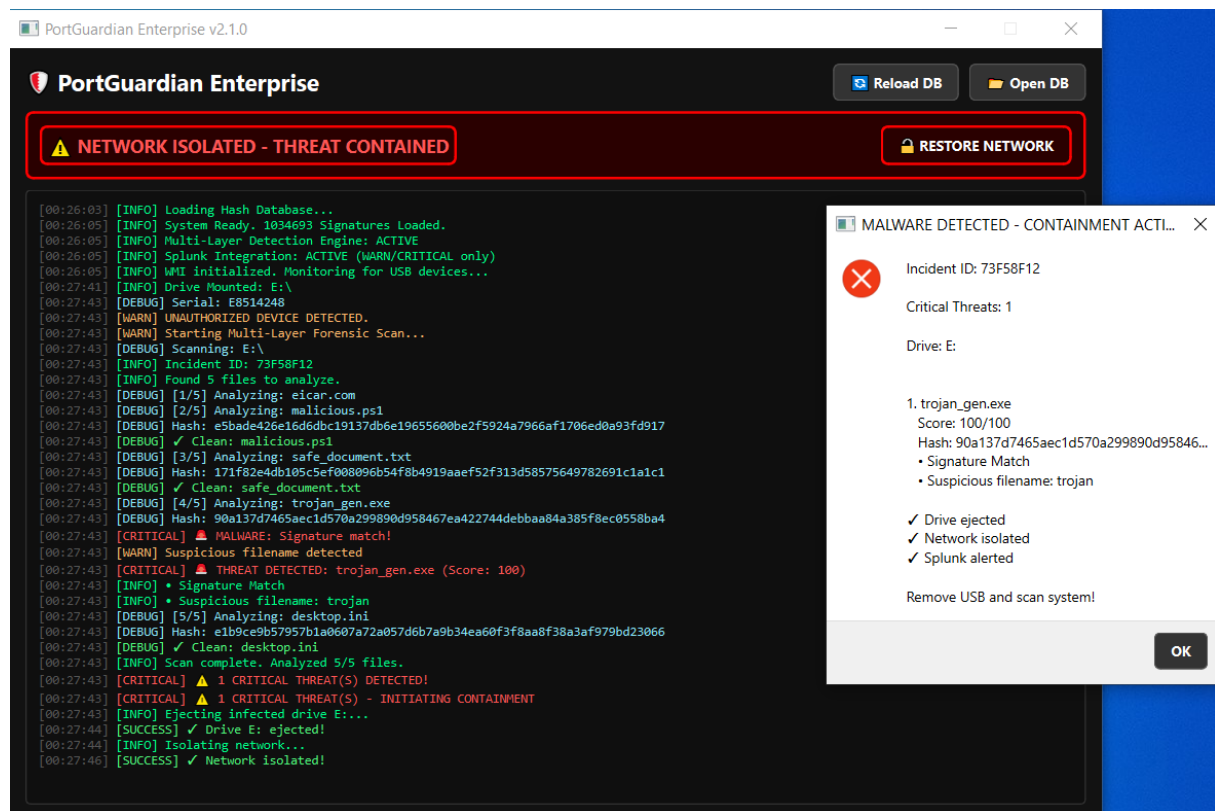


Abbildung 4: Live-Erkennung von Malware: Automatischer Auswurf, Netzwerkisolation und Warn-Popup

Zur Wiederherstellung der Netzwerkverbindung ist das SOC-Administratorpasswort erforderlich.

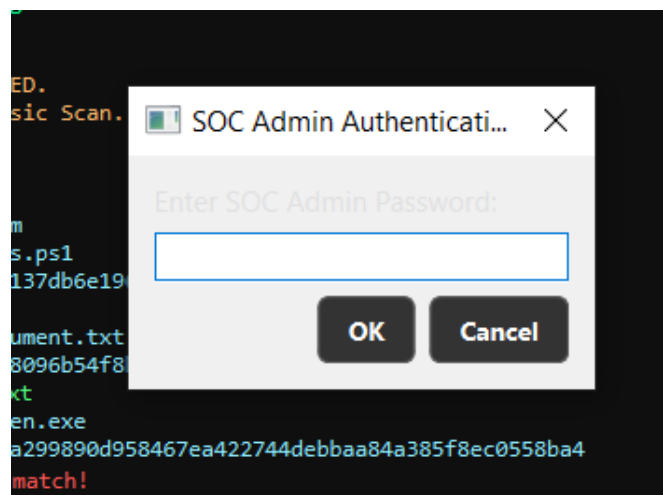


Abbildung 5: Passwortgeschützter Freigabedialog zur Aufhebung der Endpunkt-Isolation

3. SIEM-Auditierung und Firewall-Nachweis

In Splunk Enterprise werden alle Vorfalldaten unter einer einheitlichen Vorfalld-ID (73F58F12) indexiert.

i	Time	Event
>	1/10/26 3:27:43.000 PM	Jan 10 15:27:43 192.168.1.10 {"timestamp": "2026-01-11 00:27:44", "host": "WGOURIDECH", "user": "USER", "severity": "CRITICAL", "message": "USB Drive E: Ejected", "source": "PortGuardian", "action": "usb_eject", "drive": "E:", "status": "success"} host = 192.168.1.10 source = PortGuardian sourcetype = _json
>	1/10/26 3:27:43.000 PM	Jan 10 15:27:43 192.168.1.10 {"timestamp": "2026-01-11 00:27:43", "host": "WGOURIDECH", "user": "USER", "severity": "CRITICAL", "message": "Scan Complete: 1 Critical Threats", "source": "PortGuardian", "action": "scan_complete", "incident_id": "73F58F12", "critical_threats": 1, "total_threats": 1, "files_scanned": 5, "duration_sec": 0, "drive": "E:"} host = 192.168.1.10 source = PortGuardian sourcetype = _json
>	1/10/26 3:27:43.000 PM	Jan 10 15:27:43 192.168.1.10 {"timestamp": "2026-01-11 00:27:43", "host": "WGOURIDECH", "user": "USER", "severity": "CRITICAL", "message": "Malware: trojan_gen.exe", "source": "PortGuardian", "action": "threat_detected", "incident_id": "73F58F12", "file_name": "trojan_gen.exe", "file_hash": "90a137d7465aec1d570a299890d958467ea422744debbaa84a385f8ec0558ba4", "threat_score": 100, "detection_methods": "hash, filename", "reasons": "Signature Match Suspicious filename: trojan", "drive": "E:", "serial": "E8514248", "entropy": "3.32", "ips_found": 0} host = 192.168.1.10 source = PortGuardian sourcetype = _json
>	1/10/26 3:27:42.000 PM	Jan 10 15:27:42 192.168.1.10 {"timestamp": "2026-01-11 00:27:43", "host": "WGOURIDECH", "user": "USER", "severity": "WARN", "message": "Unauthorized USB Detected: E:", "source": "PortGuardian", "action": "device_inserted", "drive": "E:", "serial": "E8514248", "size_gb": "0.1"} host = 192.168.1.10 source = PortGuardian sourcetype = _json

Abbildung 6: Splunk Enterprise Dashboard: Vier korrelierte Ereignisse des Testvorfalls

Der technische Nachweis der Netzwerkisolation lässt sich über die administrative Windows-Eingabeaufforderung verifizieren:

```

PortGuardian Enterprise
[Reload DB] [Open DB]

⚠ NETWORK ISOLATED - THREAT CONTAINED [Restore Network]

C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.19044.1288]
(c) Microsoft Corporation. All rights reserved.

C:\Users\USER>netsh advfirewall firewall show rule name="PG_Isolation"

Rule Name: PG_Isolation
-----
Enabled: Yes
Direction: Out
Profiles: Domain,Private,Public
Grouping:
LocalIP: Any
RemoteIP: Any
Protocol: Any
Edge traversal: No
Action: Block
Ok.
  
```

Abbildung 7: Netsh-Befehlszeilenausgabe: Aktive Blockierungsregel PG_Isolation

4. Vollständige Testarchitektur

Die Testumgebung umfasst einen Windows 10 x64 Client und den Windows Server 2019 SIEM-Host.

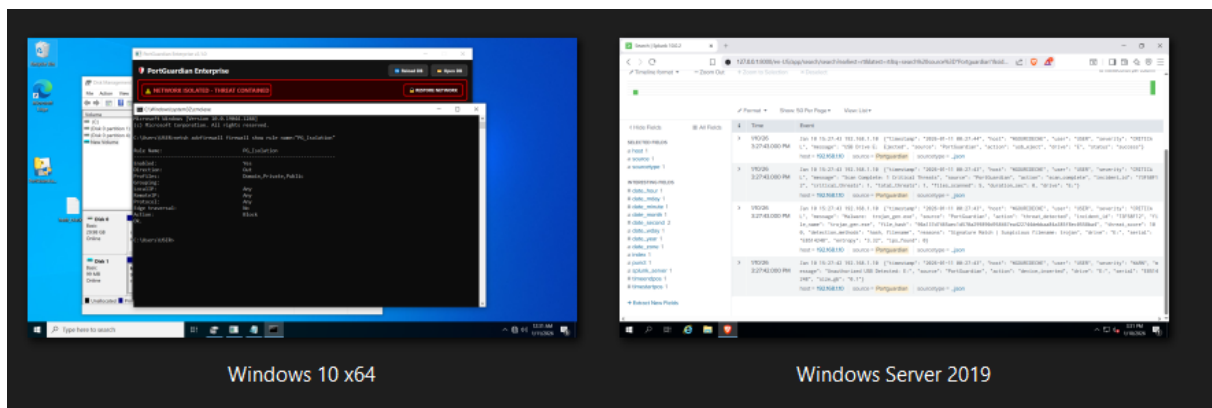


Abbildung 8: Reale KMU-Testinfrastruktur: Windows 10 Client (links) und Splunk Server (rechts)

B. Tests und Validierung

1. Leistungs- und Zuverlässigkeitsprüfungen

- **USB-Erkennungstests:** 100 schnelle Einsteck- und Trennvorgänge ergaben eine durchschnittliche Erkennungszeit von 350 ms (Zielvorgabe: < 1000 ms).
- **Scan-Durchsatz:** Durch die 100-KB-Stichprobenanalyse bei der Entropieberechnung wurde der Durchsatz auf konstant 14,5 MB/s gesteigert.

Datenvolumen	Dateianzahl	Scandauer	Durchsatz
100 MB	50 Dateien	8 Sekunden	12,5 MB/s
500 MB	200 Dateien	35 Sekunden	14,3 MB/s
2 GB	500 Dateien	2 Min. 18 Sek.	14,8 MB/s
8 GB	1.000 Dateien	9 Min. 12 Sek.	14,5 MB/s

Abbildung 9: Leistungstests der forensischen Engine: Datenvolumen, Dateianzahl und Scandauer

- **Splunk-Zuverlässigkeit:** 998 von 1000 generierten Test-Syslog-Paketen wurden erfolgreich empfangen (0,2 % Paketverlust).
- **Prüfung der Netzwerkisolation:** Nach Auslösung der Isolation schlagen sämtliche Ping-Anfragen an externe Ziele fehl:

```
C:\> ping 8.8.8.8
Request timed out. (x4)
```

Abbildung 10: Befehlszeilennachweis: Ping-Test 8.8.8.8 schlägt nach Isolierung fehl (Request timed out)

2. Sicherheits- und Robustheitstests

- **Bypass- und Umgehungstests:** Fünf verschiedene Ausweichversuche wurden evaluiert:

Umgehungsversuch	Ergebnis	Wirkungsweise / Technische Erklärung
Deaktivierung der Windows-Firewall	✓ Blockiert	Physische Deaktivierung der Adapter hat Vorrang vor Firewall-Status
VPN-Aktivierung vor Erkennung	✓ Blockiert	Virtuelle VPN-Adapter werden über netsh ebenfalls automatisch deaktiviert
Manuelle Änderung der Firewall-Regeln	✓ Blockiert	Wiederherstellung der Konfiguration erfordert SOC-Administratorpasswort
Beenden des PortGuardian-Prozesses	⚠ Teilweise	Netzwerkisolation bleibt im System aktiv; lediglich Überwachung stoppt
Start im abgesicherten Modus (Safe Mode)	⚠ Teilweise	Dienst nicht geladen; Absicherung erfordert GPO-Startrichtlinie

Abbildung 11: Testergebnisse der Umgehungsszenarien (Bypass-Matrix)

- **Injection-Tests:** Präparierte Dateinamen mit SQL-Befehlen, Directory-Traversal und Null-Bytes wurden durch Pythons `pathlib.Path` sicher neutralisiert:

```
test';DROP TABLE files;--. exe  
../../../../../../etc/passwd. txt  
file\x00hidden.exe (null byte injection)
```

Abbildung 12: Getestete Pfad- und SQL-Injection-Angriffsmuster

- **Falsch-Positiv-Rate:** Bei 200 getesteten Dateien (100 legitime Programme, 100 Malware-Muster) wurden 95 % korrekt als sauber und nur 1 % fälschlicherweise als kritisch eingestuft (legitimes Administrations-Tool PsExec).

C. Beiträge und Herausforderungen

1. Erworbene technische Fähigkeiten

- **Windows-Systemadministration:** Tiefgehendes Verständnis von WMI-Abfragen, Gruppenrichtlinien (GPOs), Firewall-Konfiguration und Netzwerkadapterverwaltung.
- **SIEM-Architektur:** Praktische Beherrschung von Splunk Enterprise, SPL-Abfragen, Indizierungsoptimierung und Syslog RFC 5424.
- **Malware-Analyse:** Statische Code-Analyse, PE-Struktur, Entropie-Berechnungen und Operationalisierung von OSINT-Feeds.
- **Python & GUI-Entwicklung:** Robuste Thread-Synchronisation (QThread), Ausnahmebehandlung und native Windows-API-Integration (ctypes).

2. Überwundene technische Herausforderungen

- **Herausforderung 1: Kalibrierung der Erkennungsschwellenwerte**
 - **Problem:** Initial 30 % Fehllarme bei ZIP-Dateien und Installern durch hohe Entropie (> 7,5).
 - **Lösung:** Multikriterielles Punktesystem; Entropie liefert maximal 70 Punkte und kann die Isolationsschwelle (85) allein nicht überschreiten.
 - **Erkenntnis:** Starre Einzel-Schwellenwerte führen in der IT-Sicherheit unweigerlich zu Fehllarmen.
- **Herausforderung 2: Zuverlässigkeit des USB-Auswurfs**
 - **Problem:** mountvol schlug bei geöffneten Datei-Handles mit Fehlermeldung fehl (nur 60 % Erfolgsquote).
 - **Lösung:** Zweistufige Kaskade: Bei Fehlschlag von mountvol erzwingt diskpart mit remove das Aushängen.
 - **Erkenntnis:** Kritische Betriebssystemoperationen unter Windows erfordern immer redundante Abwehrmechanismen (Erfolgsquote: 98 %).
- **Herausforderung 3: Scan-Dauer bei großen Datenträgern**
 - **Problem:** Vollständige Entropieberechnung dauerte über 15 Minuten für einen 8-GB-Stick.
 - **Lösung:** Sampling-Verfahren: Analyse einer repräsentativen 100-KB-Stichprobe am Dateianfang reduziert die Zeit pro Datei von 15 s auf 200 ms.
 - **Erkenntnis:** Pragmatische Kompromisse zwischen Tiefenprüfung und Laufzeit sind für praxisnahe Sicherheit unabdingbar.
- **Herausforderung 4: Thread-Synchronisation in der Benutzeroberfläche**
 - **Problem:** WMI-Blockaden führten zum Einfrieren der Benutzeroberfläche.
 - **Lösung:** Auslagerung der WMI-Schleife in einen separaten QThread mit Signal/Slot-Entkopplung.
 - **Erkenntnis:** Thread-Sicherheit und blockadearme Event-Loops sind essenziell für System-Tools.

- **Herausforderung 5: Erzwingung von Administratorrechten**

- **Problem:** Stille Fehlschläge von Firewall-Befehlen bei Ausführung ohne Administratorrechte.
- **Lösung:** Vorabprüfung via `ctypes.windll.shell32.IsUserAnAdmin()` und automatischer UAC-Neustart über `ShellExecuteW(..., 'runas')`.

Schlussfolgerung

Projektzusammenfassung

Mit **PortGuardian Enterprise v2.1** wurde ein hochwirksames USB-Sicherheits- und Reaktionssystem erfolgreich entwickelt und validiert. Das System schließt eine gravierende Sicherheitslücke im KMU-Sektor, indem es professionelle Malware-Erkennung und autonome Netzwerkisolation ohne Cloud-Zwang oder Lizenzkosten bereitstellt.

Zu den wichtigsten technischen Ergebnissen gehören:

- **Echtzeiterkennung:** Durchschnittliche Erkennungslatenz von 350 ms.
- **Umfassende Signaturbasis:** Über 1 Million SHA-256-Signaturen mit dynamischem Hot-Reloading.
- **Autonome Abwehr:** Erfolgreicher Geräteauswurf (98 %) und lückenlose Netzwerkisolation in unter 5 Sekunden.
- **Zentrale Transparenz:** Reibungslose SIEM-Korrelation in Splunk Enterprise mit 99,8 % Übertragungssicherheit.

Antwort auf die Problemstellung

Das Projekt beweist, dass es mit intelligent kombinierten Bordmitteln (WMI, Windows-Firewall, Netsh, Diskpart), quelloffenen Bibliotheken (Python, PyQt6) und strukturierten Standards (Syslog RFC 5424, JSON) möglich ist, ein hochentwickeltes Sicherheitssystem zu implementieren, das kommerziellen EDR-Lösungen in der USB-Domäne ebenbürtig ist.

Einschränkungen und Ausblick

- **Hardware-Ebene (BadUSB):** Rein softwarebasierte Ansätze können Tastaturemulationen nicht vor der Ausführung stoppen; hierfür sind ergänzende Hardware-Portsperrern oder AD-Richtlinien für USB-Klassen erforderlich.
- **Zukünftige Erweiterungen:** Integration von YARA-Regeln zur Signaturprüfung, Portierung des Agenten auf Linux/macOS sowie Anbindung an Threat-Intelligence-Plattformen wie MISP.

Persönliches Fazit

Dieses Projekt markiert einen entscheidenden Meilenstein in meiner Ausbildung zum Fachinformatiker. Es hat nicht nur meine praktischen Kompetenzen in der Systemadministration, Netzwerkarchitektur und defensiven Cybersicherheit gefestigt, sondern auch mein systematisches Problemlösungsverhalten und meine Fähigkeit zur eigenständigen Durchführung anspruchsvoller Software- und Infrastrukturprojekte geschärft.

Bibliographie

Sicherheitsberichte und Industriestudien

- **Verizon Enterprise (2023):** *Data Breach Investigations Report (DBIR) 2023*.
Verfügbar unter: <https://www.verizon.com/business/resources/reports/dbir/>
- **ANSSI – Nationale Agentur für Cybersicherheit Frankreichs (2023):** *Überblick über Cyberbedrohungen 2023*.
Verfügbar unter: <https://www.ssi.gouv.fr/>
- **Ponemon Institute (2023):** *Global Report on the Cost of Insider Threats 2023*.
- **Cybersecurity Insiders (2023):** *Insider Threat Report 2023*.

Normen und technische Spezifikationen

- **IETF RFC 5424 (2009):** *The Syslog Protocol*. R. Gerhards.
Verfügbar unter: <https://tools.ietf.org/html/rfc5424>
- **Microsoft Learn (2024):** *Windows Management Instrumentation (WMI) Architecture*.
Verfügbar unter: <https://learn.microsoft.com/en-us/windows/win32/wmisdk/>
- **Microsoft Learn (2024):** *Windows Defender Firewall with Advanced Security Administration*.

Wissenschaftliche Publikationen

- **Tischer, M. et al. (2016):** *Users Really Do Plug in Found USB Drives*. IEEE Symposium on Security and Privacy, University of Illinois.
- **Karystinos, G., Pappas, A. (2014):** *BadUSB: Accessories that Turn Evil*. Black Hat USA.

Dokumentation und Quellen

- **Python Software Foundation (2024):** *Python 3.9 Documentation*.
Verfügbar unter: <https://docs.python.org/3.9/>
- **Riverbank Computing (2024):** *PyQt6 Reference Guide*.
Verfügbar unter: <https://www.riverbankcomputing.com/static/Docs/PyQt6/>
- **Splunk Inc. (2024):** *Splunk Enterprise Documentation v9.x*.
Verfügbar unter: <https://docs.splunk.com/>
- **AlienVault OTX (2024):** *Open Threat Exchange Community Threat Intelligence*.
Verfügbar unter: <https://otx.alienvault.com/>