TECHNISCHE HOCHSCHULE DEGGENDORF

Bachelor Cyber Security

Kryptologie 2

Projektbericht

Musterlösung zur Cryptochallenge: CurveBall

Autor:

Manuel Friedl, Matrikel-Nr.: 1236626
Christof Renner, Matrikel-Nr.: 22301943

Betreuer:

Prof. Dr. Martin Schramm

Deggendorf - May 15, 2025

Eigenständigkeitserklärung

Hiermit bestätige ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen Publikationen, Vorlagen und Hilfsmitteln als die angegebenen benutzt habe. Alle Teile meiner Arbeit, die wortwörtlich oder dem Sinn nach anderen Werken entnommen sind, wurden unter Angabe der Quelle kenntlich gemacht.

Musterstadt, _____
(Datum) (Unterschrift)

Musterlösung zur CurveBall-Challenge

Ziel der Aufgabe

Ziel war es, die simulierte Schwachstelle CVE-2020-0601 ('CurveBall') auszunutzen. Dazu musste ein manipuliertes ECC-Schlüsselmaterial erzeugt werden, mit dem eine Signatur erstellt werden kann, die von einer verwundbaren Validierungslogik akzeptiert wird.

Durchführung

1. Mit dem Tool `gen-key.py` wurde ein manipulierter privater Schlüssel generiert:

   ```
   python gen-key.py USERTrustECCCertificationAuthority.crt
   ```

2. Es entstand eine Datei `p384-key-rogue.pem` mit einem privaten Schlüssel basierend auf einem manipulierten Generatorpunkt G'.

3. Die Datei `simulate_vuln_check.py` überprüft die Signatur:

   ```
   from ecdsa import SigningKey, NIST384p, BadSignatureError
   import hashlib

   with open("p384-key-rogue.pem", "rb") as f:
       rogue_key = SigningKey.from_pem(f.read())

   rogue_pub = rogue_key.verifying_key
   message = b"Simulated CVE-2020-0601 message"
   digest = hashlib.sha256(message).digest()
   signature = rogue_key.sign_digest(digest)

   try:
       if rogue_pub.verify_digest(signature, digest):
           print('Verwundbare Pruefung akzeptiert die Signatur.')
   except BadSignatureError:
       print("Ungueltige Signatur.")
   ```

4. Die simulierte Validierung gab aus:

   ```
   Verwundbare Prüfung akzeptiert die Signatur.
   ```

Erklärung

Die Windows CryptoAPI überprüfte früher nicht, ob der öffentliche Schlüssel aus dem erwarteten Generatorpunkt G stammt. Ein Angreifer kann also G' und d' so wählen, dass $Q = d' \cdot G'$ zum legitimen CA-Key Q passt.

Die veraltete Logik prüfte:

- Ist Q derselbe wie der vertrauenswürdige Key? $\Rightarrow$ Ja
- Ist die Signatur gültig für Q? $\Rightarrow$ Ja

Damit akzeptierte sie die Signatur, obwohl sie mit einem manipulierten Schlüssel erstellt wurde.

Fazit

Die Challenge konnte erfolgreich abgeschlossen werden. Die Signatur wurde unter Verwendung eines manipulierten Schlüsselpaars erzeugt, welches mathematisch äquivalent zum Originalkey erschien, jedoch auf einem manipulierten Generatorpunkt basierte.

Die simulierte Umgebung erlaubte die Prüfung der Schwachstelle ohne ein verwundbares Windows-System.