

# Sicherheit von Betriebssystemen – VO 05: Windows Security

Thomas Stipsits

Research Group for Industrial Software (INSO)

<https://www.inso-world.com>



**RISE**   
Research Industrial Systems Engineering



**FACHHOCHSCHULE  
WIENER NEUSTADT**  
University of Applied Sciences - Austria





# Agenda

- Encryption & Data Protection
- Boot Security
- Logon & Credential Security
- System Security
- Windows Registry
- Windows Services
- Windows DLL Handling
- AppLocker
- Virus & Threat Protection
- Ausblick: Agentic Security
- Zusammenfassung

# BitLocker Encryption & Data Protection

- Bitlocker als Lösung für **Full Disk Encryption (FDE)** seit Windows Vista im Einsatz
- Offizieller Nachfolger von SysKey-Verschlüsselung des SAM
- Eigene Systempartition, die im Bootprozess (UEFI) dazu dient Bitlocker zu initialisieren und verschlüsselte Daten zu laden
- **Problematik:** Root-Of-Trust
- Schlüsselversionen für BitLocker:
  1. Pre-Windows 8: Trusted Platform Module (TPM) oder USB-Key
  2. Windows 8: TPM, USB-Key oder Volume Passwort
  3. Windows 8.1: „Windows Device Encryption“ als „FDE“ für Home Editions
  4. Windows 11 24H2: „Windows Device Encryption“ als default aktiv!

# Windows Device Encryption

- Windows Device Encryption ist eine vereinfachte Version der Bitlocker FDE für Windows Home Editionen
- Wird seit Windows 11 24H2 bei Installation mit Microsoft Konten standardmäßig aktiviert (bei lokalen Accounts noch inaktiv) wenn Vorbedingungen (z.B. TPM 2.0 vorhanden) erfüllt sind
- Keys werden in TPM ohne PIN-Abfrage gespeichert
- Recovery Keys werden online mit Microsoft Account, EntraID oder AD verknüpft
- Alle Partitionen und Festplatte, welche beim Windows Setup verbunden sind werden mit Device Encryption verschlüsselt
- **Problematik:** Kein Schutz bei Verlust des Rechners (Laptop-Diebstahl etc.)

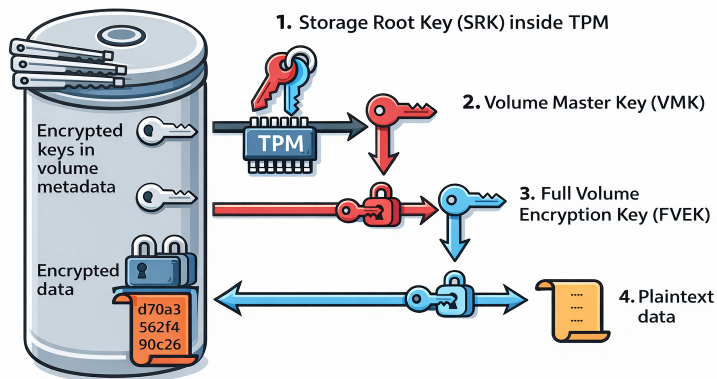
# Trusted Platform Modules (TPMs)

- Eigenständiger Chip auf Mainboard der u.a. als „Schlüsselspeicher“ genutzt wird
- Enthält üblicherweise folgende Komponenten:
  1. **Krypto-Prozessor** — Zufallszahlen- und Schlüsselgeneratoren
  2. **Flüchtige Speicher** — Insb. für Erstellung sogenannter Attestations (Bestätigungen zu Fähigkeiten oder Zustand des TPMs)
  3. **Nicht flüchtige Speicher** — u.a. zur Speicherung des Storage Root Key (SRK), mit welchem alle durch das TPM zu schützende Schlüssel verschlüsselt werden.
- Als Alternative zu einem eigenen TPM Chip kann in Windows 11 auch Microsoft Pluton verwendet werden

# Windows Remote Device Actions

- Wird das Gerät mit dem Windows Intune Cloud Dienst verbunden (Cloud Device Management) bietet Microsoft mehrere „Remote Actions“ an
- Kommen insbesondere bei Diebstahl oder Verlust, im Fehlerfall, oder auch im Fall eines Sicherheitsvorfalles zu tragen
- Für Wirksamkeit muss das Gerät in Intune registriert und mit dem Internet verbunden sein
- Es werden etliche Befehle angeboten, beispielsweise:
  - Bitlocker Schlüsselrotation
  - Gerätewipe, Löschen von Unternehmens-/Nutzerdaten etc.
  - Restart
  - Gerätesuche & Standort anzeigen
  - Durchführung diverser Scans von Windows Defender

# Bitlocker Entschlüsselungsprozess

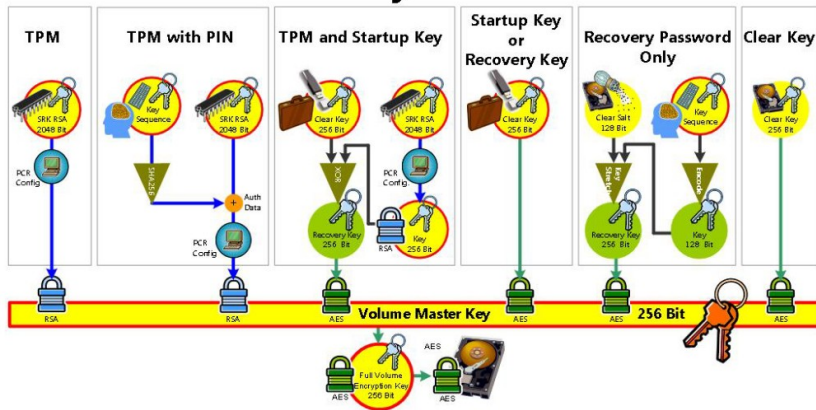


(Vergleiche

<https://blog.elcomsoft.com/2021/01/understanding-bitlocker-tpm-protection/>)

# BitLocker Schlüsselarchitektur

## BitLocker Key Architecture



(Vergleiche <https://csrc.nist.gov/csrc/media/projects/cryptographic-module-validation-program/>

[//csrc.nist.gov/csrc/media/projects/cryptographic-module-validation-program/](https://csrc.nist.gov/csrc/media/projects/cryptographic-module-validation-program/)

# UEFI & Windows Boot Security



(Vergleiche [https://de.m.wikipedia.org/wiki/Datei:Uefi\\_logo.svg](https://de.m.wikipedia.org/wiki/Datei:Uefi_logo.svg))

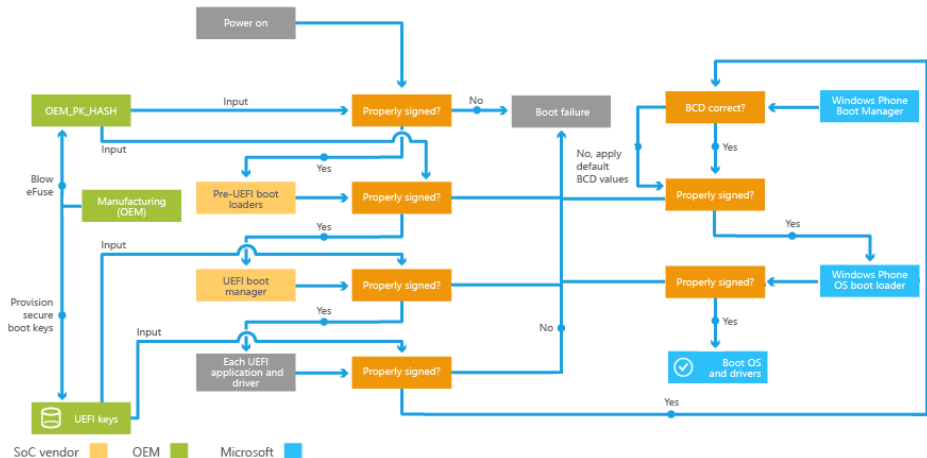
# Windows Bootprozess

- Boot-Prozess früher MBR (legacy) Boot – viele Freiheiten für Hersteller
- 2005 (UEFI v1) bzw. 2006 (UEFI v2) – Standardisierung des Bootprozesses
- Windows 8 adaptierte UEFI mit Secure Boot Mechanismus
- Secure Boot als mitigierende Maßnahme für Bootkits
- **Bootkits** – Infizieren BootLoader oder andere Boot relevante Komponenten um vor OS geladen zu werden
- Bootkits sind typischerweise Kernel-Modus Rootkits (Unbeschränkte Zugriffsrechte)

# Secure Boot

- Grundidee: Sicherstellen der **Integrität der Komponente** vor Ausführung durch Signaturen
- Windows 8 adaptierte UEFI mit Secure Boot Mechanismus
- Secure Boot als mitigierende Maßnahme für Bootkits
- Nur effektiv wenn Trustchain **vollständig** geprüft wird
- Für vollständige Trustchain auf allen Bootlevel:
  1. **Plattform Secure Boot** – Hardwareseitig, Verifiziert Initialisierung & Pre-Boot
  2. **UEFI Secure Boot** – Verifiziert UEFI Applikationen & Treiber sowie OS Bootloader
  3. **OS Secure Boot** – OS-Spezifischer Bootvorgang, Verifiziert OS Kernel etc.

# Windows Bootprozess Overview



(Vergleiche <https://docs.microsoft.com/de-de/windows-hardware/drivers/bringup/secure-boot-and-device-encryption-overview>)

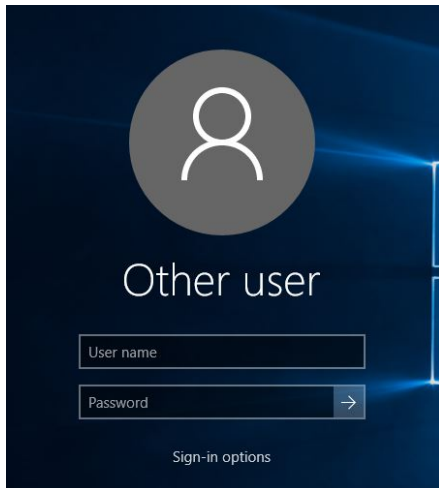
# Windows Bootprozess – Platform & UEFI Boot

1. SHA-256 Wert des **Root-Certs** wird mit **OEM\_PK\_Hash** verglichen  
(Herstellerseitig hardgecoded)
2. **Signatur von Pre-UEFI Boot Loader** gegen Root Zertifikat **validieren** & laden
3. **Signatur von UEFI Boot Manager** gegen UEFI-Keys **validieren** & laden
4. **UEFI Boot Manager lädt Apps und Treiber** und prüft die Signaturen gegen UEFI-Keys
5. **OS-Hersteller-spezifischer Boot Manager** wird geladen

# Windows Bootprozess – Windows Boot

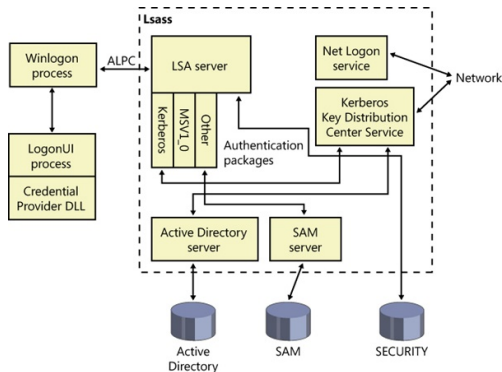
1. **Boot-Manager prüft die Boot Configuration Data (BCD)**. Falls BCD nicht intakt – Default-Werte aus Secure Boot Policy
2. Falls FDE (Bitlocker) aktiv – **Systempartition von Boot-Manager decrypted**
3. **Windows-Boot Loader wird** anschließend durch den Boot Manager **hinsichtlich der Signatur geprüft & geladen**.
4. **Signatur der** zum Boot benötigten **Treiber und des Kernels werden geprüft & geladen**
5. **Integritätscheck des Kernels & wichtiger Windows Komponenten**
6. Boot-Prozess wird abgeschlossen & **Trust-Chain wird durch Kernel fortgeführt**

# Windows Credentials & Logon Security



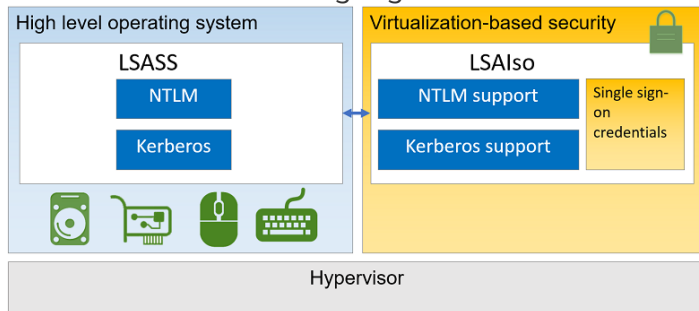
# Windows Logon Process — Pre-Credential Guard

- Credentials werden bei Login abgefragt und von Credentials Provider DLLs gehasht zu LSASS weitergegeben
- LSASS prüft Kontorechte, holt Credential-Hash von SAM und vergleicht Hashes



# Windows Logon Process — Windows Credential Guard

- Verhindert Auslesen von Hashes aus dem LSASS Speicher
- Deckt Kerberos TGTs und NTLM Authentifizierung ab, ABER NICHT lokale Accounts und Microsoft Accounts
- Hashes werden in geschützten isolierten Prozess „LsaIso.exe“ verschlüsselt gespeichert, nur durch ALPC Calls zugänglich



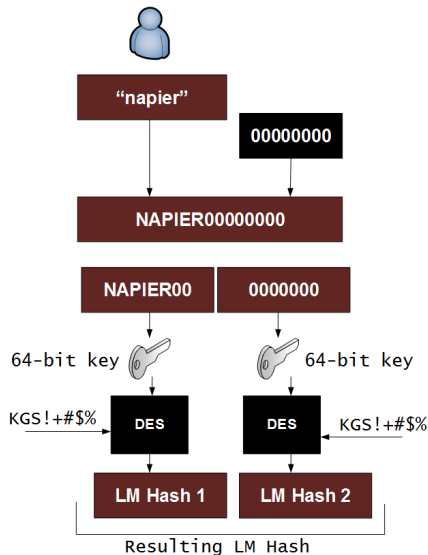
# Windows Credentials – Hashformate

- Security Accounts Manager speichert gehashte Credentials (LM/NTLM)
- SAM Einträge wurden durch „SysKey“ verschlüsselt (>Win 2000 & <Win 10)
- Der Syskey wird gesplitted und in (durch regedit) unlesbaren Einträgen im „SYSTEM“ Hive versteckt
- Im „SECURITY“ Hive befinden sich die erste Hälfte von Domänencredentials (gecached)
- SysKey nicht mehr als Feature in Windows 11 enthalten, stattdessen BitLocker um gesamtes Filesystem zu verschlüsseln
- Credentials werden sowohl in LM als auch NTLM Format gespeichert
- **LM** kryptographisch sehr schwach, seit Win Vista deaktiviert
- **NTLM** aktuell in Verwendung, kein Salt

# LAN Manager Hash (LM)

- Passwort wird auf 14 Zeichen gecropped/gepadded
- 14-Zeichen String wird in 2x 7-Zeichen strings gesplittet
- 7-Zeichen Strings werden zur Generierung von 64-bit DES Keys verwendet
- *KGS!+#\$%* wird mit beiden Keys mit DES-ECB verschlüsselt
- Die beiden 8-Byte Ergebnis-Strings werden konkateniert und bilden den LM Hash

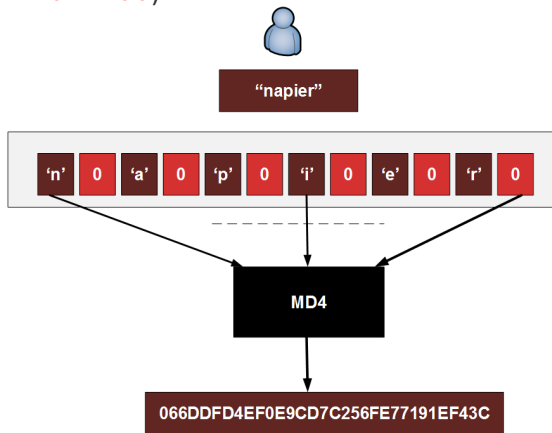
# LAN Manager Hash (LM) Visualisierung



(Vergleiche <https://asecuritysite.com/encryption/lmhash>)

# NT LAN Manager Hash (NTLM)

- Passwort wird in unicode convertiert ( $A = 0x0041$ )
- Chars werden im *Little Endian* Format angefügt und MD4 (**ohne SALT!**) gebildet ( $A = 0x4100$ )



(Vergleiche <https://asecuritysite.com/encryption/lmhash>)

# Windows Registry – Einführung

- Systemdatenbank die für unterschiedliche Zwecke genutzt wird
- Einsehbar mit `C:\WINDOWS\system32\regedit.exe`
- Enthält u.A. Konfigurationsdaten, Sicherheitsdatenbank, systemweite Softwareeinstellungen ...
- Organisiert in sogenannten „*Hives, Keys* und *Values*“
  - *Values* = Wertetupel (Name, Type & Wert)
  - (*Sub-*)*Keys* = Ordner, die die Struktur der Registry bilden
  - *Hives* = „Root-Keys“ die Teile des Systems repräsentieren
- z.B.: AutoRun RegKey:  
`HKCU\Software\Microsoft\Windows\CurrentVersion\Run`

# Windows Registry – The dark side

- Oftmals auch von Malware missbraucht (z.B. AutoRun Registry Keys)
- Zugriff auf Registry Keys von installierten Services ist ebenfalls kritisch (Service Registry Permission)
- Modifizieren von Values kann zu unerwünschten Nebeneffekten führen
- siehe z.B. Steam Privilege Escalation (07.08.2019)
  - Steam Client Service setzte Permissions aller Subkeys auf „Full Read Write access“
  - Ein Link auf einen anderen Value setzt die Permissions dieses Keys ebenfalls
  - Beliebiger Value konnte so writeable gemacht werden

(Vergleiche <https://amonitoring.ru/article/steamclient-0day/>)

# Windows Services und ihre Eigenschaften

- Windows Services sind eine Kernkomponente des OS für **langlaufende Prozesse**
- Services können **ohne User-Interaktion** starten und können auch nach logout laufen
- Besitzen eigene **Windows-Session**, können bei Boot automatisch starten
- Sind für den User „**unsichtbar**“, erledigen aber notwendige Tätigkeiten
- z.B.: Spooler(Druckerwarteschlange), ...

# Windows Services und ihre Konfiguration

- Services können mittels `sc qc *SERVICENAME*` abgefragt werden:

```
SERVICE_NAME: WCAssistantService
TYPE          : 10  WIN32_OWN_PROCESS
START_TYPE    : 2   AUTO_START
ERROR_CONTROL : 1   NORMAL
BINARY_PATH_NAME : C:\Program Files (x86)\Lavasoft\Web Companion\Application\
                Lavasoft.WCAssistant.WinService.exe
LOAD_ORDER_GROUP :
TAG           : 0
DISPLAY_NAME  : WC Assistant
DEPENDENCIES  :
SERVICE_START_NAME : LocalSystem
^^I
```

# Windows Services – Unquoted Service Paths

- Binary-Paths, die nicht mit Quotes umschlossen sind könnten angreifbar sein
- Aus Beispiel : *C:\Program Files (x86)\Lavasoft\Web Companion\Application\Lavasoft.WCAssistant.WinService.exe*
- Ersetzen/Erstellen von exe Files kann zu Privilege Escalation führen.
- Folgende .exe Files würden ausgeführt werden:
  1. C:\Program.exe
  2. C:\Program Files (x86)\Lavasoft\Web.exe
  3. C:\Program Files (x86)\Lavasoft\Web Companion\Application\Lavasoft.WCAssistant.WinService.exe

# Windows Services – Insecure Service Permissions

- Sind die Service Permissions falsch gesetzt kann dies ebenfalls zu Privilege Escalation bzw. Persistence führen
- z.B.: Binary-Path des Services ist für User writeable
- Angreifer kann Pfad durch arbiträren Pfad ersetzen und Neustart des Services erzwingen
- Überprüfung z.B. mittels *accesschk*:
  1. `accesschk.exe -uwcqv „Authenticated Users“ *`
  2. `accesschk.exe -ucqv upnphost`

(Vergleiche <https://docs.microsoft.com/de-de/sysinternals/downloads/accesschk>)

# Windows Dynamic Link Libraries (DLLs)

- Binärdatei, welche aufrufbare Subroutinen (Funktionen) kapselt
- Anwendungsübergreifend aufrufbar
- Einsparungen an Speicher auf Festplatte & Arbeitsspeicher
- DLL Code wird 1 mal in Arbeitsspeicher geladen & von mehreren Anwendungen aufgerufen
- DLLs können von Jedermann erstellt & kompiliert werden
- Wesentliche Betriebssystemfunktionen werden über DLLs aufrufbar gemacht
- z.B. Kernel32.dll – Kernel API, User32.dll – GUI related Services
- DLL-Aufrufe von .exe Dateien können durch *Procmon* angesehen werden.

(Vergleiche <https://docs.microsoft.com/de-de/sysinternals/downloads/procmon>)

# Procmon DLL View

Time ...	Process Name	PID	Operation	Path	Result
1:34:1...	Bginfo.exe	2012	CreateFile	C:\Users\visus\Desktop\bginfo\VERSION.dll	NAME NOT FOUND
1:34:1...	Bginfo.exe	2012	CreateFile	C:\Users\visus\Desktop\bginfo\snmpapi.dll	NAME NOT FOUND
1:34:1...	Bginfo.exe	2012	CreateFile	C:\Users\visus\Desktop\bginfo\NETAPI32.dll	NAME NOT FOUND
1:34:1...	Bginfo.exe	2012	CreateFile	C:\Users\visus\Desktop\bginfo\netutils.dll	NAME NOT FOUND
1:34:1...	Bginfo.exe	2012	CreateFile	C:\Users\visus\Desktop\bginfo\svrcli.dll	NAME NOT FOUND
1:34:1...	Bginfo.exe	2012	CreateFile	C:\Users\visus\Desktop\bginfo\wkscli.dll	NAME NOT FOUND
1:34:1...	Bginfo.exe	2012	CreateFile	C:\Users\visus\Desktop\bginfo\ODBC32.dll	NAME NOT FOUND
1:34:1...	Bginfo.exe	2012	CreateFile	C:\Users\visus\Desktop\bginfo\MSIMG32.dll	NAME NOT FOUND
1:34:1...	Bginfo.exe	2012	CreateFile	C:\Users\visus\Desktop\bginfo\LxTheme.dll	NAME NOT FOUND
1:34:1...	Bginfo.exe	2012	CreateFile	C:\Users\visus\Desktop\bginfo\oledlg.dll	NAME NOT FOUND
1:34:1...	Bginfo.exe	2012	CreateFile	C:\Users\visus\Desktop\bginfo\OLEACC.dll	NAME NOT FOUND
1:34:1...	Bginfo.exe	2012	CreateFile	C:\Users\visus\Desktop\bginfo\WINMM.dll	NAME NOT FOUND
1:34:1...	Bginfo.exe	2012	CreateFile	C:\Users\visus\Desktop\bginfo\OLEACCR.CLL	NAME NOT FOUND
1:34:1...	Bginfo.exe	2012	CreateFile	C:\Users\visus\Desktop\bginfo\CRYPTBASE.dll	NAME NOT FOUND
1:34:1...	Bginfo.exe	2012	CreateFile	C:\Windows\System32\wbem\wbemcomn.dll	NAME NOT FOUND
1:34:1...	Bginfo.exe	2012	CreateFile	C:\Users\visus\Desktop\bginfo\CRYPTSP.dll	NAME NOT FOUND
1:34:1...	Bginfo.exe	2012	CreateFile	C:\Users\visus\Desktop\bginfo\RpcRtRemote.dll	NAME NOT FOUND
1:34:1...	Bginfo.exe	2012	CreateFile	C:\Windows\System32\wbem\NTDSAPI.dll	NAME NOT FOUND
1:34:1...	Bginfo.exe	2012	CreateFile	C:\Users\visus\Desktop\bginfo\dwmapl.dll	NAME NOT FOUND
1:34:1...	Bginfo.exe	2012	CreateFile	C:\Users\visus\Desktop\bginfo\SspiCli.dll	NAME NOT FOUND
1:34:1...	Bginfo.exe	2012	CreateFile	C:\Users\visus\Desktop\bginfo\netmb1.dll	NAME NOT FOUND
1:34:1...	Bginfo.exe	2012	CreateFile	C:\Users\visus\Desktop\bginfo\IPHLPAPI.DLL	NAME NOT FOUND
1:34:1...	Bginfo.exe	2012	CreateFile	C:\Users\visus\Desktop\bginfo\WINNSI.DLL	NAME NOT FOUND
1:34:1...	Bginfo.exe	2012	CreateFile	C:\Users\visus\Desktop\bginfo\dhcpcsvc6.DLL	NAME NOT FOUND
1:34:1...	Bginfo.exe	2012	CreateFile	C:\Users\visus\Desktop\bginfo\dhcpcsvc.DLL	NAME NOT FOUND
1:34:1...	Bginfo.exe	2012	CreateFile	C:\Users\visus\Desktop\bginfo\WINSTA.dll	NAME NOT FOUND

# DLL Suchpfad

- Windows folgt spezifizierten Suchpfad bei Laden einer DLL:
  1. DLL Redirection Pfad
  2. API Sets
  3. SxS Manifest Redirection (Nur für Nicht-UWP Desktop Apps)
  4. Liste geladener Module
  5. Known DLLs
  6. Package Dependency Graph des Prozesses (definiert im App Manifest)
  7. Das Verzeichnis, von dem aus die .exe geladen wurde
  8. Das Systemverzeichnis **C:\Windows\System32**
  9. Das 16-bit Systemverzeichnis **C:\Windows\System**
  10. Das Windows Verzeichnis **C:\Windows**
  11. Das derzeitige Verzeichnis
  12. Verzeichnisse, welche in der PATH Variable gespeichert sind

# DLL Hijacking

## DLL HIJACKING (WINDOWS)

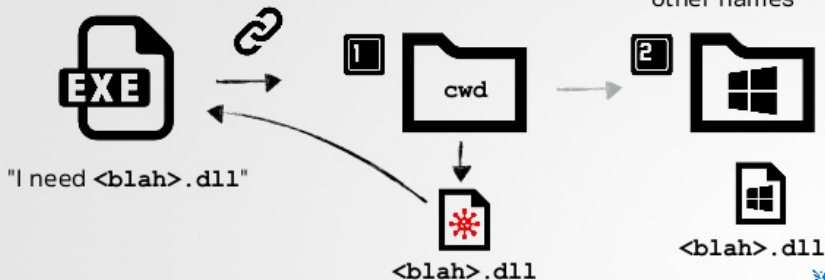
an overview

definition

"an attack that exploits the way some Windows applications **search and load** Dynamic Link Libraries (DLLs)"

"binary planting"  
"insecure library loading"  
"dll loading hijacking"  
"dll preloading attack"

other names



Synack

(Vergleiche <https://www.slideshare.net/Synack/can-secw>)

# DLL Hijacking Auswirkungen

- Angreifer kann Suchpfad hijacken und damit Code ausführen
- Code wird im Context der Anwendung ausgeführt
- Angreifer kann dadurch Zugriff auf andere (User-/System-)Accounts erhalten
- Kann beispielsweise zur Privilege Escalation, Persistence oder UAC Bypass genutzt werden
- **DLL Proxying** ist Abwandlung des Prinzips, Execution Pfad wird zu valide DLL weitergeleitet
- Ähnliche Ergebnisse können durch Ausnutzen von **DLL Injection** Schwachstellen erreicht werden

# Windows AppLocker

- Sicherheitsentscheidungen können nicht immer auf User Accounts basieren
- Windows benutzt AppIDs um Anwendungen über das Prozesszugriffstoken zu identifizieren
- AppLocker für Windows 7, 8 & 10 in Enterprise Version verfügbar
- Windows AppLocker ermöglicht die **Einschränkung der ausführbaren Programme mittels Regeln**
- Sowohl White- als auch Black-List Regeln umsetzbar
- Konfiguration mittels **Lokaler Sicherheitsrichtlinien (secpol.msc)**

# Windows AppLocker unterstützte Dateierarten

- Einschränkung von folgenden Dateierarten möglich:
  - **Ausführbare Binärdateien** – EXE, COM
  - **Dynamische programmbibliotheken** – DLL und OCX
  - **Microsoft (De-)Installer Dateien** – MSI & MSP
  - **Powershell** – PS1
  - **Batchdateien** – BAT & CMD
  - **VisualBasic** – VBS
  - **JavaScript** – JS

# Windows AppLocker Regeleset

- Regeln können hinsichtlich der folgenden Kriterien aufgestellt werden:
  1. **Zertifikat zur Codesignatur** – Anwendungen mit gültiger Hersteller Signatur (z.B. Google)
  2. **Verzeichnispfad** – Anwendungen die unter gegebenen Pfad liegen (z.B. C:\Tools)
  3. **Dateihash** – Hash der auszuführenden Datei (z.B. SHA256 eines Scripts)
- AppLocker-Regeln werden an verschiedenen Stellen in der **Windows Registry** gespeichert
- z.B.: `HKLM\Software\Policies\Microsoft\Windows\SrpV2`

# Windows AppLocker Beispielregel SSDN

- z.B. Whitelist Regel für Pidgin.exe (SID=Administrator Gruppe)

```
<FileHashRule Id="641b2233-5230-41a3-97ae-aec18d1d7374" Name="pidgin.exe"
  Description="Testrule" UserOrGroupSid="S
-1-5-21-1278764787-1901765877-3840102479-500" Action="Allow">
<Conditions>
  <FileHashCondition>
    <FileHash Type="SHA256" Data="0
xffdcdc467c5d9f433549f282e33d337fffaaba1db605c4de3b22c95ae6138dba"
SourceFileName="pidgin.exe" SourceFileLength="58232"/>
  </FileHashCondition>
</Conditions>
</FileHashRule>
```

# Windows AppLocker Bypasses

- Je nach Regelset kann AppLocker durch unterschiedliche Techniken umgangen werden.
- Beispieltechniken:
  - **DLL Hijacking** – Durch Wechsel des Zugriffstoken können SID basierte Blacklists umgangen werden
  - **Living Off The Land Binaries** – Verwenden von Trusted Binaries (z.B. MSBuild.exe)
  - **Miskonfiguration** – Oftmals werden Regeln falsch konfiguriert (z.B. Mehrere CMD bzw. PowerShell Binaries am Rechner, nicht alle in Blacklist)

# Virus & Threat Protection



# Virus & Threat Protection

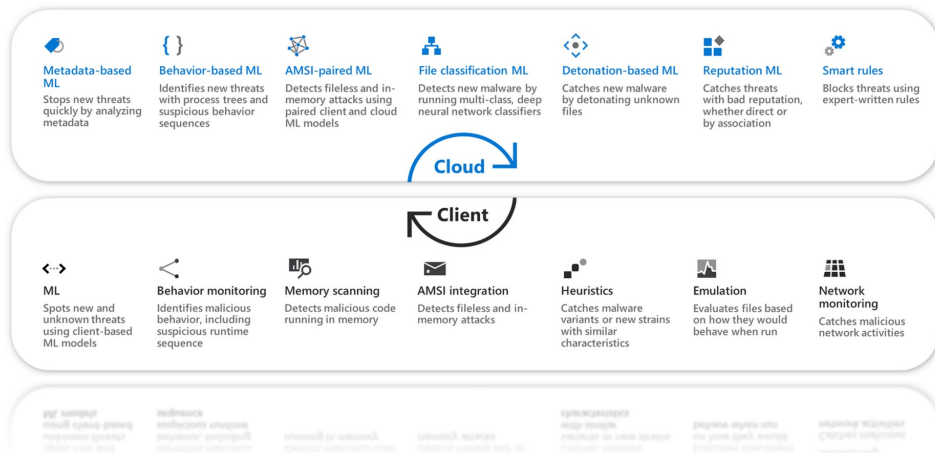
- Viele Maßnahmen zur Verteidigung gegen Bedrohungen in Windows
- Historisch gewachsen und an bestimmten Bedrohungslagen ausgerichtet - z.B. DEP bei Buffer Overflows, AMSI etc.
- Flickwerk an Sicherheitsmaßnahmen löst viele Probleme, Maßnahmen können aber umgangen werden
- Windows 11 setzt auf viele Verbesserungen, insbesondere:
  - Defender SmartScreen — Schutz vor Phishing, Malware Sites und Download von potentiell schadhaften Dateien
  - Manipulationsschutz (Tamper Protection) — Schutz kritischer Sicherheitskonfigurationen vor Änderung
  - Überwacher Ordnerzugriff — Zugriffsschutz auf definierte Ordner, beispielweise zum Schutz vor Ransomware Angriffen
  - Defender Anti Virus

# Windows Defender Antivirus

- AV-Lösung von Windows – effektiv in 2012 mit Windows 8 in heutiger Form etabliert
- Seit 2015 wird an Machine Learning und AI Integration gearbeitet
- Einsatz von „Definitions“ für Bedrohungen und Heuristiken für eine Detektion mittels „Probability-“ und Verhaltens-Scoring
- **ABER:** Cloud-Integration wird immer wichtiger, insbesondere wenn lokale ML Modelle nicht zweifelsfreie
- „Defender Next-Generation Protection“ umfasst:
  - Machine Learning
  - Big Data Analyse
  - Cloud-Integration
  - Tamper Protection
  - Web protection

# Windows Defender ATP

## Microsoft Defender ATP next generation protection engines



(Vergleiche <https://www.microsoft.com/en-us/security/blog/2019/06/24/>

[inside-out-get-to-know-the-advanced-technologies-at-the-core-of-microsoft-defender-atp-next-generation-protection/](https://www.microsoft.com/en-us/security/blog/2019/06/24/inside-out-get-to-know-the-advanced-technologies-at-the-core-of-microsoft-defender-atp-next-generation-protection/))

Vielen Dank!

**Alles erledigt, die Windows Welt ist sicher!**

# Windows Defender Endpoint Protection Fails

- Antivirus und Endpoint-Protection Systeme sind komplex - Viele Datenquellen müssen überwacht werden
- Betriebssystemfunktionalitäten sind komplex und über Jahrzehnte gewachsen, viele Redundanzen und technische Schulden
- Grundprobleme oft nicht trivial zu lösen
- Einzelne Mechanismen lösen oft nicht grundlegende Probleme -> z.B. AMSI mit grundlegenden Problemen seit Jahren, z.B.
  - „InvokeMimikatz“ wird erkannt, „ln“+„vo“+„ke“+„-M“+„im“+„ik“+„at“+„z“ nicht

# Ausblick: Agentic Security



**Microsoft  
Copilot**

# Agentic Security

- Microsoft plant zukünftig Windows als „Agentic OS“
- Starke Einbindung der Copilot AI, auch im Cloudumfeld wird dies stark forciert
- KI Systeme kämpfen mit ihren eigenen Sicherheitsproblemen
- Microsoft plant daher Umsetzung grundlegende Sicherheitsparadigmen:
  - Eigene Accounts & Sandboxes als Workspaces für AI Agents
  - Limitierung der Zugriffsrechte von Agents
  - Ausschließlich signierte Agents & Komponenten
- **ABER:** Sicherheit und Privacy bestehender AI Umsetzungen **wiesen immer wieder Mängel auf**

(Vergleiche <https://acuvity.ai/the-clawdbot-dumpster-fire-72-hours-that-exposed-everything-wrong-with-ai-security/>)

# Zusammenfassung

- Die Windows Registry kann durch Malware in vielerlei Hinsicht missbraucht werden
- (Lokale) Windows Credentials werden in LM/NTLM Format lokal gespeichert
- Windows Defender umfasst eine Vielzahl an Schutzmechanismen, die für sich aber **nicht** unumgebar sind
- Windows Services sind attraktive Ziele für Angreifer
- DLLs bieten einiges an Angriffsfläche
- UEFI ist nur wirkungsvoll, wenn ganze Bootkette gesichert ist
- AppLocker kann ebenfalls durch unterschiedliche Techniken umgangen werden

# Literatur/Quellen

- Microsoft Documentation - <https://docs.microsoft.com/de-de/>
- Microsoft Sysinternals - <https://docs.microsoft.com/en-us/sysinternals/>
- Bitlocker - <https://docs.microsoft.com/en-us/windows/security/information-protection/bitlocker/bitlocker-overview>
- UEFI Specifications - <https://uefi.org/specifications>
- Windows Internals: Band 1: Systemarchitektur, Prozesse, Threads, Speicherverwaltung, Sicherheit und mehr Developer Reference, M. Russinovich et al., 2018, ISBN: 9783960883395

# Literatur/Quellen II

- Windows Internals: Band 2 Developer Reference, M. Russinovich et al., 2021, ISBN: 9780135462409
- Windows Security Internals: A Deep Dive into Windows Authentication, Authorization, and Auditing, J. Forshaw, 2024, ISBN: 9781718501997
- Mastering Windows Security and Hardening: Secure and protect your Windows environment from cyber threats using zero-trust security principles, M. Dunkerley und M. Tumbarello, 2022, ISBN: 9781803236544
- Microsoft Defender for Endpoint in Depth: Take any organization's endpoint security to the next level, P. Huijbregts et al., 2023, ISBN: 9781804615171

**Vielen Dank!**

`https://establishing-security.at/`