Bericht: Umgehung von Deep Packet Inspection und Sicherheitsmechanismen

Inhaltsverzeichnis

Einleitung

Takko setzt nach meiner Erinnerung eine Deep Packet Inspection (DPI) ein, um willkürliche Downloads von Angestellten zu blockieren und unerwünschte Aktivitäten zu verhindern. Trotz dieser Sicherheitsmaßnahme besteht die Möglichkeit, blockierte Inhalte herunterzuladen, indem man alternative Ansätze nutzt. Dieser Bericht dokumentiert eine Methode, die ich entdeckt habe, und beleuchtet die potenziellen Sicherheitsrisiken.

Beobachtung

Bei dem Versuch, ein Plugin-Update in Visual Studio Code über den Plugin Store durchzuführen, wurde der Download blockiert. Mögliche Gründe könnten die DPI oder andere Sicherheitslösungen wie SentinelOne sein. Ich habe daraufhin eine Technik angewendet, die häufig in Schadcode vorkommt, um diese Blockade zu umgehen.

Methode

Die Methode basiert auf der Verwendung von Encoding-Verfahren, speziell Base64, um blockierte Dateien in einer alternativen Form herunterzuladen und anschließend zurückzuwandeln.

Schritte

1. **Datei-Download und Encoding auf externer Instanz**
    - Auf einer externen Instanz (z. B. Google Colab) wird die blockierte Datei heruntergeladen.
    - Ein Python-Skript ermittelt die Dateigröße und den SHA256-Hash der Originaldatei zu Logging-Zwecken.
    - Die Datei wird in Base64 kodiert und unter dem Namen `<file_name>.b64` gespeichert.

    Beispiel: Encoding-Prozess

```
                                    File Encoding Summary
┌──────────────────────────┬──────────────┬──────────────────────────────────────────
──────────────────────────────────┐
│ File                     │ Size (bytes) │ SHA256 Hash
│
├──────────────────────────┼──────────────┼──────────────────────────────────────────
──────────────────────────────────┤
│ plugin.VSIXPackage       │     15443996 │
9ff08246ce36239fe3048b1ad2df46d1ae6c67671b6f24f26bddc24aa4747bda │
│ plugin.VSIXPackage.b64 │     20591996 │
00c865be0b439acfad21f306aeab49003522440fcc71071e816786d288bf3671 │
└──────────────────────────┴──────────────┴──────────────────────────────────────────
──────────────────────────────────┘
File 'plugin.VSIXPackage' has been successfully encoded to
'plugin.VSIXPackage.b64'.
```

2. **Download der kodierten Datei**

    Die Base64-kodierte Datei kann problemlos heruntergeladen werden, da sie die Sicherheitsmechanismen der DPI umgeht.

3. **Decoding auf lokalem Rechner**

    Ein weiteres Python-Skript wandelt die Base64-kodierte Datei zurück in die Originaldatei. Die Datei wird anschließend mit den ursprünglichen Hashes verglichen, um die Integrität zu bestätigen.

Beispiel: Decoding-Prozess

```
(venv) PS amy> python .\to_exe.py
                                       File Decoding Summary
┏━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┓
┃ File                     ┃ Size (bytes) ┃ SHA256 Hash
┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━┇━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┩
│ plugin.VSIXPackage.b64 │     20591996 │
00c865be0b439acfad21f306aeab49003522440fcc71071e816786d288bf3671 │
│ plugin.VSIXPackage     │     15443996 │
9ff08246ce36239fe3048b1ad2df46d1ae6c67671b6f24f26bddc24aa4747bda │
└─────────────────────────┴──────────────┴─────────────────────────────────
──────────────────────────────────┘
File 'plugin.VSIXPackage.b64' has been successfully decoded to
'plugin.VSIXPackage'.
(venv) PS amy>
```

4. **Installation in Visual Studio Code**

   Die Originaldatei konnte erfolgreich als Plugin-Update in Visual Studio Code installiert werden.

Sicherheitsimplikationen

Die beschriebene Methode demonstriert, dass blockierte Inhalte durch Encoding-Verfahren an Sicherheitsmechanismen wie DPI, IDS (Intrusion Detection System) oder EDR (Endpoint Detection and Response) vorbeigeschleust werden können. Dies eröffnet potenziell Angriffsvektoren für böswillige Akteure:

1. **Einbringen von Schadsoftware**

   Die Möglichkeit, Daten an Sicherheitslösungen vorbei ins Netzwerk zu schleusen, könnte von Angreifern genutzt werden, um Schadsoftware einzuschleusen.

2. **Exfiltration von Daten**

   Die gleiche Technik könnte verwendet werden, um sensible Daten aus dem Netzwerk herauszuschleusen.

Fazit

Die beschriebene Methode zeigt eine Schwachstelle in der bestehenden Sicherheitsinfrastruktur auf. Es ist wichtig, diese Schwachstelle zu adressieren, um potenzielle Risiken zu minimieren. Eine detaillierte Analyse und Anpassung der DPI-Regeln sowie eine Prüfung auf Encoding-Techniken in IDS und EDR-Systemen könnten hier Abhilfe schaffen.

Empfohlene Maßnahmen

- Überprüfung und Anpassung der DPI-Regeln, um Encodings wie Base64 zu erkennen und zu blockieren.
- Einsatz von erweiterten Sicherheitslösungen, die Encoding-Techniken identifizieren und analysieren können.
- Sensibilisierung der Mitarbeiter bezüglich sicherheitskritischer Verfahren und potenzieller Umgehungsmethoden.
- Durchführung von Penetrationstests, um weitere Schwachstellen aufzudecken und zu beheben.

Metadaten

> Autor: Hendrik Siemens
>
> Datum: 08.05.2025
>
> SHA512 Hash:
> eb603c54b51426d51c8646cccb7cfe90696c6f612b5d7034a3356d9b7e8dc4a00ed9d2577638910dd5ee6e3373643caaf01715472154c480d899d9e31dfec416