Malware und Schutzmaßnahmen

IT-Sicherheit

ITT-Net-Is

29. März 2025

1 Was ist Malware?

Malware (ein Kofferwort aus ”malicious software”, deutsch: Schadsoftware) bezeichnet Programme, die entwickelt wurden, um unerwünschte und schädliche Funktionen auf einem Computersystem auszuführen. Im Gegensatz zu regulärer Software, die dem Nutzer dient, verfolgt Malware schädliche Absichten wie Datendiebstahl, Systemschädigung oder unerlaubte Systemkontrolle.

2 Verbreitungswege von Malware

Malware kann über verschiedene Wege auf ein System gelangen:

- E-Mail-Anhänge
- Infizierte Websites
- Manipulierte Downloads
- Externe Speichermedien
- Sicherheitslücken in Betriebssystemen und Anwendungen
- Soziale Netzwerke und Messenger-Dienste

3 Arten von Malware

3.1 Computerviren

Computerviren sind Schadprogramme, die sich selbst reproduzieren können, indem sie ihren eigenen Code in andere ausführbare Dateien oder Systembereiche einfügen. Sie werden aktiviert, wenn die infizierte Datei ausgeführt wird. Wichtige Merkmale:

- Benötigen einen ”Wirt” (eine andere Datei), um sich zu verbreiten
- Können nicht selbständig über Netzwerke verbreitet werden
- Führen oft destruktive Aktionen aus (Dateien löschen, Systeme beschädigen)
- Beispiele: Boot-Viren, Makro-Viren, Skript-Viren

3.2 Computerwürmer

Würmer sind eigenständige Programme, die sich selbst über Netzwerke verbreiten können, ohne eine Wirtsdatei zu benötigen. Charakteristika:

- Benötigen keine Benutzerinteraktion zur Verbreitung
- Nutzen oft Sicherheitslücken in Netzwerkprotokollen oder -diensten aus
- Verursachen häufig Netzwerküberlastungen durch ihre Verbreitungsaktivität
- Beispiele: Morris-Wurm, WannaCry (kombiniert mit Ransomware), Conficker

3.3 Trojaner

Trojaner (oder Trojanische Pferde) tarnen sich als nützliche Software, enthalten jedoch versteckte schädliche Funktionen. Eigenschaften:

- Keine Selbstreplikation wie bei Viren oder Würmern
- Täuschen Benutzer durch vermeintlich nützliche Funktionen
- Schaffen oft Hintertüren für weiteren Schadcode
- Beispiele: Remote Access Trojans (RATs), Banking-Trojaner, Adware-Trojaner

3.4 Information-Stealer

Information-Stealer (Datendiebe) sind spezialisierte Malware, die auf das Sammeln und Übermitteln vertraulicher Informationen abzielen:

- Spionieren Tastatureingaben aus (Keylogger)
- Stehlen Zugangsdaten für Online-Dienste
- Sammeln Kreditkarteninformationen und andere persönliche Daten
- Suchen gezielt nach bestimmten Dateitypen (Dokumente, Bilder)
- Beispiele: Formjacking-Angriffe, Zeus, Emotet

3.5 Ransomware

Ransomware verschlüsselt Dateien auf dem Opfersystem und fordert ein Lösegeld (meist in Kryptowährungen) für die Entschlüsselung:

- Blockiert den Zugriff auf Daten oder das gesamte System
- Setzt Zahlungsfristen mit Drohungen der Datenlöschung
- Nutzt starke Verschlüsselungsalgorithmen
- Beispiele: CryptoLocker, Locky, Ryuk, NotPetya

3.6 Weitere Malware-Typen

- Spyware: Überwacht Benutzeraktivitäten ohne Wissen des Nutzers
- Adware: Zeigt unerwünschte Werbung an
- Rootkits: Verstecken sich tief im System und gewähren anhaltenden Zugriff
- Botnets: Netzwerke infizierter Computer, die ferngesteuert werden
- Fileless Malware: Operiert nur im Arbeitsspeicher ohne Dateien auf der Festplatte
- Cryptojacking: Missbraucht Rechenleistung des Opfers zum Schürfen von Kryptowährungen

4 Tarnungsmethoden von Malware

4.1 Obfuskierung

Obfuskierung bezeichnet Techniken, die den Code einer Malware so verändern, dass er schwieriger zu analysieren ist:

- Umbenennung von Variablen und Funktionen
- Einfügen von nutzlosem Code
- Verschleierung der Programmlogik
- Verwendung ungewöhnlicher Programmierkonzepte

Hier ist ein kurzes Beispiel für Code-Obfuskierung in Python:

```python
# Ursprünglicher, lesbarer Code
def check_password(password):
    if password == "secret123":
        return True
    else:
        return False
```

Und hier ist eine obfuskierte Version:

```python
# Obfuskierte Version
import base64
____ = lambda _, __: base64.b64decode(_).decode() == __
_____ = lambda _: ''.join([chr(ord(c) ^ 42) for c in _])
__ = "K]GJ]LXWVc"
def ________(_______):
    return ____(_____("kc{i{pOqm{wmv"), _____(__)) if _ := _______ else not _
```

Notes

Diese obfuskierte Version verwendet mehrere Techniken:

- Verwirrende Variablennamen (nur Unterstriche in unterschiedlicher Anzahl)
- Verschleierung der Passwort-Zeichenkette durch XOR-Verschlüsselung
- Base64-Dekodierung
- Unnötige Lambda-Funktionen
- Verwirrende Kontrollfluss-Logik mit einem Conditional Expression

Die obfuskierte Version macht genau dasselbe wie die ursprüngliche Funktion, ist aber viel schwieriger zu verstehen und zu analysieren. Dies ist ein einfaches Beispiel – in der Praxis können Obfuskierungstechniken noch viel komplexer sein und mehrere Schichten von Verschleierung umfassen.

4.2 Polymorphe Malware

Polymorphe Malware verändert bei jeder Infektion automatisch ihren Code, behält aber ihre Funktionalität:

- Erzeugt bei jeder Verbreitung eine einzigartige Signatur
- Erschwert die Erkennung durch Signaturen
- Nutzt Verschlüsselungsroutinen mit wechselnden Schlüsseln
- Behält den gleichen Funktionskern

4.3 Metamorphe Malware

Metamorphe Malware geht über polymorphe Techniken hinaus und verändert ihre gesamte Struktur:

- Schreibt ihren eigenen Code vollständig um
- Verändert die Funktionsabfolge
- Nutzt alternative Methoden für die gleiche Funktionalität
- Äußerst schwer durch herkömmliche Antivirenprogramme zu erkennen

4.4 Fileless Malware

Fileless Malware hinterlässt keine Dateien auf der Festplatte:

- Läuft ausschließlich im Arbeitsspeicher
- Nutzt legitime Systemtools (Living-off-the-Land)
- Infiltriert die Registry oder andere persistente Bereiche
- Umgeht Datei-basierte Erkennungsmethoden

4.5 Anti-Analyse-Techniken

Moderne Malware implementiert Methoden, um Analyse und Erkennung aktiv zu verhindern:

- Erkennung von virtuellen Maschinen und Analyse-Umgebungen
- Verzögerte Ausführung des schädlichen Codes
- Erkennung von Debugging-Tools
- Abbruch der Ausführung bei Entdeckung von Analysetools
- Gezielte Ausführung nur in bestimmten Umgebungen oder Regionen

5 Schutzmaßnahmen gegen Malware

5.1 Präventive Maßnahmen

- Regelmäßige Software-Updates und Patches
- Sorgfältiger Umgang mit E-Mail-Anhängen und Downloads
- Starke, einzigartige Passwörter
- Restriktive Benutzerrechte (Prinzip der geringsten Berechtigung)
- Sichere Konfiguration von Netzwerkgeräten
- Schulung und Sensibilisierung der Benutzer

5.2 Technische Schutzmaßnahmen

- Antivirensoftware mit aktuellem Signaturendatenbank
- Firewalls (Hardware und Software)
- Intrusion Detection/Prevention Systems (IDS/IPS)
- E-Mail- und Web-Filter
- Anwendungssteuerung (Application Whitelisting)
- Netzwerksegmentierung

5.3 Backups und Wiederherstellung

- Regelmäßige Datensicherungen
- Offline-Backups (nicht ständig mit dem System verbunden)
- Test der Wiederherstellungsprozesse
- Dokumentierte Notfallpläne

6 Antivirensoftware und ihre Bewertungskriterien

6.1 Signaturbasierte Erkennung (Primäre Erkennungsrate)

- Vergleicht Dateien mit bekannten Malware-Signaturen
- Effektiv gegen bekannte Bedrohungen
- Benötigt regelmäßige Updates
- Unwirksam gegen neue, unbekannte Malware (Zero-Day-Exploits)

6.2 Heuristische Erkennung (Sekundäre Erkennungsrate)

- Analysiert Verhaltensweisen und Code-Strukturen
- Kann auch unbekannte Malware erkennen
- Basiert auf Regeln und Algorithmen
- Risiko von Fehlalarmen (False Positives)

6.3 Verhaltensbasierte Erkennung

- Überwacht das Laufzeitverhalten von Programmen
- Erkennt verdächtige Aktivitätsmuster
- Kann Zero-Day-Bedrohungen identifizieren
- Ressourcenintensiver als signaturbasierte Methoden

6.4 Cloud-basierte Erkennung

- Überprüft unbekannte Dateien gegen Online-Datenbanken
- Nutzt kollektive Intelligenz vieler Systeme
- Reduziert lokale Ressourcenbelastung
- Benötigt Internetverbindung

6.5 Bewertungskriterien für Antivirenlösungen

6.5.1 Primäre Erkennungsrate (Virusdefinitionen)

Die Fähigkeit, bekannte Malware anhand von Signaturen zu identifizieren:

- Umfang und Aktualität der Signaturdatenbank
- Geschwindigkeit der Integration neuer Signaturen
- Erkennungsrate bei standardisierten Tests (z.B. AV-TEST, AV-Comparatives)

6.5.2 Sekundäre Erkennungsrate (Heuristik)

Die Fähigkeit, unbekannte oder modifizierte Malware zu erkennen:

- Effektivität der heuristischen Analyse
- Verhältnis zwischen Erkennungsrate und Fehlalarmen
- Anpassungsfähigkeit an neue Bedrohungstypen

6.5.3 Reparaturleistung

Die Fähigkeit, infizierte Systeme zu bereinigen:

- Vollständige Entfernung von Malware (inkl. Registryeinträgen, versteckten Dateien)
- Wiederherstellung beschädigter Systemdateien
- Behandlung von Bootsektor-Infektionen
- Wiederherstellung verschlüsselter oder beschädigter Daten (wenn möglich)

6.5.4 Auswirkung auf die Systemleistung

Der Ressourcenverbrauch der Antivirensoftware:

- CPU- und RAM-Nutzung im Ruhezustand
- Ressourcenverbrauch während Scans
- Einfluss auf Startzeit des Systems
- Verzögerungen bei alltäglichen Operationen

6.5.5 Schutzbereiche (Einfallstore)

Die abgedeckten Infektionswege und Angriffsvektoren:

- E-Mail-Schutz
- Web-Schutz (HTTP/HTTPS-Filterung)
- Schutz vor Netzwerkangriffen
- USB-/Wechselmedien-Schutz
- Skript- und Makroschutz
- Schutz vor Social-Engineering-Angriffen

6.5.6 Funktionalität

Der Umfang der gebotenen Sicherheitsfunktionen:

- Echtzeit-Schutz
- On-Demand-Scanning
- Automatische Updates
- Quarantäne-Management
- Ausnahmeregeln und Anpassungsmöglichkeiten
- Zusatzfunktionen (Firewall, Kindersicherung, VPN, etc.)

6.5.7 Bedienung

Die Benutzerfreundlichkeit der Software:

- Intuitive Benutzeroberfläche
- Klare Darstellung von Bedrohungen und Maßnahmen
- Anpassbarkeit für verschiedene Nutzergruppen
- Hilfestellung und Dokumentation
- Benachrichtigungskonzept

6.5.8 Zentrale Administrierbarkeit

Die Verwaltungsmöglichkeiten in Unternehmensumgebungen:

- Zentrales Management-Dashboard
- Gruppenrichtlinien und -konfigurationen
- Berichterstattung und Alarmierung
- Ferninstallation und -wartung
- Integration in bestehende IT-Management-Systeme

6.5.9 Selbstschutz der Antivirensoftware

Die Fähigkeit, sich gegen Angriffe auf die eigene Funktionalität zu schützen:

- Schutz vor Deaktivierung durch Malware
- Manipulationssicherheit der eigenen Komponenten
- Schutz der Update-Mechanismen
- Widerstandsfähigkeit gegen DoS-Angriffe auf die Software

7 Praxisaufgabe: Bewertung einer Antivirensoftware:

7.1 Aufgabenstellung

Bewerten Sie eine Antivirensoftware nach folgenden Kriterien:

1. Primäre Erkennungsrate (Virusdefinitionen)
2. Sekundäre Erkennungsrate (Heuristik)
3. Reparaturleistung
4. Ausbremsen der Systemleistung
5. Schutzbereiche (Einfallstore)
6. Funktionalität
7. Bedienung
8. Zentrale Administrierbarkeit
9. Selbstschutz der Antivirensoftware

7.2 Methodik zur Bewertung

1. **Auswahl der Software:** Wählen Sie eine aktuelle Antivirenlösung aus (kostenlos oder kommerziell).
2. **Informationssammlung:**

- Hersteller-Dokumentation
- Unabhängige Testberichte (AV-TEST, AV-Comparatives, etc.)
- Nutzerbewertungen

1. **Bewertungsschema:**

- Entwickeln Sie eine Skala (z.B. 1-5 Punkte pro Kriterium)
- Definieren Sie klare Bewertungsmaßstäbe für jedes Kriterium
- Gewichten Sie die Kriterien nach Relevanz für Ihren Einsatzbereich

1. **Auswertung:**
    - Erstellen Sie eine Gesamtbewertung
    - Heben Sie Stärken und Schwächen hervor
    - Formulieren Sie eine Empfehlung mit Begründung

7.3 Beispiel für ein Bewertungsformular

Kriterium	Bewertung (1-5)	Begründung
Primäre Erkennungsrate		[Hier Ihre Beobachtungen eintragen]
Sekundäre Erkennungsrate		[Hier Ihre Beobachtungen eintragen]
Reparaturleistung		[Hier Ihre Beobachtungen eintragen]
Systembelastung		[Hier Ihre Beobachtungen eintragen]
Schutzbereiche		[Hier Ihre Beobachtungen eintragen]
Funktionalität		[Hier Ihre Beobachtungen eintragen]
Bedienung		[Hier Ihre Beobachtungen eintragen]
Zentrale Administrierbarkeit		[Hier Ihre Beobachtungen eintragen]
Selbstschutz		[Hier Ihre Beobachtungen eintragen]
Gesamtbewertung		[Zusammenfassung und Empfehlung]

7.4 Sicherheitshinweise zur Durchführung

- Führen Sie Tests mit potenziell gefährlicher Malware nur in isolierten Umgebungen durch
- Verwenden Sie Virtualisierung oder Sandbox-Lösungen
- Ziehen Sie unabhängige Testberichte heran, anstatt selbst mit aktiver Malware zu experimentieren
- Beachten Sie, dass einige Tests in produktiven Umgebungen Risiken bergen können