Prof. Dr. Ghassan Karame
M. Sc. Pascal Zimmer
M. Sc. Annika Wilde

Aufgabe 4

Protokollsicherheit – Praxis

Belohnung: 20 Punkte
Modus: Gruppenarbeit (max. 3 Studierende)
Abgabefrist: 27.05.2024 - 14:00 Uhr

In dieser Aufgabe werden wir uns mit der praktischen Umsetzung von Protokollangriffen beschäftigen. Beachtet, dass ihr für die Abgabe einer Gruppe in Moodle beigetreten sein müsst. Wir stellen euch Framework zur Verfügung, in dem die Protokolle implementiert sind und in dem ihr Nachrichten abfangen, verändern, speichern und verschicken könnt. Diese Aufgabe ist in zwei Teile gegliedert.

1. Framework

Die in Moodle bereitgestellten Binaries sowie die TCP-Proxy bilden das Framework für diese Aufgabe. Es handelt sich um ein TCP-Netzwerk, das vollkommen lokal und ohne Internetverbindung ausgeführt werden kann. Hierzu kommunizieren die Parteien im Netzwerk (Alice, Bob und Co.) mit einem Verteilerserver, der die Nachrichten routet. Ihr könnt euch den Server als *das Internet* vorstellen, während Alice und Co. die Clients, also die Nutzer des *Internets* sind.

Vor dem Server ist eine Proxy (d.h. eine Zwischenstation, die Nachrichten weiterleitet) geschaltet. Hier könnt ihr euch ins Netzwerk einklinken. Ihr habt also die volle Kontrolle über das gesamte Netzwerk, da jegliche Nachrichten, die ins Netzwerk gehen, über euch verteilt werden. Somit könnt ihr z.B. Pakete modifizieren oder gar nicht erst an den Server weiterleiten (*droppen*). Eine grobe Skizze eines Beispiels des bereitgestellten Frameworks findet ihr in Abbildung 1. Wenn Alice nun die Nachricht *Hallo, Bob* an Bob verschicken will, trägt sie sich selbst als Absender (*sender*), Bob als Empfänger (*receiver*) und *Hallo, Bob* als Inhalt (*content*) in ihre Nachricht ein. Diese Nachricht schickt Alice dann an die Proxy, von der Alice allerdings nichts weiß, sie kennt nur *das Internet*. Die Nachricht wird dann von euch, der Proxy, gedroppt oder (ggf. modifiziert) an den Server weitergereicht, woraufhin der Server die Nachricht dann an den *receiver* weiterleitet, sofern er sich im Netzwerk befindet. Befindet sich der *receiver* nicht im Netzwerk, wird eine Fehlermeldung ausgegeben.

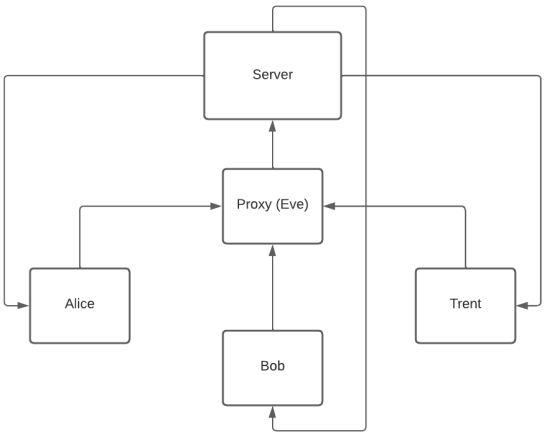


Abbildung 1: Netzwerk mit exemplarischen Parteien Alice, Bob und Trent

1.1. Starten des Frameworks

Das Framework besteht aus:

a) Der Netzwerk-Binary. Diese kann über die Kommandozeile gestartet werden und nimmt die unten genannten Flags entgegen.

```
./network --protocol=<PROTOKOLL> (--server-port=<SERVER-PORT>) (--proxy-port=<PROXY-PORT>)
```

- `protocol` gibt das Protokoll an, das durchlaufen werden soll. (siehe Teilaufgaben)
- (*Optional*, default bei `8035`) `server-port` gibt die Portnummer an, auf der der Server lauschen soll.
- (*Optional*, default bei `8034`) `proxy-port` gibt die Portnummer an, auf der die Proxy lauschen soll. Eine explizit angegebene Portnummer von `0` sorgt dafür, dass das Protokoll testweise ohne Proxy durchlaufen werden kann.

b) Der TCP-Proxy (`https://github.com/ickerwx/tcpproxy`), die benutzt werden soll. Im Ordner `tcpproxy/proxymodules` findet ihr die Datei `syssec.py`. Hier könnt ihr Pakete modifizieren oder droppen. Die Proxy wird mit dem Kommandozeilenbefehl

```
python3 tcpproxy.py -ti 127.0.0.1 -tp <SERVER-PORT> -li 127.0.0.1 -lp <PROXY-PORT> -v -om
    ↪syssec
```

gestartet werden. Wenn die Netzwerk-Binary ohne Server- und Proxy-Port-Flags gestartet wurde, lautet der Befehl, mit dem die Proxy gestartet wird also:

```
python3 tcpproxy.py -ti 127.0.0.1 -tp 8035 -li 127.0.0.1 -lp 8034 -v -om syssec
```

Beachtet, dass die Proxy vor dem Netzwerk gestartet werden muss, damit das Zusammenspiel der Ports korrekt funktioniert.

1.2. Proxy-Module

In der Datei `syssec.py` findet ihr eine Beispielimplementierung zum droppen und modifizieren von Paketen. In der Funktion `execute` nehmt ihr die Daten (`data`) auf. In der Funktion könnt ihr sie dann bearbeiten und zurückgeben. Dementsprechend sieht das Droppen von Paketen mit der `id` von `1` so aus:

```
if data_json["id"] == 1:
        return bytes()
```

Und das Reflektieren eines Pakets mit `id=2` wie folgt:

```
if data_json["id"] == 2:
    orginal_sender, orginal_receiver = data_json["sender"], data_json["receiver"]
    data_json["sender"], data_json["receiver"] = orginal_receiver, orginal_sender
    data = json.dumps(data_json)
    return data + "\n"
```

Hierbei ist das anfügen des Delimiters `\n` zu beachten. Wird dieser vergessen, wird eure Nachricht vom Server nicht als solche erkannt.

Hinweise

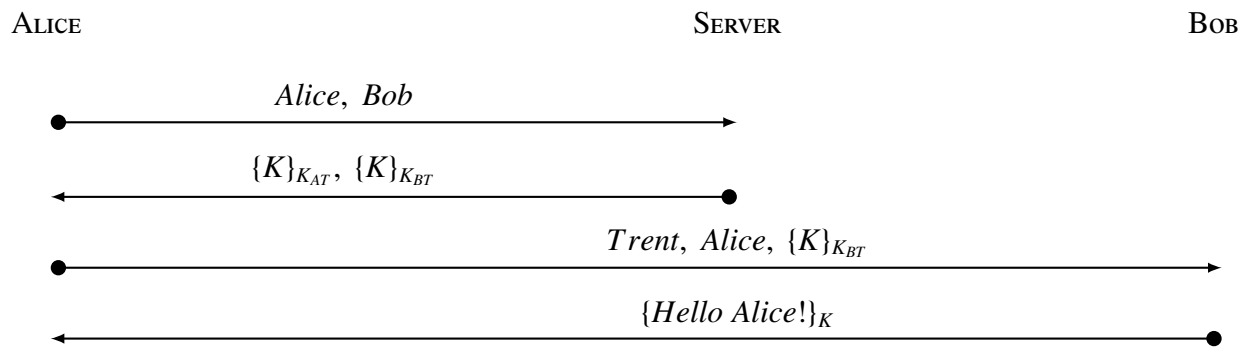
- Eine Flag garantiert nicht die korrekte Lösung der Aufgabe. Überlegt euch, ob der gewählte Angriff auf das Protokoll in der Theorie sinnvoll ist. Wenn er das ist **und** euch eine Flag angezeigt wird, wurde vermutlich die richtige Lösung gefunden. Die Flag ist daher nur ein Indikator. Eine Extraktion der Flag aus der Binary hilft euch beim Lösen der Aufgabe daher **nicht**.
- Es ist **nicht** erforderlich, einen eigenen Client ins Netzwerk zu bringen. Alle Aufgaben können ausschließlich mit der Paketmodifikation bzw. dem Droppen von Paketen, wie oben dargestellt, gelöst werden.
- Es ist **kein** DoS-Angriff (Denial of Service) notwendig.
- Es ist **keine** Modifikation von Handshakes notwendig. Die Handshakes dienen zur Registrierung einer Partei im Netzwerk und dürfen nicht verändert werden.

2. Aufgaben

In den folgenden Aufgaben werdet ihr Angriffe auf verschiedene Protokolle finden und im Framework implementieren. Das Framework stellt Implementierungen aller Protokolle bereit.

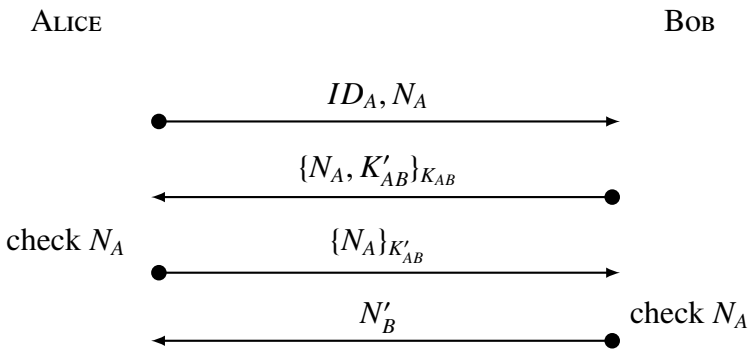
2.1. Warm Up

Dieser Teil der Aufgabe dient dazu, euch mit dem Framework vertraut zu machen. Dafür werdet ihr zunächst den Angriff auf ein einfaches Protokoll implementieren. Das gegebene Protokoll dient dazu, einen Sitzungsschlüssel zwischen Alice und Bob auszuhandeln. Der Sitzungsschlüssel wird von einem Server gewählt, dem Alice und Bob beide vertrauen und mit dem beide einen symmetrischen Langzeitschlüssel haben. Ziel des Angreifers ist es, sich Alice gegenüber als Bob auszugeben. Beachtet hierbei, dass ihr selbst nicht Ver- oder Entschlüsseln müsst, aber dennoch ein korrekter Sitzungsschlüssel ausgehandelt werden soll, damit Ver- oder Entschlüsselt werden könnte. Sendet als Angreifer anstatt des verschlüsselten "Hello Alice!" arbiträre Daten an Alice zurück. Das Protokoll kann mit dem Befehl `./network -protocol=mitm` ausgeführt werden.



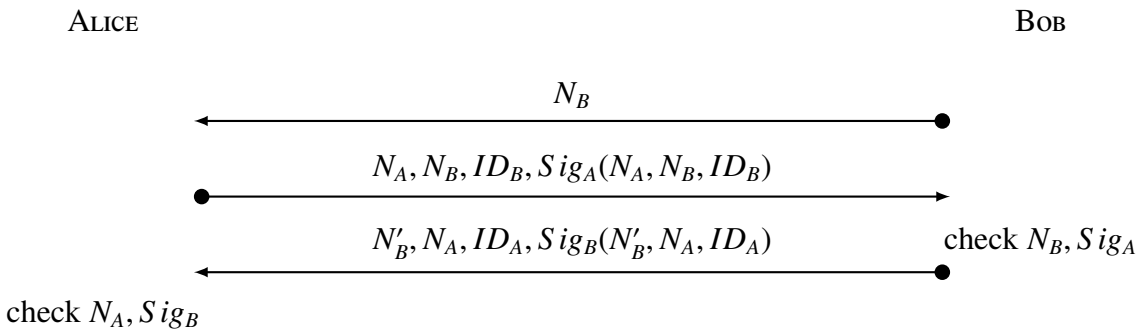
2.2. Key Establishment

Dieses Protokoll dient dazu einen Sitzungsschlüssel K'_{AB} zwischen zwei Parteien ohne die Interaktion mit einem Server zu etablieren. Hierbei ist der Schlüssel K_{AB} ein Langzeitschlüssel zwischen Alice und Bob. Eure Aufgabe ist es einen Angriff zu entwickeln, mit dem sich der Angreifer Alice gegenüber als Bob ausgeben kann. Das Protokoll kann mit dem Befehl `./network -protocol=slke` ausgeführt werden.



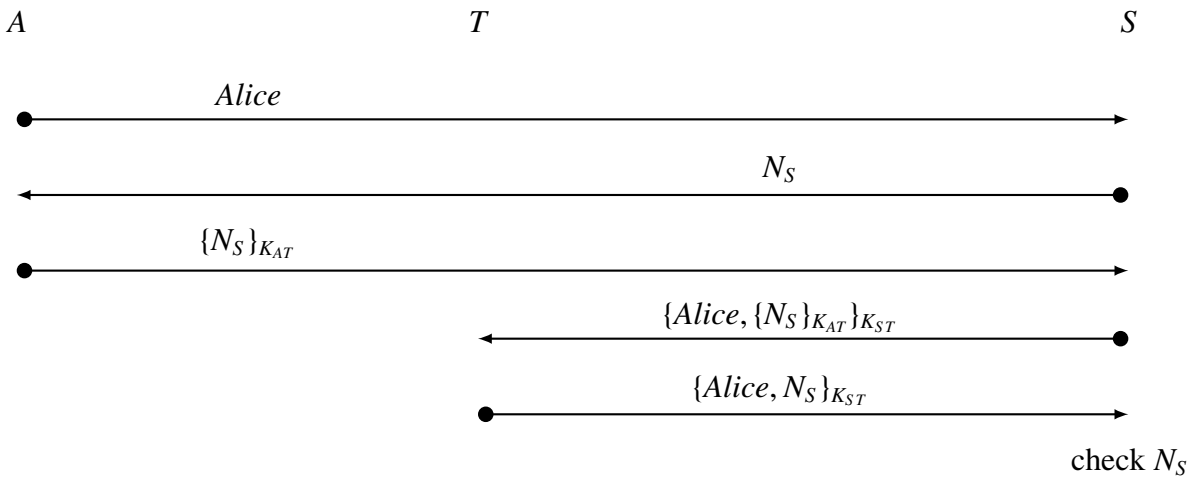
2.3. Authentication without Trusted Party

Mit Hilfe dieses Protokolls können sich zwei Parteien Alice und Bob gegeneinander authentifizieren. Hierbei ist `Sig` ein beliebiges Signaturschema. Beide Parteien überprüfen die Nonces und die betreffenden Signaturen mit Hilfe von öffentlichen Schlüsseln. Eure Aufgabe ist es einen Angriff zu entwickeln, mit dem sich der Angreifer Alice gegenüber als Bob ausgeben kann. Das Protokoll kann mit dem Befehl `./network -protocol=sla` ausgeführt werden.



2.4. Authentication with Trusted Party

In diesem Szenario nutzt ein Client A seinen Provider T um das Netzwerk eines Serviceanbieters S zu besuchen. Mithilfe des Protokolls authentifiziert sich A gegenüber S über T als trusted third party. A und S vertrauen einander. Beide Parteien teilen einen symmetrischen Schlüssel mit dem Server (K_{AT} für A und K_{ST} für S). Ziel des Angreifers ist es, sich S gegenüber als A auszugeben. Hierbei soll der Angreifer mehr als nur ein einfaches Relay sein. Das Protokoll kann mit dem Befehl `./network -protocol=authentication` ausgeführt werden.



Deliverables

Für die Abgabe gilt allgemein:

- Die Abgabe erfolgt innerhalb **einer** PDF-Datei, vorzugsweise in LaTeX mit der verfügbaren Lösungsvorlage. Es ist aber auch möglich eine PDF mit Word (oder ähnlich) zu erstellen und dort die Protokollabläufe einzuzeichnen.
- **Handschriftliche Abgaben werden nicht akzeptiert.**

Für die volle Punktzahl sind folgende Aspekte für eine Lösung erforderlich:

- Schematischer Protokollablauf mit Angreifer und markierten Änderungen, die dieser durchgeführt hat.
- Eine Beschreibung, die beschreibt, warum ein Angriff möglich ist, wie dieser durchgeführt wird und was das Ergebnis für den Angreifer ist.
- Außerdem ist für jedes der Protokolle eine Python-Datei mit eurem ausreichend und sinnvoll kommentierten Python-Code einzureichen. Der Code muss direkt mit der bereitgestellten Version von `tcpproxy` unter Python 3.10 ausführbar sein.

Insgesamt sind also 5 Dateien abzugeben (4 × Code und 1 × PDF). Um euch das Zeichnen der Protokollabläufe mit LaTeX zu erleichtern, haben wir euch den entsprechenden Code **in Moodle** zur Verfügung gestellt.

Plagiate[1]

ACM definiert Plagiat als die fälschliche Darstellung von Schriften, Ideen oder anderen kreativen Arbeiten einer anderen Person (einschließlich unveröffentlichter und veröffentlichter Dokumente, Daten, Forschungsvorschläge, Computercode oder anderer Formen des kreativen Ausdrucks, einschließlich elektronischer Versionen) als die eigene Arbeit. Ein Plagiat ist ein klarer Verstoß gegen die ACM-Publikationspolitik und ein möglicher Verstoß gegen den ACM-Ethikkodex. Plagiate können auch eine Verletzung des Urheberrechts darstellen. Plagiate äußern sich in einer Vielzahl von Formen, darunter:

- wortwörtliches Kopieren, nahezu wortwörtliches Kopieren oder absichtliches Paraphrasieren von Teilen der Arbeit eines anderen;
- die Verwendung automatisierter Tools, die vorhandene Arbeiten als eigenen Text umformulieren, ohne dass eine angemessene Namensnennung erfolgt;
- Kopieren von Elementen einer fremden Arbeit, wie Gleichungen, Tabellen, Diagramme, Illustrationen, Darstellungen oder Fotos, die nicht allgemein bekannt sind, oder Kopieren oder absichtliches Paraphrasieren von Sätzen ohne ordnungsgemäße oder vollständige Quellenangabe;
- wortwörtliches Kopieren von Teilen einer fremden Arbeit mit falscher Quellenangabe

Bei der Verwendung von Stack-Overflow oder ähnlich, bitten wir demnach um Angabe der Quelle von Code-Bausteinen, die nicht selbstständig geschrieben worden sind.

[1] Übersetzung von `https://www.acm.org/publications/policies/plagiarism-overview` mit DeepL.