IT- und Netzwerksicherheit
WiSe 2016/17
Praktisches Übungsblatt Nr. 2

Nils Aschenbruck

Jan Bauer, Alexander Bothe, Florian Krampe, Thomas Hänel,

Timmy Schüller, Bertram Schütz, Matthias Schwamborn

Constantin Schraeder

Veröffentlichung	14.11.2016
Abgabe	20.11.2016

Allgemeine Informationen zu den praktischen Aufgaben:

- Lesen Sie sich das Aufgabenblatt **aufmerksam von vorne bis hinten** durch, bevor Sie mit der Bearbeitung beginnen!
- Die erfolgreiche Bearbeitung von 4 der 6 praktischen Aufgaben ist für die Zulassung zur Prüfung erforderlich.
- Alle praktischen Aufgaben müssen **alleine** bearbeitet werden.
- **Code-Plagiate werden nicht toleriert!** Gleiches gilt, falls Code von anderen Quellen ohne Referenz oder ungewöhnlich viel Code (auch mit Referenz) übernommen wird. Dies betrifft alle beteiligten Teilnehmer.
- Die Abgabe der Aufgabe muss am jeweiligen Abgabetermin **bis spätestens 23:59 Uhr** erfolgt sein. Im allgemeinen Fall muss dies per E-Mail an die folgende Adresse

  sys-lehre@uos.de

  geschehen. Zugehörige Dateien sind in ein ZIP-Archiv zu packen und als Anhang beizufügen. Der Betreff der E-Mail und der Name der angehängten Datei müssen nach dem folgenden Schema gewählt sein:

  ITS_WiSe_201617_PA2_*MatrNr*

  wobei ***MatrNr*** durch die jeweilige Matrikelnummer ersetzt werden muss. Beim Entpacken des Archivs muss automatisch ein **Unterordner nach dem gleichen Namensschema** erzeugt werden. Beachten Sie mögliche Änderungen der Abgabemodalitäten bei einzelnen Aufgaben.
- **Befolgen Sie alle aufgeführten Konventionen** für Dateinamen, Ordnerstruktur, Programmparameter, etc. Ansonsten verlängern Sie unnötig die Korrekturdauer und verzögern somit die Ergebnisverkündung.
- Bei Fragen von allgemeinem Interesse nutzen Sie bitte die **Mailingliste** zur Vorlesung. Alternativ können Sie auch in die wöchentliche **Tutoren-Fragestunde** kommen, wobei vorab eine kurze Problembeschreibung per E-Mail an den Tutor (**cschraed@uos.de**) erfolgen sollte.

Praktische Aufgabe 2: IoT Sniffing

Das Message Queue Telemetry Transport (MQTT) Protokoll [1] ist ein leichtgewichtiges open-source Client Server Nachrichtenprotokoll, das für den Einsatz auf ressourcenbeschränkten Geräten in Netzwerken mit geringer Bandbreite entwickelt wurde. Seit 2013 wird MQTT daher von der OASIS als Protokoll für das Internet of Things (IoT) [2] standardisiert. MQTT basiert auf einer Publish-Subscribe Architektur und ermöglicht Machine-to-Machine (M2M) Kommunikation über TCP/IP in vielen Bereichen des IoTs, von Industrie 4.0 bis zu Smart Homes. Geräte können als MQTT-Publisher Nachrichten an einen Broker übertragen und als Subscriber empfangen. Jeder Nachricht wird dazu vom Publisher ein bestimmtes Topic zugewiesen, das dann (mehrere) Subscriber abonnieren können. Mit der Version 3.1 wurden erstmals Security-Features integriert. Zum einen wird SSL/TLS [3] unterstützt und zum anderen gibt es eine Benutzerverwaltung zur Authorisierung. Diese erlaubt es, einzelnen Benutzern, die sich mit Passwörtern authentisieren müssen, über Access Control Lists (ACLs) [4] Schreib-/Leserechte auf einzelne Topics vergeben zu können. Allerdings wird der Benutzername und das dazugehörige Passwort beim MQTT-Verbindungsaufbau im Klartext übertragen (siehe [5]), falls TLS nicht genutzt wird. Da TLS standardmäßig deaktiviert ist und zudem das Generieren und Verteilen von Zertifikaten erfordert, wird diese Sicherheitsoption in der Praxis häufig vernachlässigt.

Szenario

Stellen Sie sich vor, das Facility-Management der Universität hätte die Hörsaal-Steuerung mittels MQTT automatisiert. Nun ließe sich beispielsweise der Beamer, der als Subscriber auf dem Topic `/uos/<gebauede>/<raum>/beamer-control` auf Steuerungsbefehle wartet, zu Beginn einer Veranstaltung automatisiert mit der Nachricht `beamer_on` einschalten. Abgeschaltet würde der Beamer automatisch zum Ende der Veranstaltung durch den Befehl `beamer_off`. Wenn das Facility-Management jedoch versäumt hat, TLS zu aktivieren, könnte das jedoch auch schon während der Veranstaltung passieren.

Aufgabe

Versuchen Sie die Sicherheitslücke im oben skizzierten Szenario auszunutzen und den Benutzernamen samt Passwort beim MQTT-Verbindungsaufbau abzuhören und damit über eine gefälschte Nachricht den Beamer auszuschalten. Schreiben Sie dazu den Angreifer als C-Programm *sniffer*, der die MQTT-Kommunikation in der bereitgestellten Testumgebung (s.u.) abhört und bei einem Verbindungsaufbau der Beamersteuerung auf das Topic `uos/93/E06/beamer-control` die ungeschützten Zugangsdaten (*user* und *password*) übernimmt. Mit dieser Information verschickt der Angreifer daraufhin eine

gefälschte `beamer_off` Nachricht über das gleiche Topic an den Beamer. Dazu kann der Aufruf des Publisher-Clients genutzt werden:

```
mosquitto_pub -m "beamer_off" -t "/uos/93/E06/beamer-control" -u <user>
                                -P <password>
```

Nach dem Versenden der Nachricht soll das Programm terminieren, sich aber auch beim vergeblichen Sniffen durch *strg + c* ohne Memory Leaks beenden lassen.

Das Format der CONNECT-Nachricht, die beim MQTT-Verbindungsaufbau zwischen Client und Broker ausgetauscht wird, finden Sie unter [5]. Gehen Sie der Einfachheit halber davon aus, dass der Client beim Aufbau wie in der Testumgebung das *Will Flag* und die *Clean Session* Option nicht gesetzt hat. Zudem könnte die zur Verfügung gestellte *PA2.h* Datei nützlich sein, um den Zugriff auf die TCP-Nutzdaten zu vereinfachen.

Testumgebung

Installieren Sie den Mosquitto MQTT Broker und die zugehörigen Clients (*mosquitto-clients*) in Ihrer VM. Außerdem benötigen Sie folgende Pakete: *python*, *sudo* und *linux-headers*. Entpacken Sie das Archiv *PABlatt02_files.tar.gz* von der Website und führen Sie das darin enthaltene Python-Script aus:

```
python run_mqtt.py
```

Das Script generiert zu Beginn zufällige Passwörter für beide Benutzer in dem Szenario, den Beamer *beamer* und die Beamersteuerung *remote-control*. Daraufhin wird eine entsprechenden Passwortdatei und eine ACL erstellt. Nun wird der Mosquitto Broker ausgeführt und die beiden Dateien, wie in der *mosquitto.conf* definiert, eingelesen. Danach wird ein Subscriber gestartet, hier der fiktive Beamer, der auf dem Topic `/uos/93/E06/beamer-control` auf Befehle lauscht. Jetzt versucht das Skript, durch `sudo ./sniffer` Ihren Angreifer auszuführen, der jede Kommunikation zum Broker abhört. Anschließend wird die Beamersteuerung als Publisher gestartet und eine reguläre `beamer_on` Nachricht übertragen. Bei erfolgreichem Angriff erscheint die `beamer_off` Nachricht unter der `beamer_on` Nachricht beim abschließenden Bericht auf der Kommandozeile.

Kompilieren vereinfachen

Schreiben Sie auch für diese Aufgabe wieder ein einfaches Makefile, um das Kompilieren Ihres C-Programm zu vereinfachen, welches das Programm per `default` Target kompiliert. Zudem sollte Ihr Makefile das Target `clean` unterstützen, d.h. durch den Befehl `make clean` sollen Sie Ihr Projekt bereinigen können.

Allgemeine Funktionsprüfung

Stellen Sie auch sicher, dass Ihr Programm **keine Memory Leaks erzeugt**. Um Ihr Programm auf Speicherfehler zu testen, können Sie das Programm *valgrind* [6] verwenden. Um einige systeminhärente Warnmeldungen zu unterdrücken, können Sie wieder die in Aufgabe 1 zur Verfügung gestellte Valgrind-Suppression-Datei (*alpine.supp*) nutzen.

Vorsicht! Wenn Sie diese Anforderungen nicht einhalten, kann dies zum Nichtbestehen der Aufgabe führen.

Happy Coding!

Abgabe:

- ZIP-Archiv (*ITS_WiSe_201617_PA2_MatrNr.zip*), das beim Entpacken (z.B. *unzip*) automatisch den Ordner *ITS_WiSe_201617_PA2_MatrNr* anlegt (s. Allg. Informationen auf Seite 1).
- In diesem Ordner muss sich das vollständige Programm befinden, das sich (mit Ihrem Makefile durch Ausführen von make) kompilieren lässt, d.h.:
  - alle Quelldateien (.c) [+ Headerdateien (.h)]
  - Makefile, das *sniffer* erstellt.

Abgabevariante:

- Standard E-Mail (Betreff und Anhang gemäß Allg. Informationen)

Quellenangaben

[1] `http://mqtt.org`

[2] `https://en.wikipedia.org/wiki/Internet_of_things`

[3] `https://en.wikipedia.org/wiki/Transport_Layer_Security`

[4] `https://en.wikipedia.org/wiki/Access_control_list`

[5] `http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html#_Toc398718028`

[6] `http://valgrind.org/`