TECHNISCHE HOCHSCHULE DEGGENDORF

Bachelor – Cyber Security

Kryptologie 2

Projektdokumentation

Cryptochallenge: CurveBall (CVE-2020-0601)

Autoren
Manuel Friedl – 1236626
Christof Renner – 22301943

Betreuer
Prof. Dr. Martin Schramm

Deggendorf, 22. Juli 2025

Inhaltsverzeichnis

1 Einleitung und Projektkontext

1.1 Motivation

Die Schwachstelle **CurveBall** (CVE-2020-0601) in der Windows-CryptoAPI ermöglicht es, X.509-Zertifikate mit manipulierten Elliptic-Curve-Parametern zu signieren, sodass betroffene Windows-Versionen die Signaturen fälschlich als gültig akzeptieren.[1] Im Modul *Kryptologie 2* fehlte bislang ein modernes Hands-On-Szenario, um diesen Fehler praktisch zu demonstrieren.

1.2 Projektziele

1. **Didaktik**: Vollständiger Angriffszyklus von Discovery bis Exploit.
2. **Sicherheit**: Deployment selbst muss trotz absichtlich verletzter Krypto sicher sein.
3. **Portabilität**: Schnelle, plattformunabhängige Nutzung via Docker/Podman.

1.3 Bedrohungsmodell

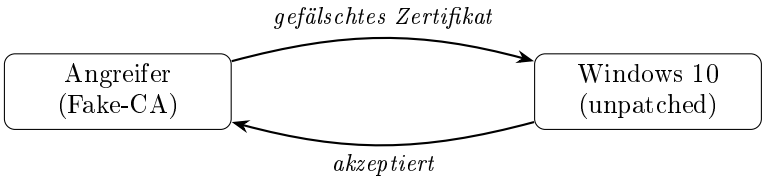

Abbildung 1: Simplifiziertes Threat-Model: fehlende Parameter-Validierung

2 Herangehensweise

2.1 Zielsetzung

- Demonstration des Angriffs in einer kontrollierten Umgebung.
- Vermittlung von DevSecOps-Best-Practices (Linting, CI, Scans).
- Bereitstellung als *„One-Click“*-Container, ohne lokale OpenSSL-Konfiguration.

2.2 Methodik

Wir arbeiteten in zwei Sprints à zwei Wochen. Abbildung 2 zeigt den iterativen Ablauf.

Abbildung 2: Iterativer Projektablauf

[1] Microsoft Security Advisory ADV200002, 14. 01. 2020

3 Arbeitsaufteilung

Die Arbeitsaufteilung erfolgte pragmatisch basierend auf den individuellen Stärken der Teammitglieder, wobei gleichzeitig Wert auf Wissenstransfer und gemeinsames Lernen gelegt wurde.

Teammitglied	**Hauptaufgaben**	**Spezifische Aufgaben**
Christof Renner	Kryptographie & Frontend	• Analyse der CVE-2020-0601 Schwachstelle • Entwicklung der Python-Skripte `gen_key.py` und `simulate_vuln_check.py` • Entwicklung des Web-Interface und Zertifikats-Visualizers • Erstellung verständlicher Challenge-Beschreibungen
Manuel Friedl	DevOps & Infrastruktur	• Docker-Containerisierung mit Multi-Stage-Builds • CI/CD-Pipeline mit GitHub Actions • Security-Scanning und Linting-Integration • GitHub Container Registry Konfiguration

Dabei wurde durch regelmäßige Abstimmungen und gemeinsame Review-Sessions sichergestellt, dass alle Komponenten nahtlos zusammenarbeiten. Die technische Dokumentation wurde gleichmäßig zwischen beiden Teammitgliedern aufgeteilt, wobei jeder etwa 50% der Dokumentationsarbeit übernahm.

3.1 Kollaborative Arbeitsweise

Trotz der Spezialisierung arbeiteten beide Teammitglieder eng zusammen. Code-Reviews waren Standard, wobei jeder die Arbeit des anderen validierte. Die tägliche Kommunikation erfolgte über Matrix-Chat mit kurzen Standups, ergänzt durch wöchentliche Reviews mit dem Betreuer.
Das Projektmanagement wurde über GitHub Project Board abgewickelt, was eine transparente Aufgabenverfolgung und effiziente Koordination ermöglichte. Besonders bei komplexen Implementierungen wurden Pair-Programming-Sessions durchgeführt.

3.2 Herausforderungen und Lösungsansätze

Die größten technischen Herausforderungen lagen in der Komplexität der elliptischen Kurven-Mathematik und der Anforderung, sichere Container für absichtlich unsichere Software zu erstellen. Organisatorisch mussten unterschiedliche Stundenpläne koordiniert und Abhängigkeiten zwischen den Arbeitspaketen geschickt gemanagt werden.

Diese ausgewogene Arbeitsaufteilung führte zu einem erfolgreichen Projektergebnis und einem erheblichen Lernzuwachs für beide Teammitglieder.

4 Technische Implementierung

4.1 Funktionsweise der Vulnerability

Windows validiert EC-Signaturen ohne sicherzustellen, dass der öffentliche Schlüssel Q und der Generator G wirklich zur deklarierten Kurve gehören. Angreifer ersetzen G durch einen Punkt mit kleiner Untergruppen-Order, sodass ECDSA trivial gebrochen wird.

5 Containerisierung

5.1 Dockerfile

```
FROM python:3.12-slim AS build
RUN apt-get update && apt-get install -y --no-install-recommends \
    libssl-dev build-essential && rm -rf /var/lib/apt/lists/*
COPY requirements.txt .
RUN pip install --prefix=/install -r requirements.txt

FROM python:3.12-slim
COPY --from=build /install /usr/local
COPY curveball-ctf /app
WORKDIR /app
USER 1001:1001
ENTRYPOINT ["python", "-m", "http.server", "8080"]
```

Auflistung 1: Auszug aus dem finalen Dockerfile

5.2 Architektur

Abbildung 3: Container-Deployment in der Lehrumgebung

6 CI/CD-Pipeline

GitHub Actions orchestriert *Lint* → *Build* → *Test* → *Scan* → *Push*.

6.1 Workflow-Datei (gekürzt)

```
jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - uses: actions/setup-python@v5
        with: { python-version: "3.12" }
```

```
8
9       - name: Ruff Lint
10        run: ruff check .
11
12      - name: Build Image
13        uses: docker/build-push-action@v5
14        with:
15          tags: curveball:ci
16          push: false
17
18      - name: Trivy Security Scan
19        uses: aquasecurity/trivy-action@v0.20.0
20        with:
21          image-ref: curveball:ci
22          severity: HIGH,CRITICAL
```

Auflistung 2: ci.yml – Kernschritte

6.2 Linting-Tools

- **ruff**: Python-Lint inkl. Import-Sortierung
- **markdown-lint**: prüft `README.md`
- **hadolint**: Dockerfile-Best-Practices

7 Nicht umgesetzte VM-Erweiterung

Ursprünglich war eine vorgefertigte Windows 10-VM (1909, ungepatcht) geplant, um den Angriff bis zum System-Rootstore zu demonstrieren. Dies scheiterte aus folgenden Gründen:

Licensing Weitergabe eines vorinstallierten Windows-Images verstößt gegen EULA.

Storage 8 GB-Image hätte Git LFS/Kosten gesprengt.

CI-Runner GitHub-Actions erlaubt keine Nested-Virtualisation.

Die Container-Variante ist mit rund 280 MB wesentlich leichter verteilbar.

8 Ausblick

1. **Windows-Live-Lab**: über Azure Lab Services echte, gepatchte und ungepatchte Hosts anbieten.
2. **Automatisierte Angriffskette**: Browser-Automation mit Playwright für einen End-to-End-Exploit.
3. **Gamification**: Flag-Server, Leaderboard und Achievements.

9 Fazit

Unser Projekt zeigt, wie sich ein kritischer Krypto-Bug in eine didaktisch wertvolle, aber dennoch sichere Übungsumgebung überführen lässt. Neben dem technischen Verständnis für CurveBall gewannen Studierende Einblicke in DevSecOps-Workflows, die heute in jeder Entwicklungsumgebung zum Standard gehören.