IT- und Netzwerksicherheit
WiSe 2016/17

Praktisches Übungsblatt Nr. 4

Nils Aschenbruck

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

Timmy Schüller, **Bertram Schütz**, Matthias Schwamborn, Stefanie Thieme

Constantin Schraeder

Veröffentlichung	12.12.2016
Abgabe	20.12.2016

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 via **Stud.IP** geschehen. Zugehörige Dateien sind in ein ZIP Archiv zu packen und im Aufgabenbereich der Veranstaltung hochzuladen. Der Name der Datei muss nach folgendem Schema gewählt sein:

  ITS_WiSe_201617_PA4_*MatrNr*

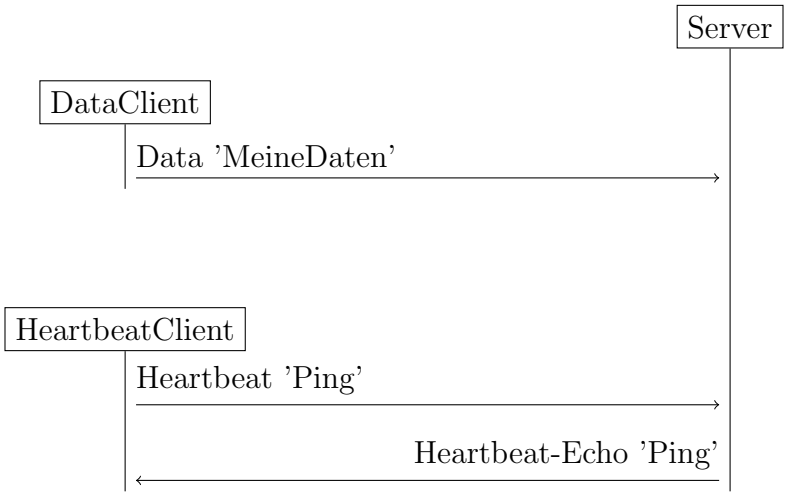
  wobei ***MatrNr*** durch die jeweilige Matrikelnummer ersetzt werden muss. Beim Entpacken des Archivs muss automatisch ein **Unterordner nach dem gleichen Namensschema** erzeugt werden. Markieren Sie nach dem Upload die jeweilige Aufgabe als fertig. Beachten Sie mögliche Änderungen der Abgabemodalitäten 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 4: Buffer-Underflow

In dieser Aufgabe sollen Sie einen Buffer-Underflow-Angriff implementieren. Auf der Veranstaltungswebseite finden Sie dazu ein einfaches Projekt, welches die Grundlagen der Heartbleed-Sicherheitslücke nachstellt. Informieren Sie sich zunächst über Heartbleed, so dass Sie verstehen, wie die Sicherheitslücke auftritt, und wie diese von einem Angreifer ausgenutzt werden kann. Sinnvolle Informationen finden Sie in den Quellen [1] und [2].

Das Beispielprojekt

Laden Sie sich zunächst das Beispielprojekt von der Veranstaltungswebseite und machen Sie sich mit den Programmen vertraut. **Betrachten Sie zunächst die bereits implementierten Referenzprogramme.** Führen Sie das mitgelieferte Makefile aus und kompilieren Sie so die Referenzprogramme. Es ist für diese Aufgabe dringend ratsam, dass Sie eine funktionierende Alpine-Linux Maschine (entsprechend in der Vorbereitungsaufgabe beschrieben) verwenden.
Starten Sie zunächst das *Server*-Programm in einem Terminal. Führen Sie in einem zweiten Terminal das *DataClient*-Programm aus, welches Beispiel-Daten an den Server schickt. Sobald der *DataClient* terminiert ist, starten Sie den *HeartbeatClient*. Dieser schickt eine reguläre Heartbeat-Anfrage und erhält die korrekte Echo-Antwort vom Server. Der hier beschriebene Ablauf, ohne Ausnutzung der Sicherheitslücke, ist in der nachfolgenden Abbildung dargestellt.



Angriff per Heartbleed-Client

Betrachten Sie nun den modifizierbaren Quellcode des Projektes. **Erweitern Sie die Methode *serializeHeartbeatMessage()* im *HeartbeatClient***, so dass Ihr modifizierter HeartbeatClient, statt einer regulären Heartbeat-Anfrage, einen Heartbleed-Angriff an den Server schickt. Beachten Sie dazu, welche Struktur und welchen Inhalt die bösartige Heartbeat-Nachricht des Client haben muss.
Kompilieren Sie Ihren modifizierten *HeartbeatClient* per *make*.
Testen Sie Ihren modifizierten Client, in dem Sie den Ablauf aus der obrigen Abbildung erneut nachstellen. Verwenden Sie dabei nun Ihren modifizierten *HeartbeatClient*, den Referenz-*DataClient* und den Referenz-*Server*. Ihr modifizierter *HeartbeatClient* soll, unter Ausnutzung der Heartbleed-Sicherheitslücke, die Beispiel-Daten im Referenz-*Server*-Buffer auslesen (als Inhalt der Echo-Nachricht), welche zuvor vom Referenz-*DataClient* dorthin geschickt wurden.

Sicherheitslücke im Server schließen

Schließen Sie die Sicherheitslücke im Server, so dass eine Heartbleed-Anfrage erkannt und nicht beantwortet wird. **Modifizieren Sie dazu die Methode *deserializeMessage()* im *Server*-Programm.** Geben Sie eine Meldung auf der Konsole aus, sobald Ihr modifizerter Server eine bösartige Heartbeat-Anfrage erkennt. Reguläre Heartbeat-Anfragen sollen jedoch weiterhin von dem Server korrekt beantwortet werden. Beachten Sie dazu, wie eine reguläre Heartbeat-Anfrage aussehen sollte und was diese von einer bösartigen Heartbleed-Anfrage unterscheidet. Testen Sie Ihren Server, so dass dieser sowohl Anfragen des Referenz-*HeartbeatClients*, als auch Ihres modifizierten Clients korrekt beantwortet.

Kompilieren vereinfachen

Um das Kompilieren eines Projekts zu vereinfachen, kann das Programm *make* verwendet werden. Schreiben Sie ein einfaches Makefile, welches ihre Programme 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 sicher, dass Ihre Programme **keine Memory Leaks erzeugen**. Um Ihre Programme auf Speicherfehler zu testen, können Sie das Programm *valgrind* [3] verwenden. Um einige systeminhärente Warnmeldungen zu unterdrücken, können Sie die 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_PA4_MatrNr.zip*), das beim Entpacken (z.B. *unzip*) automatisch den Ordner *ITS_WiSe_201617_PA4_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

Abgabevariante:

- Upload des ZIP-Archives in **Stud.IP** (Unter *IT- und Netzwerksicherheit* → *Aufgaben* → *PA4* die Datei hinzufügen und anschließend links im Aktionsmenü *Aufgabe als fertig markieren*)

Quellenangaben

[1] `https://de.wikipedia.org/wiki/Heartbleed`

[2] `http://heartbleed.com/`

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