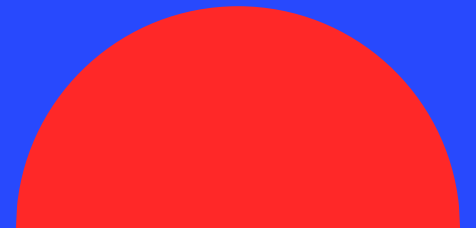
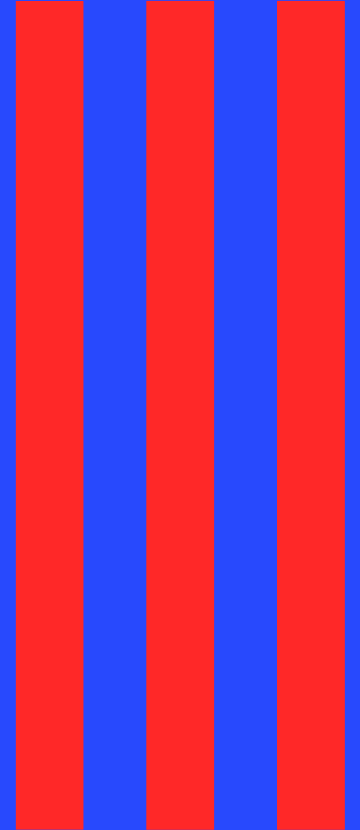


# Kryptologie II

## Curveball

Christof Renner & Manuel Friedl

28.07.2025



# Agenda

Einleitung & Projektüberblick

Hintergrund: CVE-2020-0601

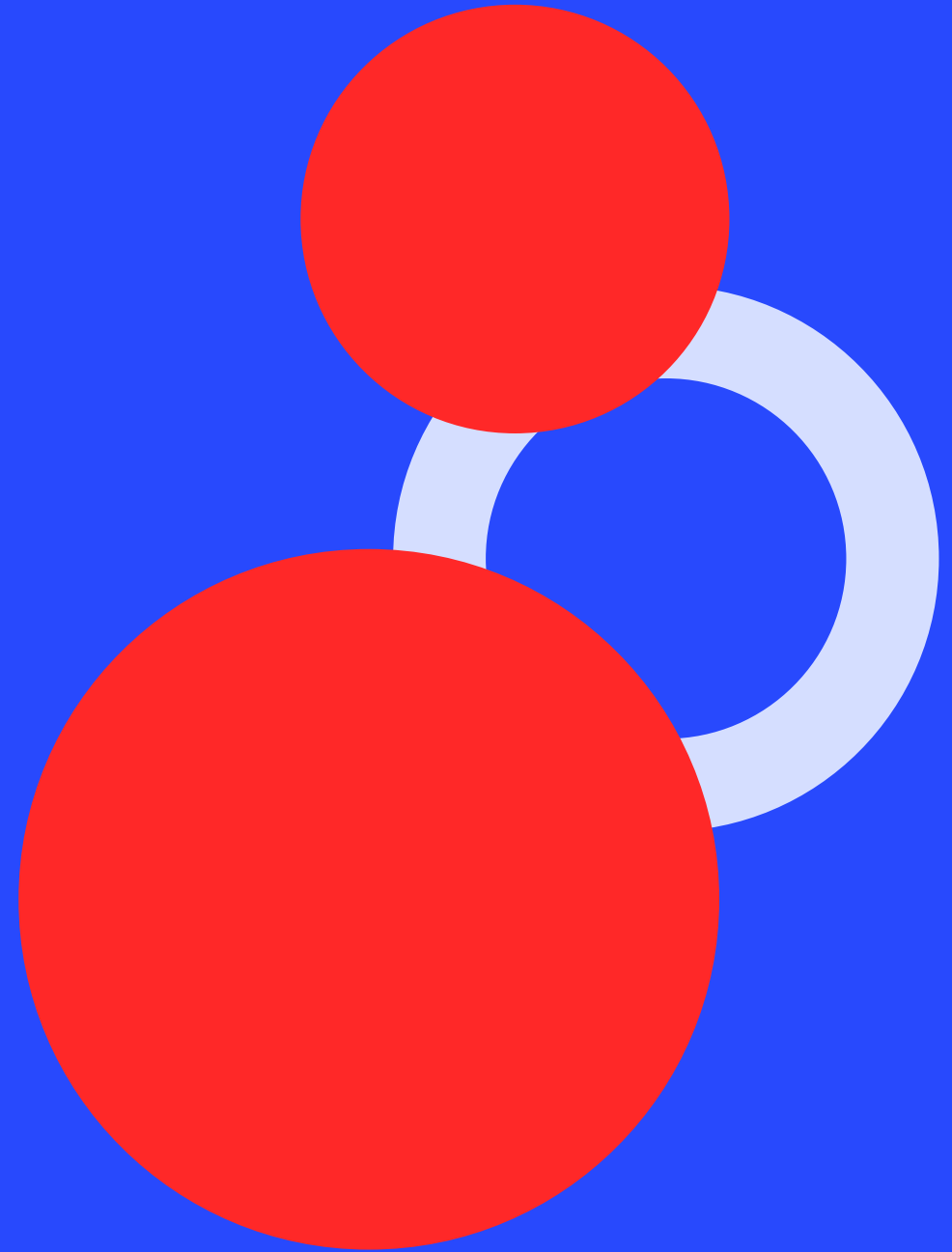
Kryptologie hinter Curveball

Aufbau der Challenge

Fazit



# Einleitung & Projektüberblick



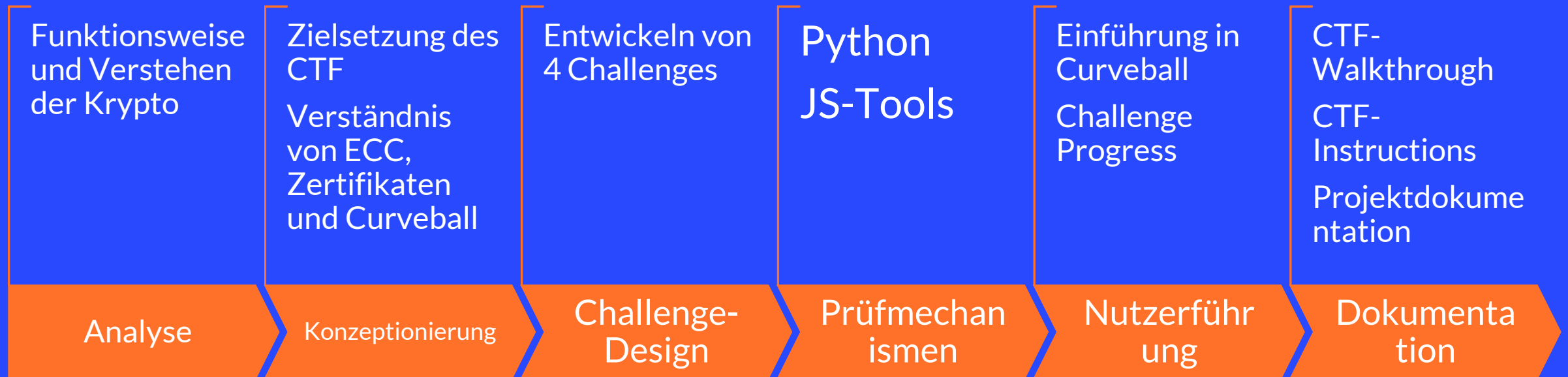
# Einleitung

Proof-of-Concept (PoC) für Microsofts  
ECC-Bug CurveBall (CVE-2020-0601)



CTF-Challenge zur  
Nachstellung des Exploits

# Projektüberblick



# Projektüberblick

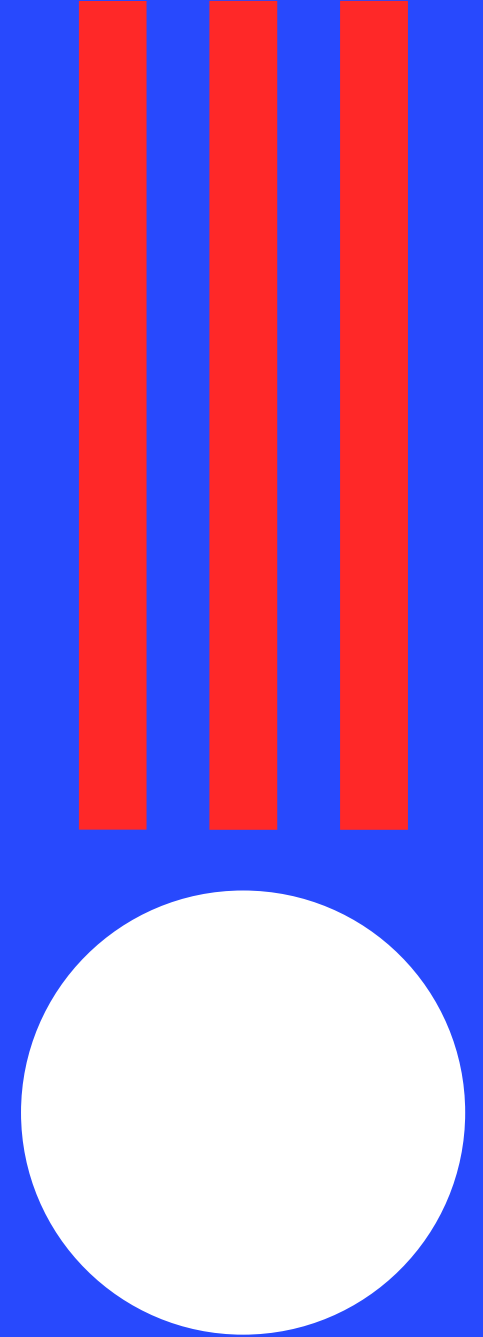
Flask

Docker

OpenSSL

HTML, CSS, JS

Hintergrund: CVE-2020-0601



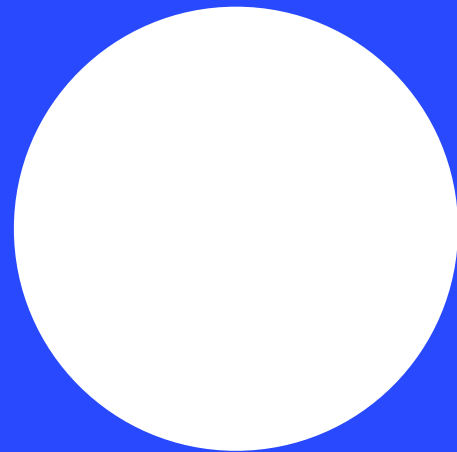
# Hintergrund: CVE-2020-0601





# Hintergrund: CVE-2020-0601

- Impact auf Windows 10 und Server 2016/2019



- Mögliche Angriffe
  - Man-in-the-Middle (HTTPS, RDP)
  - Spoofed Code-Signatures
- Weltweites Risiko für TLS-, E-Mail- und Code-Signatur-Chains

# Hintergrund: CVE-2020-0601

- CryptoAPI akzeptiert selbstdefinierte Parameter für NIST-Kurven.
- Angreifer wählt eigenes Generator-Punkt  $G'$  mit derselben Ordnung  $q$ .

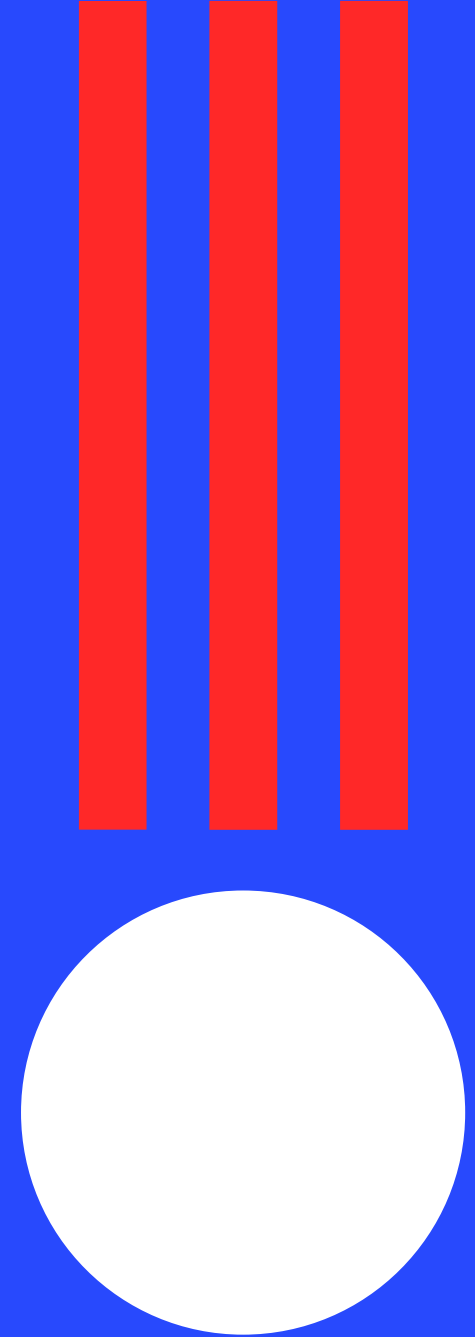
Ergebnis: gefälschte ECDSA-Zertifikate sehen echt aus.

## Lösung Microsoft:

- Ist  $P$  auf der richtigen Kurve?
- Stimmt  $G$ ?
- Ist die Ordnung korrekt  $\geq n$ ?

# Kryptologie hinter

# Curveball



# Grundlagen der ECC

- Elliptische Kurven sind Gleichungen der Form:  $y^2 = x^3 + a \cdot x + b \bmod p$
- Kurve über einem endlichen Körper z. B. **NIST P-256**
- Der Öffentliche Schlüssel Q wird erzeugt:  $Q = d * G$
- Dabei ist:
  - G der Generatorpunkt (bekannt)
  - D der private Schlüssel (geheim)
  - Q der öffentliche Schlüssel
  - Steht für die Punktmultiplikation

# Was Curveball ausnutzt

Curveball betrifft die **Überprüfung von ECC-Zertifikaten** in Windows:

- Microsoft CryptoAPI prüft **nicht vollständig**, ob der übermittelte öffentliche Schlüssel  $Q$  tatsächlich zur Kurve gehört
- Es werden also manipulierte Kurvenparameter akzeptiert

**Angreifer kann eigene elliptische Kurve definieren und einen Schlüssel darauf erzeugen welcher, zwecks fehlender Validierung trotzdem akzeptiert wird**

# Angriffsszenario

Angreifer definieren eine manipulierte Kurve, deren Generatorpunkt sie selbst kontrollieren:

1. Sie wählen einen eigenen Punkt  $G'$
2. Erzeugen einen privaten Schlüssel  $d$
3. Berechnen  $Q' = d * G'$
4. Signieren etwas mit diesem Schlüssel
5. Und bauen Zertifikat, das vorgibt, die Standardkurve zu nutzen

**Aufgrund fehlender Validierung ist es so möglich gefälschte Zertifikate auszustellen.**

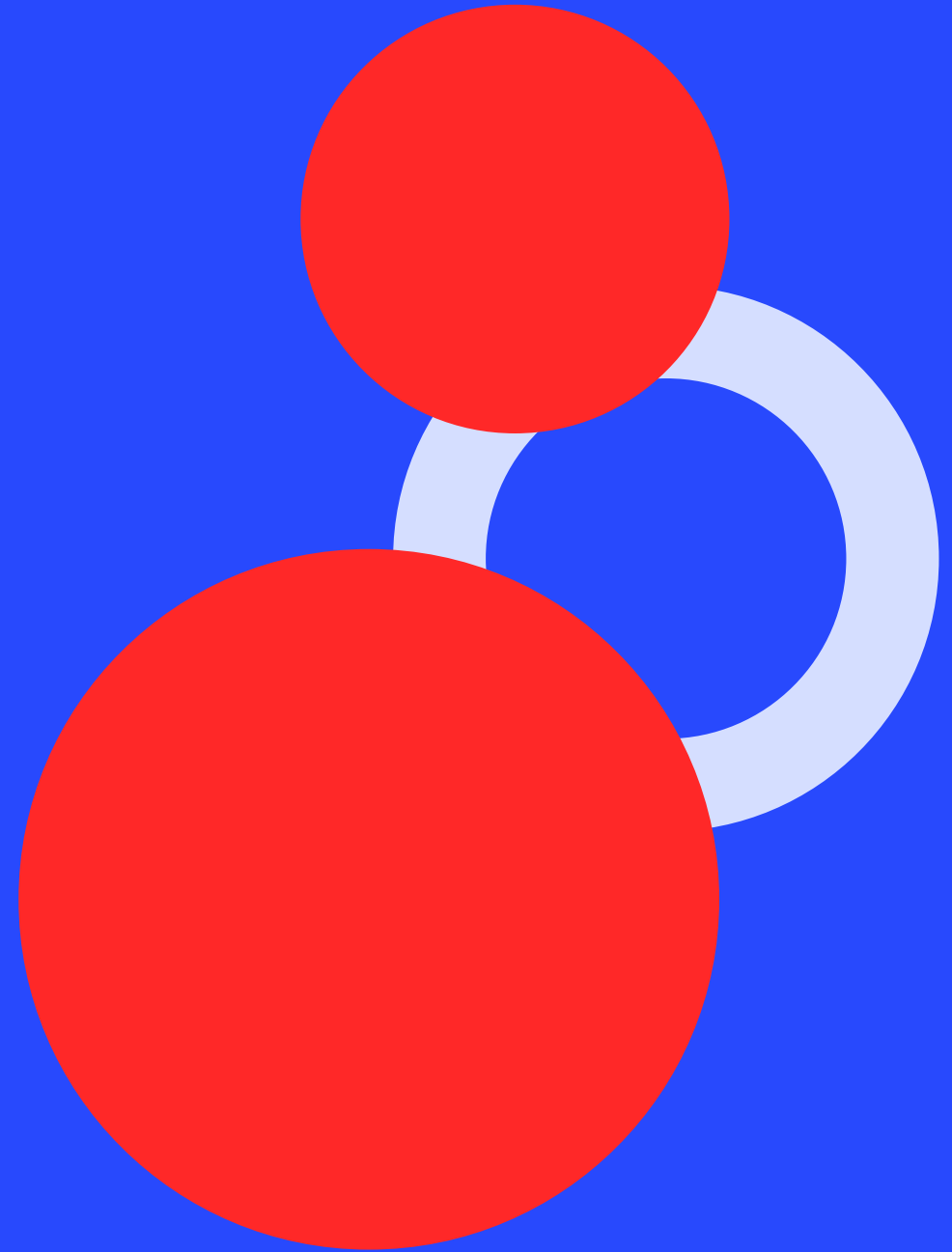
# Funktionsweise

Normalerweise wird geprüft:

- Ist der öffentliche Schlüssel  $Q$  ein gültiger Punkt auf der Kurve?
- Stammt  $Q$  aus der korrekten Ordnung?

=> Fehlende Validierung einiger Parameter

# Aufbau der Challenge





# Aufbau der Challenge

```
PS C:\Users\ChristofRenner\Git\SS25-Kryptologie2\curveball-ctf> docker-compose up --build
[+] Building 3.0s (13/13) FINISHED
=> [internal] load local bake definitions 0.0s
=> => reading from stdin 469B 0.0s
=> [internal] load build definition from Dockerfile 0.1s
=> => transferring dockerfile: 331B 0.0s
=> [internal] load metadata for docker.io/library/python:3.12-alpine 1.9s
=> [internal] load .dockerignore 0.0s
=> => transferring context: 2B 0.0s
=> [1/6] FROM docker.io/library/python:3.12-alpine@sha256:9b8808206f4a956130546a32cbdd8633bc973b19db2923b7298e6f 0.0s
=> [internal] load build context 0.0s
=> => transferring context: 1.45kB 0.0s
=> CACHED [2/6] RUN apk add --no-cache openssl=3.5.1-r0 0.0s
=> CACHED [3/6] WORKDIR /app 0.0s
=> CACHED [4/6] COPY requirements.txt /app/ 0.0s
=> CACHED [5/6] RUN pip install --no-cache-dir --requirement requirements.txt 0.0s
=> CACHED [6/6] COPY . /app 0.0s
=> exporting to image 0.0s
=> => exporting layers 0.0s
=> => writing image sha256:bf17f2dae9ec1424426cd356e046dd53a30c3c83a76d97b0204a07064f26f338 0.0s
=> => naming to docker.io/library/curveball-ctf-webserver 0.0s
=> resolving provenance for metadata file 0.0s
[+] Running 1/1
✔webserver Built 0.0s
Attaching to curveball-webserver
curveball-webserver | * Serving Flask app 'server'
curveball-webserver | * Debug mode: off
curveball-webserver | WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
curveball-webserver | * Running on all addresses (0.0.0.0)
curveball-webserver | * Running on https://127.0.0.1:8443
curveball-webserver | * Running on https://172.19.0.2:8443
curveball-webserver | Press CTRL+C to quit
```

# Aufbau der Challenge

## CURVEBALL

CVE-2020-0601 • Windows CryptoAPI Spoofing Vulnerability

### Was ist Curveball (CVE-2020-0601)?

**Curveball** ist eine der kritischsten Sicherheitslücken in der Windows-Geschichte. Diese Schwachstelle in der Windows CryptoAPI ermöglichte es Angreifern, die Validierung von ECC-Zertifikaten vollständig zu umgehen und sich als vertrauenswürdige Entitäten auszugeben.

#### Das Problem

Windows validierte **nicht korrekt**, ob der Generator-Punkt  $G$  in ECC-Zertifikaten dem Standard entspricht. Angreifer konnten eigene Generator-Punkte verwenden!

#### Die Mathematik

**Normal:**  $Q = d \times G$  ( $d$  unbekannt, schwer zu finden)

**Curveball:**  $Q = d' \times G'$  ( $d'$  und  $G'$  wählbar!)

#### Der Exploit

Durch Manipulation des Generator-Punkts konnten Angreifer Zertifikate erstellen, die Windows als "von vertrauenswürdigen CAs signiert" akzeptierte.

### Was Sie wissen sollten:

#### ECC Grundlagen

- Elliptische Kurven-Arithmetik
- Generator-Punkte
- Discrete Logarithm Problem

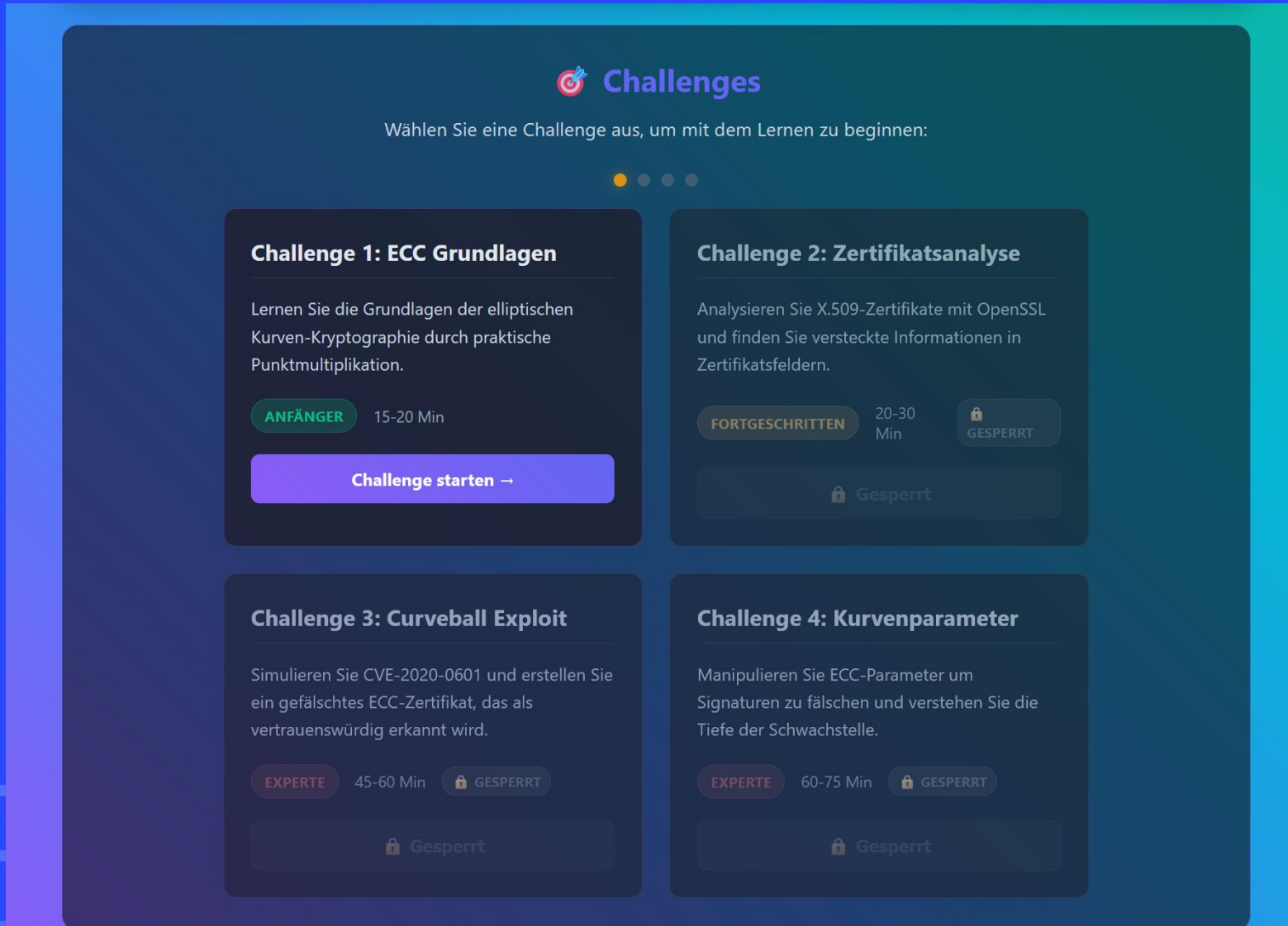
#### PKI & Zertifikate


- X.509 Zertifikat-Struktur
- Certificate Authority (CA)
- Digital Signatures

#### Technische Skills

- OpenSSL Commands
- ASN.1 Format
- Kryptographische Parameter

# Aufbau der Challenge



 **Challenges**

Wählen Sie eine Challenge aus, um mit dem Lernen zu beginnen:

● ● ● ●

### Challenge 1: ECC Grundlagen


Lernen Sie die Grundlagen der elliptischen Kurven-Kryptographie durch praktische Punktmultiplikation.


**ANFÄNGER** 15-20 Min

[Challenge starten →](#)

### Challenge 2: Zertifikatsanalyse


Analysieren Sie X.509-Zertifikate mit OpenSSL und finden Sie versteckte Informationen in Zertifikatsfeldern.


**FORTGESCHRITTEN** 20-30 Min  GESPERRT

 Gesperrt

### Challenge 3: Curveball Exploit


Simulieren Sie CVE-2020-0601 und erstellen Sie ein gefälschtes ECC-Zertifikat, das als vertrauenswürdig erkannt wird.


**EXPERTE** 45-60 Min  GESPERRT

 Gesperrt

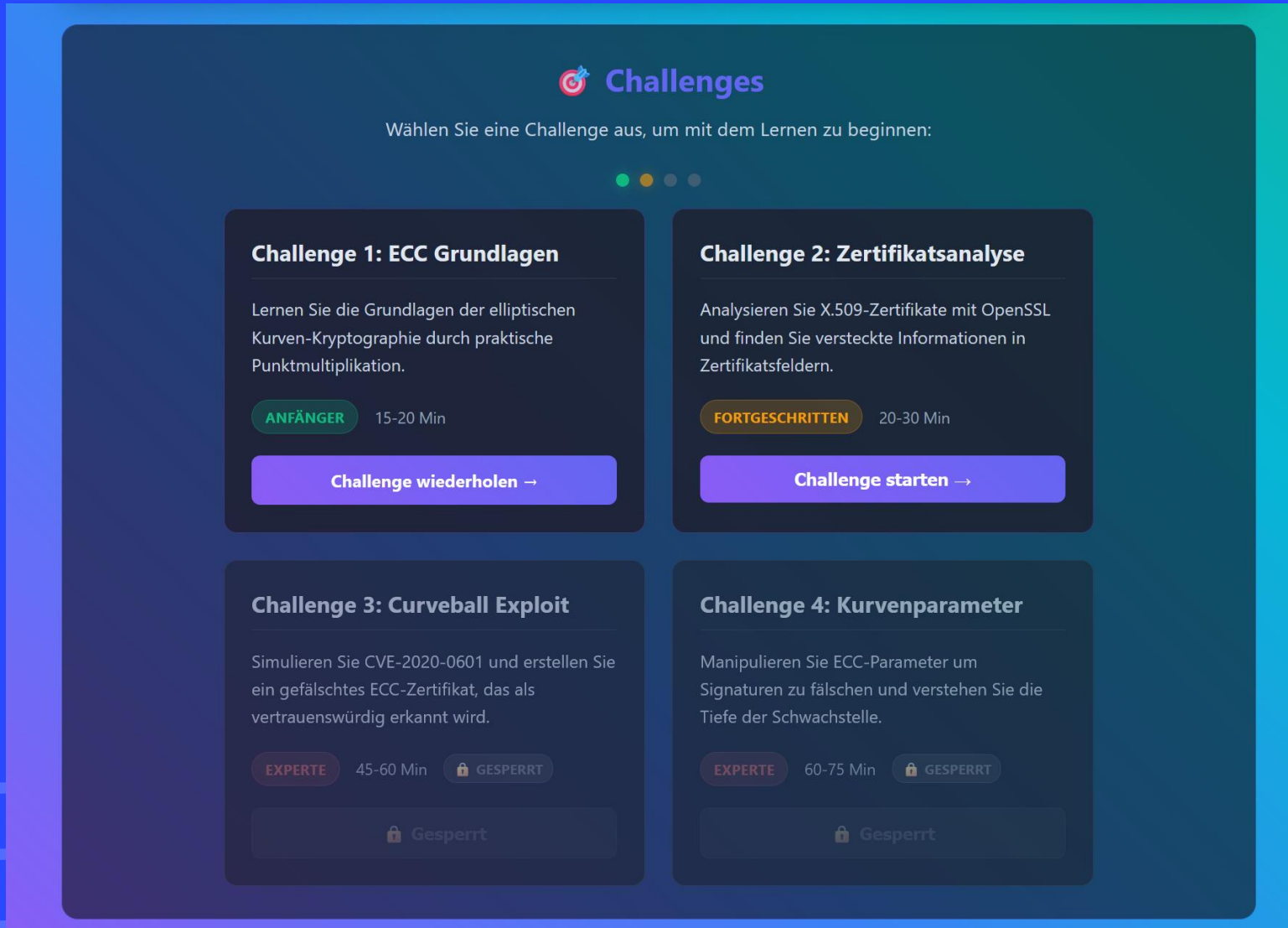
### Challenge 4: Kurvenparameter

Manipulieren Sie ECC-Parameter um Signaturen zu fälschen und verstehen Sie die Tiefe der Schwachstelle.

**EXPERTE** 60-75 Min  GESPERRT

 Gesperrt

# Aufbau der Challenge



The screenshot shows a 'Challenges' section with a dark teal background. At the top, there's a target icon and the word 'Challenges'. Below it, a prompt asks to choose a challenge. Four challenge cards are displayed in a 2x2 grid. Each card has a title, description, difficulty level in a colored pill, and duration. Challenge 1 and 2 have active buttons, while Challenge 3 and 4 are locked, indicated by a lock icon and the word 'GESPERRT'.

**Challenges**

Wählen Sie eine Challenge aus, um mit dem Lernen zu beginnen:

● ● ● ●

**Challenge 1: ECC Grundlagen**

Lernen Sie die Grundlagen der elliptischen Kurven-Kryptographie durch praktische Punktmultiplikation.

**ANFÄNGER** 15-20 Min

**Challenge wiederholen →**

**Challenge 2: Zertifikatsanalyse**

Analysieren Sie X.509-Zertifikate mit OpenSSL und finden Sie versteckte Informationen in Zertifikatsfeldern.

**FORTGESCHRITTEN** 20-30 Min

**Challenge starten →**

**Challenge 3: Curveball Exploit**

Simulieren Sie CVE-2020-0601 und erstellen Sie ein gefälschtes ECC-Zertifikat, das als vertrauenswürdig erkannt wird.

**EXPERTE** 45-60 Min **GESPERRT**

**Gesperrt**

**Challenge 4: Kurvenparameter**

Manipulieren Sie ECC-Parameter um Signaturen zu fälschen und verstehen Sie die Tiefe der Schwachstelle.

**EXPERTE** 60-75 Min **GESPERRT**

**Gesperrt**

# Aufbau der Challenge



## Challenge 4 ist gesperrt

Diese Challenge ist noch nicht freigeschaltet. Schließen Sie zunächst Challenge 3 ab, um diese Challenge zu öffnen.

### Ihr Fortschritt

Sie müssen die Challenges in der richtigen Reihenfolge abschließen:

Challenge 1: ECC Grundlagen ✓

Challenge 2: Zertifikatsanalyse ✓

Challenge 3: Curveball Exploit ⌚

Challenge 4: Kurvenparameter 🔒

← Zurück zur Übersicht

Challenge 3 bearbeiten

# Aufbau der Challenge

## Challenge 2

### Zertifikatsanalyse mit OpenSSL

Analysieren Sie ein verdächtiges X.509-Zertifikat und finden Sie versteckte Informationen



#### Die Geschichte

Sie haben ein seltsames Zertifikat gefunden. Es sieht auf den ersten Blick normal aus, aber irgendetwas daran ist... merkwürdig. Ihre Aufgabe ist es, das Zertifikat gründlich zu analysieren und herauszufinden, welche Informationen darin verborgen sind. Nicht alle Geheimnisse offenbaren sich beim ersten Blick!



**Tipp:** X.509-Zertifikate können verschiedene Felder und Extensions enthalten. Manchmal sind wichtige Informationen in unerwarteten Bereichen versteckt...



#### Zertifikat herunterladen

**mystery\_cert.pem**

Ein X.509-Zertifikat im PEM-Format zur Analyse



**mystery\_cert.pem herunterladen**



# Aufbau der Challenge

## Spezifische Felder extrahieren:

```
openssl x509 -in mystery_cert.pem -subject -noout  
openssl x509 -in mystery_cert.pem -issuer -noout
```

## Extensions und erweiterte Informationen:

```
openssl x509 -in mystery_cert.pem -text -noout | grep -A 20  
"X509v3 extensions"
```

## Alle verfügbaren Informationen durchsuchen:

```
openssl x509 -in mystery_cert.pem -text -noout | grep -i  
flag  
  
openssl x509 -in mystery_cert.pem -text -noout | grep -i  
secret
```

## 📄 Was Sie suchen sollten:

**Subject:** Informationen über den Zertifikatinhaber

**Issuer:** Die ausstellende Zertifizierungsstelle

**Extensions:** Zusätzliche Zertifikatsfelder (X509v3)

**Alternative Names:** DNS-Namen oder andere Identifikatoren

**Custom Fields:** Ungewöhnliche oder benutzerdefinierte Felder

**Comments:** Versteckte Kommentare oder Beschreibungen

## 🚩 Flag eingeben

Wenn Sie die versteckte Information gefunden haben, geben Sie die Flag hier ein:

✓ Flag prüfen

Versuche: 0

# Aufbau der Challenge

🚩 Flag eingeben

Wenn Sie die versteckte Information gefunden haben, geben Sie die Flag hier ein:

FLAG{Foo}

✓ Flag prüfen

⚠ **Maximale Anzahl von Versuchen erreicht!**

Hier sind einige Tipps zur Zertifikatsanalyse:



Verwenden Sie: `openssl x509 -in mystery_cert.pem -text -noout`

Suchen Sie in Extensions: `| grep -A 10 "X509v3 extensions"`

Durchsuchen Sie nach Flags: `| grep -i flag`

🔄 Challenge zurücksetzen

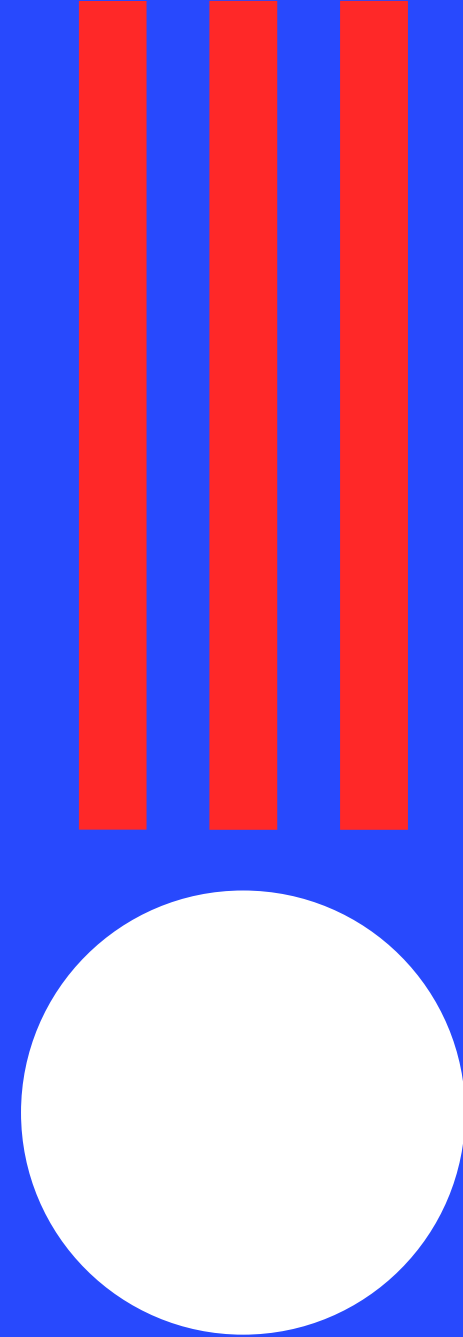
💡 Tipp: Versuchen Sie `"openssl x509 -in mystery_cert.pem -text -noout"` und durchsuchen Sie die Ausgabe.

💡 Tipp: Manchmal sind Flags in Subject Alternative Names oder anderen Extensions versteckt.

💡 Letzter Tipp: Verwenden Sie `grep`, um nach "FLAG" oder "flag" in der OpenSSL-Ausgabe zu suchen.

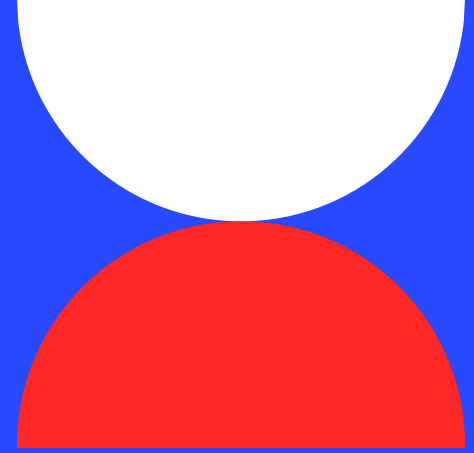


Fazit



# Fazit

- Niedrigschwelliges, plattformunabhängiges, performantes System
- Covered Grundlagendlagen bis hin zum tatsächlichen Exploit
- Umfassendes Lernmaterial bietet breites Fundament für Challenges
- Einfache Integration in bestehende Plattform
- Umfassende Projektdokumentation von Walkthrough bis technischen Hintergrund



# Fragen & Anmerkungen

