NS2 – Präsenzwochenende

Invalid Curve

Auf dem Server cloud.nds.rub.de läuft ein Server, der anfällig gegen Invalid Curve Angriffe ist. In dieser Übung soll über die Schwachstelle der private Schlüssel gestohlen werden und Application Data aus der angehängten TLS Session entschlüsselt werden.

Starten Sie zunächst den Invalid Curve Angriff mithilfe der Attacks.jar. Verwenden Sie dazu den folgenden Befehl:

```
java -jar Attacks.jar invalid_curve -connect cloud.nds.rub.de:7021 -executeAttack
```

Der Angriff erfordert ca. 4100 Anfragen und dauert ungefähr 10 - 30 Minuten. Beantworten Sie während des Angriffs die folgenden Fragen:

Frage 1

Betrachten Sie den durch den Angriff erzeugten Traffic in Wireshark. Durch welche Nachricht(en) äußert es sich, wenn das Tool einen Teil des Schlüssels lernt?

Frage 2

Wie lauten die Client und Server Randoms? Tragen Sie die Werte für die späteren Berechnungen in prf.py ein.

Frage 3

Welchen Public Key hat der Client verwendet? Wie lauten die x und y Koordinate?

Hintergrund zu Invalid Curve

Invalid Curve Angriffe machen sich zunutze, dass die generischen Double und Add Operationen, die für die Punktarithmetik auf elliptischen Kurven verwendet werden, nur einen von zwei Kurvenparametern verwenden. Die für TLS am häufigsten verwendeten elliptischen Kurven sind Weierstrasskurven der Form

$$y^2 = x^3 + ax + b.$$

Innerhalb von Double und Add wird nur der Parameter **a** benutzt. Der Parameter **b** wird implizit durch den Punkt auf dem Berechnungen durchgeführt werden bestimmt. Im Rahmen von Invalid Curve Angriffen sendet ein Angreifer Punkte, die nicht auf der vom Opfer beabsichtigten Kurve liegen, um es durch die Kontrolle über den Parameter **b** zu Berechnungen auf einer schwächeren Kurve zu verleiten. Diese Kurven werden vom Angreifer so gewählt, dass sie kleine prime Untergruppen haben.

Sendet der Angreifer einen Generator einer kleinen Untergruppe als öffentlichen DH Schlüssel an das Opfer, kann er alle möglichen Punkte vorberechnen, die das Opfer als Shared Secret berechnen könnte. Bei einer Untergruppe der Größe fünf sind es entsprechend nur fünf mögliche Punkte. Unabhängig von der Größe des privaten Schlüssels kann der Angreifer aufgrund der Eigenschaften einer Gruppe keinen anderen Punkt als einen dieser fünf berechnen. Falls der Angreifer erkennen kann, welcher dieser Punkte vom Opfer schließlich errechnet wurde, lernt er daraus einen Teil des privaten Schlüssels. Ist der Punkt Q beispielsweise Generator einer Untergruppe mit Größe fünf und der Angreifer konnte feststellen, dass das zweite Element der Untergruppe als Shared Secret berechnet wurde, lernt er die folgende Kongruenz

$$K_{priv} \quad 2 \bmod 5$$

Von dort aus kann der Angreifer weitere prime Untergruppen wählen, um mehr Kongruenzen zu lernen. Dass die Untergruppen prim sind, ermöglicht es dem Angreifer abschließend mithilfe des Chinesischen Restatzes den gesamten privaten Schlüssel aus den einzelnen Kongruenzen zu berechnen.

Um den Angriff zu verhindern, sollte geprüft werden, ob empfangene Punkte auf der beabsichtigen Kurve liegen. Alternativ gibt es auch Algorithmen für die Punktarithmetik, die für den Angriff nicht anfällig sind.

Nach dem Angriff:

Frage 4

Wie lautet das Premastersecret der Session? Verwenden Sie das Skript secp256r1.py mit dem erbeuteten privaten ECDH Schlüssel und dem öffentlichen Schlüssel des Clients.

Hinweise: Achten Sie darauf, wie bei ECDH Cipher Suites das Pemastersecret bestimmt wird.

python secp256r1.py [x] [y] [scalar]

Die Parameter können mit ‘0x‘ als Präfix auch als Hexadezimalzahl angegeben werden.

Frage 5

Wie lautet das Mastersecret? Verwenden Sie prf.py zur Berechnung.

Frage 6

Wie lautet der Keyblock? Verwenden Sie prf.py zur Berechnung.

Frage 7

Wie lautet der Inhalt der vom Client zuletzt gesendeten Application Data? Verwenden Sie aes.py mit den Schlüsseln aus dem Keyblock um die Nachricht zu entschlüsseln.