CVE-2020-0601 (Curveball) – Proof-of-Concept

Kurze Zusammenfassung und erweiterter PoC-Überblick

Einleitung

CVE-2020-0601, auch „Curveball“ genannt, ist eine Spoofing-Schwachstelle in der Windows CryptoAPI (Crypt32.dll). Angreifer können manipulierte elliptische Kurvenparameter nutzen, um gefälschte ECC-Zertifikate als vertrauenswürdig zu präsentieren.

Hintergrund

Elliptische Kurven-Kryptographie (ECC) verwendet die Struktur von Punkten auf einer elliptischen Kurve über endlichen Feldern, um öffentliche und private Schlüssel zu erzeugen. Windows' CryptoAPI unterstützt insbesondere die NIST-Standardkurven P-256 (secp256r1) und P-384 (secp384r1).

- **Kurvenparameter:** Jede Kurve wird durch Parameter a, b, den Primkörper p, einen Basispunkt G und seine Ordnung n definiert.
- **Punkt-Validierung:** Um die Echtheit eines öffentlichen Schlüssels zu prüfen, muss sichergestellt werden, dass der Punkt tatsächlich auf der Kurve liegt und zur richtigen Untergruppe gehört.
- **Fehlerquelle:** In Crypt32.dll wurde zwar kontrolliert, ob der gegebene „öffentliche Schlüssel“ zum Generator multipliziert werden kann, jedoch entfiel die vollständige Prüfung, ob alle Kurvenparameter – vor allem die Gruppenordnung und die Zugehörigkeit zur Primärkurve – korrekt sind.

Schwachstelle

- **Komponente:** Crypt32.dll (Windows 10, Server 2016/2019)
- **Typ:** Fehlerhafte ECC-Parameter- und Punktvalidierung
- **Auswirkung:** Erstellung und Einsatz gefälschter Code-Signing-Zertifikate

Proof-of-Concept (PoC)

Die PoC-Implementierung umfasst folgende Schritte:

1. **Root-CA-Zertifikat laden:** Herunterladen des legitimen Microsoft ECC-Root-CA-Zertifikats.
2. **Spoofed CA-Key ableiten:** Mit einem Python-Skript (`gen-key.py`) aus den Root-Public-Key-Daten einen neuen Privatschlüssel gewinnen.
3. **Gefälschtes CA-Zertifikat generieren:** Erzeugen eines X.509-Zertifikats mit manipulierten Parametern via OpenSSL.
4. **Leaf-Zertifikat und CSR erstellen:** Generierung eines frischen ECC-Schlüsselpaares (`openssl ecparam`) und einer Certificate Signing Request mit v3-Erweiterungen.
5. **Signatur mit Spoofed CA:** Signieren des CSR durch das gefälschte CA-Zertifikat (`openssl x509 -req`), um ein gültig scheinendes Leaf-Zertifikat zu erhalten.
6. **PKCS#12-Bundle und Code Signing:** Export aller Zertifikate und Schlüssel in eine `.p12`-Datei und Signatur einer Windows-Binärdatei mit `osslsigncode`.
7. **Verifikation:** Prüfung der Signatur unter Windows (Explorer, `signtool`) – die manipulierte CA erscheint als vertrauenswürdig.