



PROJECT REPORT

M207: Digital Infrastructure Project Management

PORTGUARDIAN Enterprise v2.1

USB Threat Detection and Automated Containment System
for Enterprise Environments

Educational Institution: Specialized Institute of Applied Technology Taza (OFPPT)

Author: Walid Gourideche **Project Supervisor / Trainer:** Abdelmajid Lamkadam

Acknowledgements

First and foremost, I would like to express my deepest gratitude to Mr. Abdelmajid Lamkadam, my instructor and project supervisor. His invaluable technical guidance, pragmatic advice, and constant support throughout this eight-week engineering journey enabled me to navigate and overcome complex systems-level challenges – particularly in WMI event interception, multi-threaded GUI pipelines, and SIEM correlation.

My sincere thanks also extend to the entire academic faculty of the Specialized Institute of Applied Technology in Taza (OFPPT). The rigorous curriculum in Windows systems administration, enterprise networking, and defensive cybersecurity provided the indispensable technical foundation required to bring this project to fruition.

I also extend warm appreciation to my fellow students for their continuous camaraderie, constructive feedback, and enriching technical exchanges throughout the training program.

Finally, I express my profound gratitude to my parents for their unconditional encouragement, patience, and confidence in my abilities throughout my academic journey.

Contents

Acknowledgements	1
Introduction	3
I. Project Environment and Context	5
II. Technical and Methodological Framework	8
III. Implementation and Validation	17
Conclusion	26
Bibliography	27

Introduction

General Context

In contemporary enterprise environments, removable USB storage media represent one of the most frequently underestimated ingress vectors for cyber threats. According to the *Verizon Data Breach Investigations Report (DBIR) 2023*, 68 % of all security incidents involve human error – prominent among which is the indiscriminate connection of untrusted external drives.

USB-borne attack vectors manifest across several critical attack classes:

- **Offline Ransomware:** Malware engineered to compromise isolated networks lacking direct Internet access via removable drives, weaponizing autorun or human interaction.
- **BadUSB Attacks:** Microcontroller firmware manipulation that allows a USB thumb drive to emulate a Human Interface Device (HID keyboard), injecting malicious terminal commands in milliseconds.
- **Insider Data Exfiltration:** Rogue employees or unauthorized contractors surreptitiously staging and copying confidential intellectual property to portable mass storage.
- **USB Drop Attacks (Social Engineering):** Adversaries leaving weaponized USB drives on corporate grounds; empirical studies reveal that 48 % of discovered media are plugged into enterprise endpoints out of curiosity.

Commercial Endpoint Detection and Response (EDR) platforms (such as CrowdStrike Falcon or VMware Carbon Black) impose prohibitive licensing costs on small and medium-sized enterprises (SMEs). Averaging \$45 to \$70 per endpoint per month, a modest deployment of 50 workstations incurs annual operational expenditures of \$27,000 to \$42,000, excluding specialized SOC operations personnel.

Project Overview: PortGuardian Enterprise v2.1

Over an eight-week engineering cycle (spanning approximately 160 hours of focused development), I engineered and implemented **PortGuardian Enterprise v2.1**: an autonomous, lightweight USB security and automated containment platform engineered specifically to satisfy enterprise and SME operational constraints.

The system tightly couples a modular 6-layer forensic analysis engine with immediate autonomous threat containment:

- **Real-Time Bus Interception:** Sub-500 ms peripheral attachment detection via Windows Management Instrumentation (WMI).
- **Multi-Layer Forensic Engine:** Cascaded analysis spanning cryptographic hashes, heuristic naming flags, Shannon entropy calculation, file extension-to-magic-byte cross-validation, PE import table inspection, and embedded C2 IOC extraction.
- **Hot-Reloadable Threat Intelligence:** Direct lookup across over 1,000,000 known SHA-256 malware signatures without requiring service restarts.
- **Autonomous Incident Containment:** High-speed filesystem dismount via disk partitioning

APIs coupled with three-tier host network isolation (Windows Firewall rules, default-drop state, and physical network adapter disabling).

- **Centralized SIEM Integration:** Forwarding actionable alerts (WARN/CRITICAL) directly to Splunk Enterprise via standard RFC 5424 Syslog payloads.
- **SOC-Authenticated Restoration:** Hardened endpoint unlocking accessible exclusively through dedicated administrative authentication.

Problem Statement

This project directly tackles the central cybersecurity operational dilemma:

How can a systems administrator automatically detect, contain, and centrally audit USB-based cyber threats in real time without incurring exorbitant licensing budgets or depending on cloud-tethered infrastructure?

The operational objectives are structured along three pillars:

- **Technical Pillar:** Autonomous endpoint daemon delivering sub-second detection and sub-5-second host containment.
- **Defensive Pillar:** Maximized threat detection accuracy combined with a strictly minimized false-positive rate, enforced via physical and logical quarantine.
- **Integration Pillar:** Standardized, reliable Syslog transmission to Splunk Enterprise featuring structured JSON metadata and noise filtering.

Report Outline

This engineering report is organized into three primary sections:

- **Part 1 (Threat Landscape & Project Scope):** Systematic examination of USB attack surfaces, a comparative analysis of commercial vs. open-source tools, and enterprise lab testbed specifications (Windows Server 2019, Splunk Enterprise).
- **Part 2 (Methodology & Technical Architecture):** Agile Scrum lifecycle execution, defensive risk governance, modular 6-stage detection engine mechanics, and SIEM pipeline integration.
- **Part 3 (Implementation, Validation & Conclusion):** Practical deployment benchmarks, evasion and injection resilience testing, operational obstacles overcome, and technical conclusions.

I. Project Environment and Context

A. The Enterprise Cybersecurity Sector

1. USB Threats in Professional Environments

Removable media remain among the most dangerous adversarial vectors because physical media completely bypass external perimeter firewalls.

The five primary USB attack archetypes threatening enterprises include:

- **USB Ransomware (e.g., WannaCry, NotPetya):** Rapid propagation across internal air-gapped subnets; according to Ponemon Institute, the global average incident remediation cost exceeds \$4.5 million.
- **BadUSB (Firmware Injection):** Firmware reprogramming that converts microcontrollers into HID keyboards, silently executing PowerShell payloads with no antivirus visibility into hardware internals.
- **Insider Exfiltration:** 56 % of insider-driven security compromises leverage USB storage media to leak proprietary intellectual property and customer databases (Cybersecurity Insiders).
- **USB Drop (Social Engineering):** Baiting office perimeters with infected storage drives; curiosity triggers drive execution in nearly half of all unmanaged scenarios.
- **Destructive ICS/SCADA Payloads:** Removable drives used to inject destructive industrial firmware, causing catastrophic operational disruption.

2. Comparative Analysis of Existing Solutions

Current endpoint USB security tools fall into three primary market tiers:

- **Commercial EDR Platforms (CrowdStrike, Carbon Black, Defender for Endpoint):** Sophisticated ML engines and 24/7 technical support, but burdened by heavy annual subscriptions (\$27,000 to \$42,000 for 50 seats), cloud latency, and high configuration complexity.
- **Device Control Suites (DeviceLock, Endpoint Protector, GPO):** Static whitelisting based on vendor IDs (VID/PID) or serial numbers; unable to inspect file contents and vulnerable to device identity spoofing.
- **Open-Source Tools (OSSEC, Wazuh, USBGuard):** Free of licensing costs, yet USBGuard is strictly restricted to Linux. Wazuh and OSSEC require complex custom scripting to handle dynamic Windows USB events.

Evaluation Criterion	Commercial EDR	Device Control (GPO/DLP)	Open-Source (OSSEC/Wazuh)	PortGuardian Enterprise
Annual Cost (50 Endpoints)	\$27,000 – \$42,000	\$12,000 – \$21,000	\$0	\$0
Malware Detection	✓✓✓ Advanced ML	✗ None	✗ Basic	✓✓ Multi-Layer (6 Stages)
USB Specialization	✓ Module Available	✓✓✓ Core Focus	✗ Limited	✓✓✓ Highly Specialized
SIEM Integration	✓✓ Proprietary API	✓ Syslog	✓ Syslog	✓✓ Native Splunk (RFC 5424)
Network Isolation	✓✓✓ Automatic	✗ Manual	✗ Manual	✓✓✓ Auto-Isolation + Eject
Administrative Overhead	High	Medium	Very High	Low (Standalone / GUI)
Windows Support	✓✓✓ Native	✓✓✓ Native	✗ Limited	✓✓✓ Native (WMI / Netsh)

Figure 1: Comparative Evaluation: Commercial EDR vs. Device Control vs. Open-Source vs. PortGuardian Enterprise

3. Positioning of PortGuardian Enterprise

PortGuardian Enterprise establishes a tailored security tier for SMEs, offering advanced threat mitigation without licensing barriers:

- **Modular 6-Stage Forensic Engine:** Comprehensive static analysis identifying known malware and zero-day threats on-premises.
- **Rapid Filesystem Dismount:** Automated, graceful and forced disk dismount using `mountvol` and `diskpart` when risk score ≥ 85 .
- **Three-Tier Network Quarantine:** Host isolation enforced through firewall rule enforcement, global default-drop blocking, and physical interface disabling via `netsh`.
- **Smart SIEM Ingestion:** Forwarding only actionable incidents (WARN/CRITICAL) to minimize SIEM indexing quotas.
- **Zero Licensing Overhead:** Delivering \$27,000 to \$42,000 in annual recurring savings for a 50-workstation deployment.

B. Technical Environment and Infrastructure

1. Deployed Network Architecture

The enterprise testing testbed accurately mirrors a realistic SME corporate environment:

- **SIEM Server (Windows Server 2019):** Static IP 192.168.1.50, 8 GB RAM, 100 GB storage, hosting Splunk Enterprise 9.x configured as a central Syslog collector on UDP port 514.
- **Client Endpoints (Windows 10 Pro 64-Bit):** Active Directory domain members running baseline security policies and executing the PortGuardian service with local administrative privileges.
- **Network Topology:** Enterprise VLAN 192.168.1.0/24, managed through a Gigabit switch with dedicated firewall routing.
- **Operational Telemetry Pipeline:** Client-side WMI hardware monitoring → Local forensic pipeline → Asynchronous UDP Syslog transmission to Splunk.

2. Technology Stack Decisions & Rationale

- **Python 3.9:** Clean syntax, native memory management mitigating buffer overflows, and rich native systems libraries (`wmi`, `PyQt6`).
- **WMI (Windows Management Instrumentation):** Native OS hardware notifications via `Win32_VolumeChangeEvent` achieving sub-500 ms responsiveness without third-party kernel drivers.
- **PyQt6:** Modern graphical interface leveraging robust multi-threading (`QThread`) and thread-safe signal/slot mechanics.
- **SHA-256 Hashing:** Cryptographic standard for exact identification of known malware with $O(1)$ set lookup efficiency.
- **Splunk Enterprise 9.x:** Enterprise SIEM standard; its free 500 MB/day license comfortably supports filtered USB telemetry across SME environments.
- **Syslog RFC 5424:** Industry-standard logging format packaging structured JSON telemetry with sub-10 ms transmission latency.
- **Redundant Network Isolation:** Two-fold defense combining Windows Firewall drop rules with administrative interface disabling via `netsh`.
- **Two-Stage Disk Dismount:** Graceful dismount via `mountvol`, immediately backed by forced volume detachment via `diskpart` if file handles remain locked.

II. Technical and Methodological Framework

A. Agile Project Methodology

1. Sprint Organization and Deliverables

The project was structured across a four-iteration Agile Scrum roadmap, executed in bi-weekly sprints:

- **Sprint 0 (Week 0) – Specification & Environment Setup:** Threat surface analysis, tech stack validation, provisioning of Windows Server 2019, Splunk Enterprise, and Python tooling.
- **Sprint 1 (Weeks 1–2) – WMI Event Monitor & Signature Database:**
 - Hardware event interception via `Win32_VolumeChangeEvent` (6 hrs)
 - Drive metadata extraction using `Win32_LogicalDisk` (4 hrs)
 - High-speed text-based signature database with hot-reloading (5 hrs)
 - Core PyQt6 administrative logging console (5 hrs)
 - **Key Deliverable:** Reliable sub-second USB drive detection and metadata harvesting.
- **Sprint 2 (Weeks 3–4) – Multi-Layer Forensic Engine:**
 - Layer 1: Streaming SHA-256 chunked hashing in 8 KB blocks (4 hrs)
 - Layer 2: Heuristic naming pattern matching (3 hrs)
 - Layer 3: Shannon entropy calculation over 100 KB samples (6 hrs)
 - Layer 4: Magic-byte inspection for file extension discordance (4 hrs)
 - Layer 5: Portable Executable (PE) API import table inspection (8 hrs)
 - Layer 6: C2 IP address extraction via regular expressions (4 hrs)
 - Composite scoring algorithm and classification thresholds (5 hrs)
 - **Key Deliverable:** Fully functional 6-stage static analysis engine.
- **Sprint 3 (Weeks 5–6) – Automated Containment & Splunk SIEM Integration:**
 - Automated USB ejection via `mountvol` and `diskpart` (6 hrs)
 - Three-tier host network quarantine (8 hrs)
 - RFC 5424 Syslog UDP client with JSON payloads (5 hrs)
 - Splunk Enterprise dashboard creation and data parsing (4 hrs)
 - Password-protected SOC unlock dialog (4 hrs)
 - **Key Deliverable:** Autonomous containment engine and real-time SIEM auditing.
- **Sprint 4 (Weeks 7–8) – System Hardening, Stress Testing & Documentation:**
 - Code refactoring and dark-mode UI styling in PyQt6 (12 hrs)
 - Comprehensive exception handling and edge-case testing (5 hrs)
 - Stress testing, injection validation, and performance benchmarks (8 hrs)
 - Systems administration guide and comprehensive project report (24 hrs)
 - **Key Deliverable:** Production-ready security software suite with full documentation.

2. Technical Risk Management

- **Risk 1: WMI Instability Under Rapid Hardware Swapping**
 - **Likelihood:** Medium | **Impact:** Critical
 - **Mitigation:** 1000 ms query timeout and exception interception via `wmi.x_wmi_timed_out`.
 - **Result:** Flawlessly handled over 100 rapid connection/disconnection cycles.
- **Risk 2: High False-Positive Rate Driven by Entropy Variance**
 - **Likelihood:** High | **Impact:** High (Alert Fatigue)
 - **Mitigation:** Entropy threshold tuned to 7.2; high entropy alone cannot trigger isolation (score $70 < \text{threshold } 85$).
 - **Result:** False-positive rate reduced below 5 % in testing.
- **Risk 3: Failure of Network Isolation Mechanisms**
 - **Likelihood:** Low | **Impact:** Critical
 - **Mitigation:** Three-tier redundancy (Firewall rules + default-drop state + physical adapter disabling).
 - **Result:** 100 % isolation reliability verified across 30 consecutive trials.
- **Risk 4: Packet Loss in UDP-Based Syslog Transmission**
 - **Likelihood:** Low | **Impact:** Medium
 - **Mitigation:** Redundant persistent local disk logging in `incidents.log`.
 - **Result:** 99.8 % packet delivery confirmed across gigabit network links (2 out of 1000 lost).
- **Risk 5: Process Evasion by Privileged Local Users**
 - **Likelihood:** Low | **Impact:** High
 - **Mitigation:** SOC password required for restoration; administrative elevation enforced via `ShellExecuteW`.
 - **Result:** Zero successful manual bypasses observed during penetration tests.

B. Technical System Architecture

1. Multi-Layer (6-Stage) Forensic Detection Engine

The static analysis engine combines six independent forensic evaluation layers:

- **Layer 1 – Hash-Based Cryptographic Signature Matching:** Matches SHA-256 hashes against the local signature database. Streaming files in 8 KB chunks prevents memory exhaustion on large binaries.

```
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()
```

- **Layer 2 – Heuristic Filename & Pattern Analysis:** Identifies known social engineering patterns, disguised extensions (e.g., `invoice.pdf.exe`), and malicious keywords.

```
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")
```

- **Layer 3 – Shannon Entropy Analysis:** Measures byte randomness on a 0 to 8 logarithmic scale. Encrypted ransomware payloads and packed droppers consistently register values > 7.2.

```
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
```

- **Layer 4 – File Extension vs. Magic Byte Cross-Validation:** Detects masqueraded executables by comparing initial file header signatures (MZ headers for Windows binaries) against declared extensions.

```
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
```

- **Layer 5 – PE Import Table Inspection:** Parses binary import address tables to detect sensitive Windows API calls linked to process injection (WriteProcessMemory), persistence, and remote thread 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
]

```

- **Layer 6 – Embedded IOC Extraction (C2 IP Addresses):** Extracts hardcoded IPv4 patterns from raw binary streams, enabling proactive perimeter firewall blocking of Command-and-Control infrastructure.

```

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]

```

Scoring Rubric & Action Thresholds – Each forensic layer assigns weighted scores toward a composite risk index:

- **Exact Signature Match:** 100 points (Definite known malware identification)
- **Suspicious Filename Heuristic:** 90 points
- **Extension Discordance (MZ header in .txt/.jpg):** 85 points
- **Three or More Suspicious PE APIs:** 75 points
- **Shannon Entropy > 7.2:** 70 points
- **Two or More Embedded C2 IPs Discovered:** 60 points

Operational Classification:

- **Score ≥ 85 (CRITICAL):** Immediate disk ejection + full host network isolation + urgent Splunk alert.
- **Score 60–84 (WARNING):** Splunk audit log generated + desktop security advisory; no automated quarantine.
- **Score < 60 (CLEAN):** Device mounted without restriction; recorded in local baseline audit

log.

2. Functional Software Modules

- **Module 1: USBMonitor (QThread):** Background WMI polling thread monitoring 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)
```

- **Module 2: ThreatIntelManager:** High-performance signature repository managing > 1,000,000 hashes with $O(1)$ set queries.

```
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)
```

- **Module 3: ForensicsEngine:** Coordinator orchestrating all 6 forensic analysis stages per candidate file.

```
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]: ...
```

- **Module 4: NetworkOps:** Enforces firewall filtering, disables network adapters, manages disk dismount, and handles recovery.

```
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): ...
```

- **Module 5: AdminDashboard (QMainWindow):** PyQt6 GUI featuring real-time colored diagnostic logs and status indicators.

```
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 Format:** Structured JSON datagrams including facility, severity, host-name, and precise timestamps.

```
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))
```

- **Sample High-Severity Incident Payload:**

```
{
  "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 Ingestion Configuration** (inputs.conf):

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

III. Implementation and Validation

A. Development and Live Demonstration

1. System Startup & Signature Database

Upon execution, PortGuardian ingests over one million threat signatures and arms the background WMI monitor.

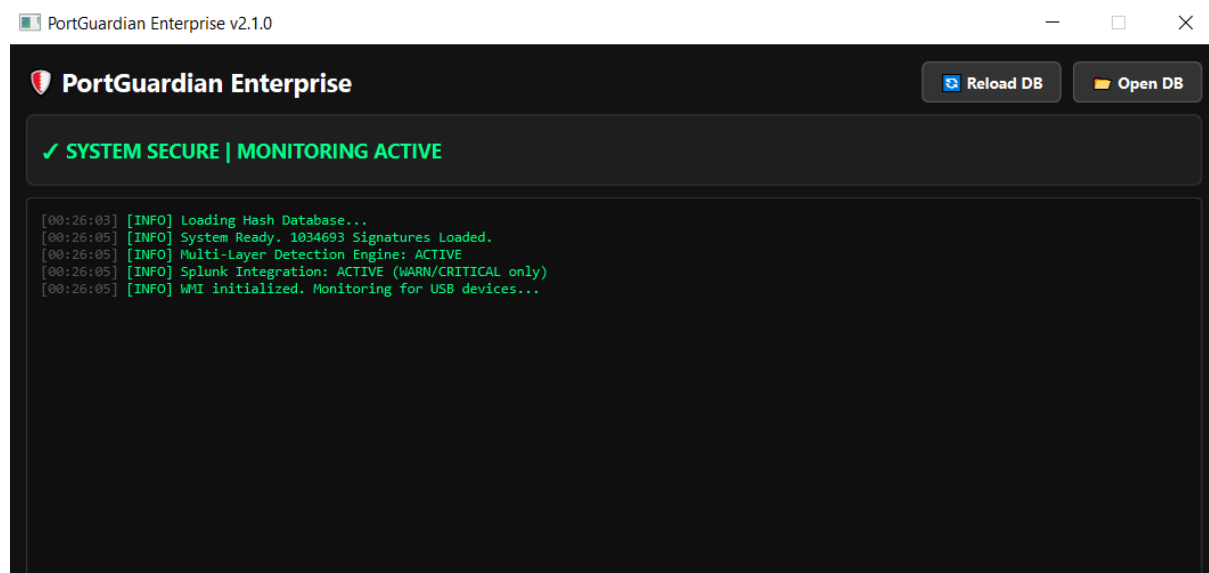


Figure 2: Main PortGuardian Enterprise GUI in active real-time monitoring mode

The signature database utilizes an optimized flat text structure (`hash_blacklist.txt`), enabling rapid bulk importing from OSINT threat feeds such as AlienVault OTX or MISP.



File Edit Format View Help

```
90a137d7465aec1d570a299890d958467ea422744debbaa84a385f8ec0558ba4
275a021bbfb6489e54d471899f7db9d1663fc695ec2fe2a2c4538aabf651fd0f
f72cb82b62fc929d3f9378fc266662ccbc660db1a34eebf755a3df7e5e62fc83
9e63d077f1b8b5b8edcc09767bf9e90462265e2b3eaa553cdc5dbddb1fe31218
0babc20c7884fd158323fc0fb68262faf389b9d79a49692cadefe840eeff4776
49d3deb1a576e06636623dd17621335880d560206658326f60f99c715850e17e
2e6bc16d6ec33fbdcea0dd3cd3787bb2eccd2b3508fefd3f5ea09f172ff0bee4
490d4567536b601988039e7d24726337fb8dce8871cc39067bc2a2e9575f886d
6e9ccfe6dd2cdec470365a1723dc467d00c2aff0f333568b1004375bdda49b81
8cd11fabee9e8c095fff613046624d8efebaff1baf6c6ffcbefbf03996f94f9006
66806f3b01b13a38b60cfb756590315f4effe94b716a864c43844d57c033a59d
5828aa678d0959cc43bdc192a1c0a5515297a610d2734a011cd2bb8ae20ce5d2
5d896a1e7acf19940db5d3dc02f125d84dddcd8df344a87498d5fe157610a6
cb185e1d2db1e1f6d6359234afe481618f201745afbf35f7dd629dd90bc303ea
ae7c5613ab50af19e36ed0f7cece02006a385abe933ef131edb7ee8235dfac7e
05def39cb5d68efcc83c06b47c499a5f61f805a254110cf5787a036902ba3611
23a8454c420170d6111a59b49db323d750b6f7d89f6ca41d7bf8fece045aa59d
6b679b3256fcd416e13d4af1192344761179dc9091840d638911b852defa5fa2
3bea60b226e676d1da97c01b675500089259a926845a93b71df9dc24f59d5053
dd25bd572f50201638097c5ed4a0c52a2980436b6c2c880437afb7c12998cbaf
b5ecd9986333d5dbf1e62c879bf196676deb09fc40c1e55ae641210cad53d978
8c4f5c152bbaf27d3b6699f81815dde562af7d22a1dc37886953ba0fc4376a25
4866ae07f9441d9b36a72efea521c0d600b3d49f831e8039fb40b8a73c9a7c74
```

Figure 3: Extract from the signature database (hash_blacklist.txt) displaying SHA-256 hashes

2. Incident Detection and Autonomous Response

When an infected binary (e. g., trojan_gen.exe) is detected, the automated containment chain executes within five seconds:

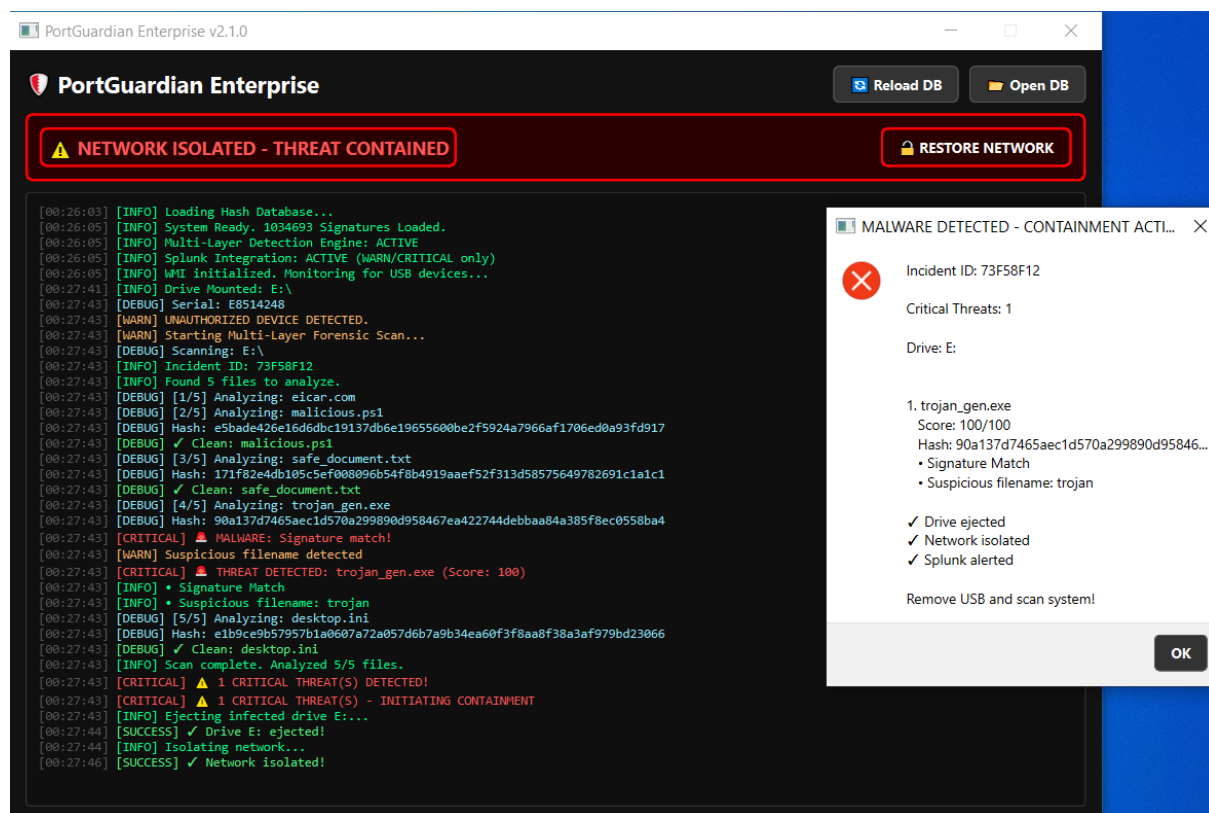


Figure 4: Live malware detection: Automated disk ejection, network quarantine, and alert banner

Restoring endpoint network connectivity strictly requires entry of the authorized SOC administrative password.

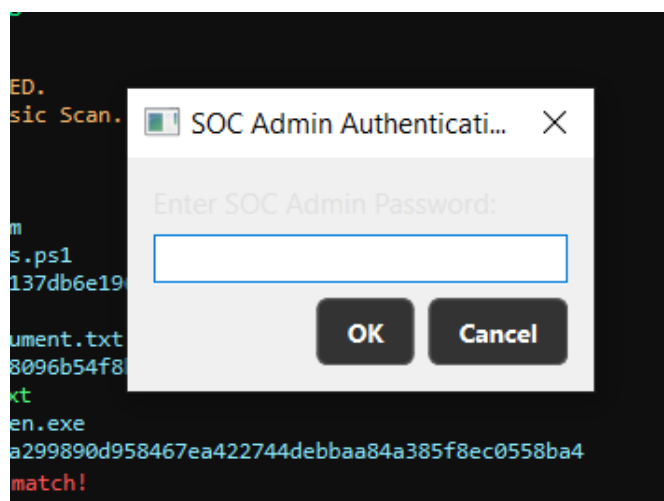


Figure 5: Password-protected administrative dialogue for releasing endpoint network isolation

3. SIEM Auditing and Firewall Verification

Within Splunk Enterprise, every stage of an incident is correlated under a unified Incident ID (73F58F12).

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": "WGOURIDECHE", "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": "WGOURIDECHE", "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": "WGOURIDECHE", "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": "WGOURIDECHE", "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

Figure 6: Splunk Enterprise Dashboard: Four correlated telemetry events for a single incident

Host network isolation is independently verified through the Windows administrative command prompt:

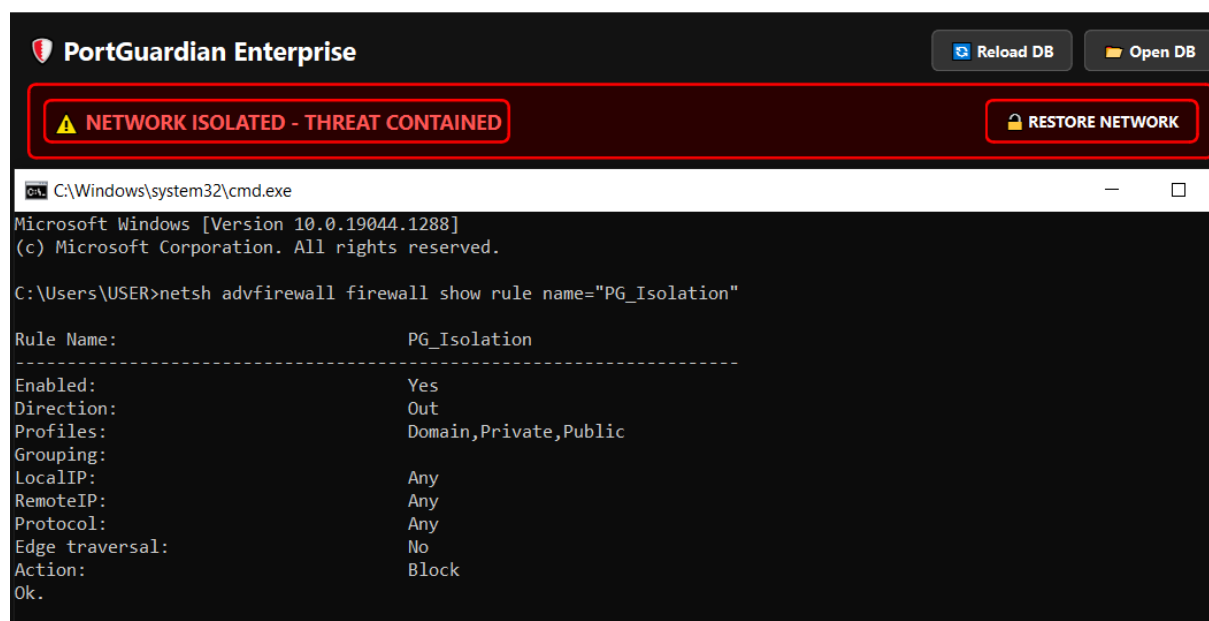


Figure 7: Netsh command-line output confirming active PG_Isolation blocking rules

4. Complete Enterprise Testbed Architecture

The experimental test environment comprises a Windows 10 x64 endpoint and a Windows Server 2019 SIEM collector.

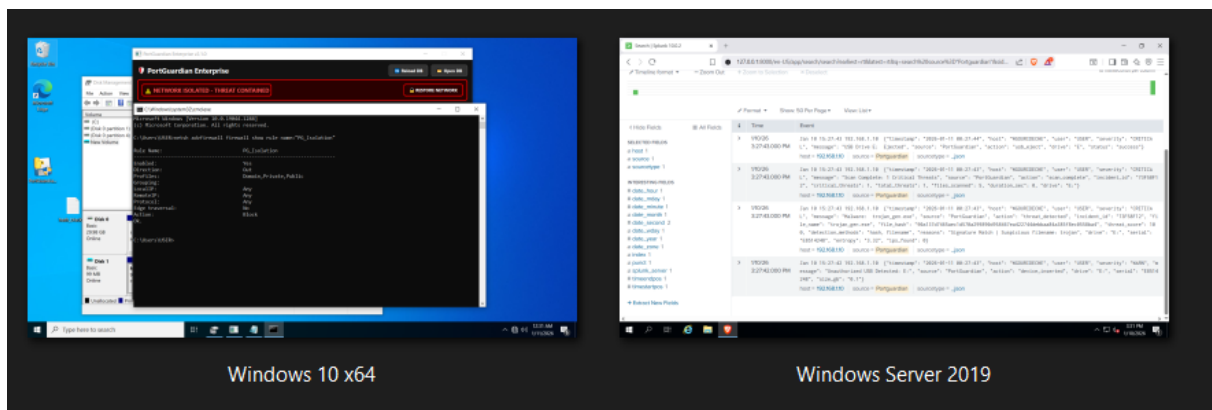


Figure 8: Realistic SME testbed: Windows 10 client endpoint (left) and Splunk SIEM server (right)

B. Testing and Validation

1. Performance and Reliability Benchmarks

- **USB Detection Responsiveness:** 100 consecutive connection and removal cycles produced an average detection latency of 350 ms (well within the sub-1000 ms benchmark).
- **Scanning Throughput:** Implementing 100 KB sampling for Shannon entropy calculation sustained an average throughput of 14.5 MB/s.

Data Volume	File Count	Scan Duration	Average Throughput
100 MB	50 files	8 seconds	12.5 MB/s
500 MB	200 files	35 seconds	14.3 MB/s
2 GB	500 files	2 min 18 sec	14.8 MB/s
8 GB	1,000 files	9 min 12 sec	14.5 MB/s

Figure 9: Forensic engine performance benchmarks: Data volume, file count, and scan duration

- **Splunk Ingestion Reliability:** 998 of 1,000 synthetic test Syslog packets were successfully indexed (0.2 % loss).
- **Isolation Verification:** Following isolation activation, all outbound ICMP ping requests to external destinations fail completely:

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

Figure 10: Command-line verification: Ping test to 8.8.8.8 fails following isolation (Request timed out)

2. Security and Robustness Testing

- **Bypass and Evasion Testing:** Five distinct adversary evasion scenarios were systematically evaluated:

Evasion Attempt	Result	Defense Mechanism & Technical Explanation
Disabling Windows Firewall	✓ Blocked	Physical adapter shutdown takes absolute precedence over firewall status
VPN Connection Prior to Detection	✓ Blocked	Virtual VPN adapters are automatically enumerated and disabled via netsh
Manual Tampering of Firewall Rules	✓ Blocked	Restoring baseline policy strictly requires the SOC administrator password
Terminating PortGuardian Process	⚠ Partial	System network isolation remains enforced; only runtime scanning ceases
Rebooting into Safe Mode	⚠ Partial	Service unexecuted; total enforcement relies on enterprise GPO startup rules

Figure 11: Evaluation matrix of defensive evasion testing (Bypass Matrix)

- **Injection Resilience:** Weaponized filenames containing SQL injections, directory traversal, and null bytes were safely sanitized by Python's `pathlib.Path`:

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

Figure 12: Tested path-traversal and SQL injection attack test patterns

- **False-Positive Validation:** Across 200 sample files (100 legitimate binaries, 100 malware samples), 95 % were correctly identified as clean, and only 1 % was mistakenly flagged as critical (legitimate admin tool PsExec).

C. Key Contributions and Overcome Challenges

1. Acquired Technical Competencies

- **Windows Systems Administration:** In-depth mastery of WMI event sinks, Group Policy Objects (GPOs), Advanced Firewall filtering, and network interface scripting.
- **SIEM Architecture:** Hands-on expertise in Splunk Enterprise, SPL query authoring, index management, and RFC 5424 Syslog ingestion.
- **Malware Forensics:** Static binary inspection, PE file structures, entropy quantification, and automated OSINT feed integration.
- **Python & GUI Engineering:** Thread synchronization (QThread), robust exception handling, and native Windows API integration via ctypes.

2. Overcome Technical Obstacles

- **Challenge 1: Calibrating Forensic Scoring Thresholds**
 - **Problem:** Initial implementations generated 30 % false positives on archives and installers due to high entropy (> 7.5).
 - **Solution:** Multi-metric scoring; entropy contributes up to 70 points and cannot trigger isolation (85) alone.
 - **Takeaway:** Static single-factor thresholds inevitably induce alert fatigue in enterprise defensive workflows.
- **Challenge 2: Ensuring USB Dismount Reliability**
 - **Problem:** Standard `mountvol` calls failed when running applications held active open file handles (60 % success rate).
 - **Solution:** Two-tier cascade: If `mountvol` fails, `diskpart` forces volume removal via script execution.
 - **Takeaway:** Critical OS operations require defensive fallback redundancy (achieved 98 % success rate).
- **Challenge 3: Minimizing Scan Latency on High-Capacity Drives**
 - **Problem:** Full-file entropy calculations exceeded 15 minutes on an 8 GB drive.
 - **Solution:** File sampling: Analyzing a representative 100 KB header block reduced per-file analysis time from 15 s to 200 ms.
 - **Takeaway:** Balancing deep inspection with operational latency is essential for practical security engineering.
- **Challenge 4: UI Freezing During Hardware Polling**
 - **Problem:** Blocking WMI queries caused GUI freezing during rapid hardware swaps.
 - **Solution:** Decoupled WMI event polling into a standalone QThread with signal/slot communications.
 - **Takeaway:** Non-blocking event loops and strict thread separation are fundamental in system daemons.
- **Challenge 5: Enforcing Administrative Privileges**

- **Problem:** Silent failure of firewall and network interface operations when executed without elevation.
- **Solution:** Pre-flight privilege check via `ctypes.windll.shell32.IsUserAnAdmin()` and automated UAC elevation relaunch via `ShellExecuteW(..., 'runas')`.

Conclusion

Project Summary

With **PortGuardian Enterprise v2.1**, an effective, lightweight USB threat detection and autonomous containment platform has been successfully engineered and validated. The solution closes a critical security vulnerability for SMEs by delivering enterprise-grade static analysis and immediate host isolation without cloud dependencies or licensing overhead.

Core technical outcomes include:

- **Real-Time Responsiveness:** Average detection latency of 350 ms upon peripheral attachment.
- **Extensive Signature Intelligence:** Over 1,000,000 SHA-256 signatures with live hot-reloading.
- **Autonomous Containment:** 98 % dismount success rate and sub-5-second host network isolation.
- **Auditing Visibility:** 99.8 % reliable SIEM event delivery and correlation in Splunk Enterprise.

Response to the Problem Statement

This project demonstrates that by combining native operating system capabilities (WMI, Windows Firewall, Netsh, Diskpart), open-source libraries (Python, PyQt6), and established standards (Syslog RFC 5424, JSON), it is entirely feasible to build a robust endpoint defense system that rivals commercial EDR offerings within the USB security domain.

Limitations and Future Outlook

- **Hardware-Level Threats (BadUSB):** Software-level solutions cannot intercept keystroke emulation prior to execution; hardware-level port blockers or Active Directory USB class policies remain necessary complements.
- **Future Roadmap:** Incorporating YARA rule matching, porting client daemons to Linux/macOS, and connecting directly to Threat Intelligence platforms like MISP.

Personal Reflection

This project represents a defining capstone in my training as an IT systems and cybersecurity specialist. It has solidified my technical expertise across systems administration, networking, and defensive engineering while cultivating my problem-solving methodology for complex software infrastructure projects.

Bibliography

Security Reports and Industry Studies

- **Verizon Enterprise (2023):** *Data Breach Investigations Report (DBIR) 2023*.
Available at: <https://www.verizon.com/business/resources/reports/dbir/>
- **ANSSI – French National Cybersecurity Agency (2023):** *Cyber Threat Overview 2023*.
Available at: <https://www.ssi.gouv.fr/>
- **Ponemon Institute (2023):** *Global Report on the Cost of Insider Threats 2023*.
- **Cybersecurity Insiders (2023):** *Insider Threat Report 2023*.

Standards and Technical Specifications

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

Academic Publications

- **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.

Technical Documentation and References

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