Fakultät Informatik
Institut für Systemarchitektur, Lehrstuhl Rechnernetze

Implementierung einer Testsuite zur Untersuchung von Möglichkeiten für DoS-Angriffe auf DTN-Protokollimplementierungen

Tobias Nöthlich
Florian Richter
Tim Krieg

Projektbericht

Betreuer
Dr.-Ing. Marius Feldman

Wintesemester 2020/2021

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation

In Katastrophengebieten oder Verzögerungstolerante Netzwerke (engl. Delay-Tolerant Networking, kurz DTN) [PLATZHALTER] Generelles Intro über Nutzen von DTN in Raumfahrt, Katastrophenhilfe, etc

[PLATZHALTER] Kurz erklären wie das ganze funktioniert

[PLATZHALTER]Überleiten auf Angriffsvektoren → unsere Testsuite kommt ins Spiel.

1.2 Anforderungen

- Anforderungsanalyse

2 Hintergrund und verwandte Arbeiten

2.1 DTN-Protokollimplementierungen

[PLATZHALTER] Generelle **kurze** Einführung in die verschiedenen DTN-Implementierungen.

2.2 ION

[PLATZHALTER] Kurze Erläuterung der ION DTN Implementation.

2.3 CORE

[PLATZHALTER] kurze Erläuterung zu CORE.

3 Methodik

- Erklärungen warum wir genau diese Frameworks gewählt haben
- kurze Erläuterungen zu den Szenarien die wir uns ausgedacht haben

4 Implementation

4.1 Shell Skripte

4.1.1 Setup

4.1.2 Start

4.1.3 ...

4.2 Verknüpfung von ION und CORE

5 Szenarien

5.1 Bundle Flooding

Das erste Szenario ist die Simulation eines Bundle Flooding Angriffs auf ein Netzwerkelement, das als Verbindungsstück zwischen einem Satelliten und dem Mission Control Center fungiert. Die für die Simulation genutzte Netzwerktopologie (siehe Abb. [PLATZHALTER]) besteht aus fünf Elementen, welche im Common Open Research Emulator durch den **Router** Knotentypen emuliert werden.

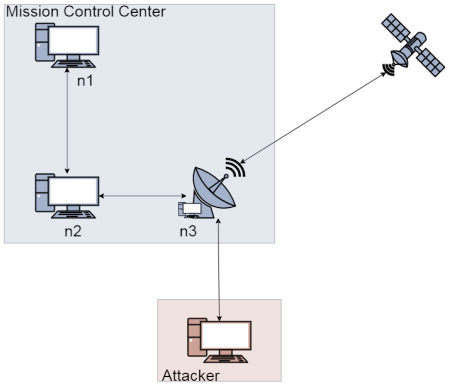

Abbildung 1: Netzwerktopologie des Bundle Flooding Szenarios

Das Grundprinzip des dem Szenario zugrunde liegenden Denial of Service Angriffs ist das Fluten des Zielsystems mit einer so großen Menge an Bundles, dass es nicht mehr in der Lage ist den legitimen Traffic zuzustellen. Da Delay-Tolerant Networking Systeme (DTN-Systeme) insbesondere für Gebiete, in denen stabile Verbindungen nicht durchgehend möglich sind, konzipiert wurden, gibt es auch in diesem Szenario Verbindungsunterbrechungen zwischen Satellit und dem Mission Control Center. Da durch unterbrochene Verbindungen nicht zustellbare Bundles zunächst auf dem letzten erreichbaren Knoten gesichert werden, muss ein Angreifer lediglich einen Knoten mit instabiler Verbindung attackieren. Es ist allerdings von Nöten, dass der Angreifer Zugriff auf

ein in das ION-Netz eingebundenes System hat, da ION die verfügbaren Kontakte aus einer Konfigurationsdatei liest. Es ist also nicht möglich, von ausserhalb eine Verbindung zu einem bestehenden Netzwerk herzustellen.

Konkret auf unser Szenario bezogen bedeutet dies, dass der *Angreifer* ein System im Netzwerk übernommen hat, und nun eine Verbindung zu *n3* besteht. Er wird nun diesen Knoten mit an den *Satelliten* adressierten Bundles fluten, bis der restliche Netzwerkverkehr zum Erliegen kommt. Dabei reichen 50 Bundles pro Sekunde aus, um in kurzer Zeit die Übertragungskapazität der Leitung zwischen *n3* und dem *Satelliten* so zu verschlechtern, dass nur knapp ein Drittel der ursprünglichen Leistung für den restlichen Traffic zur Verfügung steht. Dies zeigt sich an den Round Trip Times (RTT) der Kommunikation zwischen *n1* und dem *Satelliten*. Hat ein Bundle Ping von *n1* zum *Satelliten* vor dem Start des DoS-Angriffs eine RTT von knapp einer Sekunde, so beträgt diese wenige Sekunden nach Start des DoS-Angriffs bereits über 3 Sekunden. Circa 5 Sekunden nachdem der *Angreifer* beginnt *n3* zu fluten, ist ein Ping von *n1* zum *Satelliten* nicht mehr möglich, da bereits über 100 Bundles vom *Angreifer* auf dem Knoten *n3* auf die Weiterleitung zum *Satelliten* warten.

Ein wichtiger Faktor zum Erfolg dieses Angriffes, ist die von ION gegebene Möglichkeit, die Priorität der vom Bundle Ping ausgesendeten Bundles zu ändern. Durch das Versenden von Bundles mit der höchsten Priorität ist es dem *Angreifer* möglich, die von ihm gesendeten Bundles an den Anfang der Queue zu setzen, die die Reihenfolge der weitergeleiteten Bundles festlegt. Des Weiteren sorgt die instabile Verbindung zwischen *n3* und dem *Satelliten*, welche alle 30 Sekunden für 30 Sekunden unterbrochen wird, für ein rapides Ansteigen der auf *n3* zwischengelagerten Bundles. Die so von nicht legitimen Bundles circa 50 zu 1 dominierte Weiterleitungsqueue kann nicht mehr komplett geleert werden, was nach entsprechender Dauer zum Timeout der von *n1* kommenden Bundles führt.

Zur Visualisierung der Attacke dienen zwei Shell-Skripte, welche die Anzahl der auf jedem Knoten eingelagerten Bundles zeigen, sowie ein Skript, das den Ping von *n1* zum *Satelliten* visualisiert. Dieses Skript nutzt dabei zur besseren Übersicht die von ION zu Verfügung gestellten watch characters. So wird für jedes versendete Bundle einmal das Zeichen 'a' ausgegeben. Erreicht das auf so ausgelöste Acknowledgement *n1*, wird desweiteren die RTT angezeigt. Die zur Visualisierung benötigten Skripte werden automatisch mit Start des Szenarios ausgeführt.

5.2 DDoS Flooding

Eine Weiterentwicklung des ersten Szenarios ist die Simulation eines Bundle Flooding Angriffs auf ein Netzwerkelement, das als Verbindungsstück zwischen einem Satelliten und dem Mission Control Center fungiert. Die für die Simulation genutzte Netzwerktopologie (siehe Abb. [PLATZHALTER]) besteht aus fünf Elementen, welche im Common Open Research Emulator durch den **Router** Knotentypen emuliert werden.

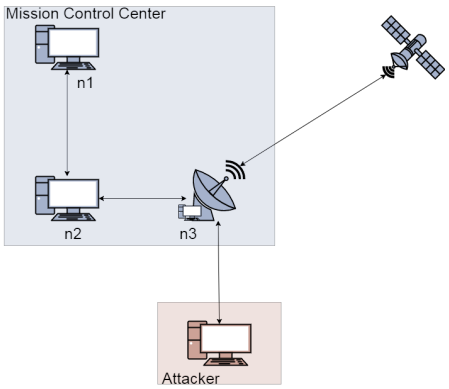

Abbildung 2: Netzwerktopologie des Bundle Flooding Szenarios

5.3 Slowloris

Im Gegensatz zu den bisher diskutierten Szenarios benötigen Application Layer Denial of Service Angriffe kein bereits kompromittiertes System um zu funktionieren. Das in diesem Abschnitt näher betrachtete Szenario setzt dabei einen Slowloris Angriff erfolgreich um. Wir nutzen in dem Szenario slowloris.py (Version 0.2.2; Yaltirakli, G., 2015)[1], ein Python Skript, welches die Attacke auf das Zielsystem ausführt.

Die hier simulierte Situation ist ein Angriff auf ein Katastrophenfrühwarnsystem. Die Netzwerktopologie (siehe Abb. [Platzhalter]) ist die Folgende:

- eine Messstation
- ein Satellit für die Datenübermittlung
- ein Kontrollzentrum
- ein Emergency Broadcast System
- ein Angreifer, der die Verbindung zwischen Messstation und Satellit stört

Dabei sind die Systeme des Frühwarnsystems in Reihe geschaltet und zusätzlich der Angreifer mit dem Satelliten verbunden. Als Transportprotokoll kommt TCP zum Einsatz. Wie auch in den vorherigen Szenarien werden die einzelnen Knoten durch CORE Routerknoten emuliert.

Das Ziel des Denial of Service Angriffs ist das Lahmlegen des Zielsystems unter minimaler Verwendung von Netzwerkressourcen. Dies wird erreicht, indem das Angreifersystem möglichst viele Verbindungen zum Zielsystem aufbaut und diese so lange wie möglich offenhält.

In diesem Szenario wird dieser Effekt durch das Ausführen von dem slowloris.py Skript auf dem Knoten, der den Angreifer simuliert, erreicht. Das Skript baut so viele Verbindungen wie möglich zu dem Satelliten auf und versendet nur Teilanfragen. Dies hat den Effekt, das auf dem Satelliten die Anfragen

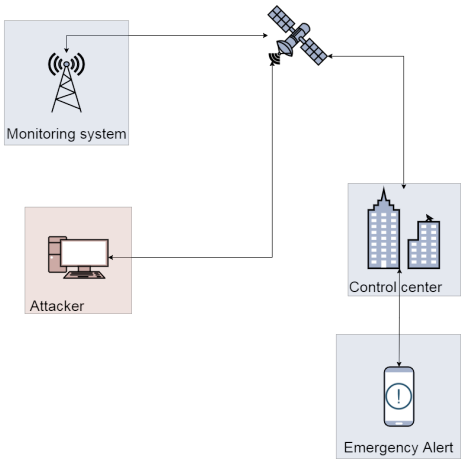

Abbildung 3: Netzwerktopologie des Slowloris Szenarios

nie vollständig abgeschlossen werden, und die Verbindungen zum Angreifer bestehen bleiben. Werden auf diese Weise genügend Verbindungen gleichzeitig blockiert, so kann der Satellit von der Messstation eingehende Verbindungsanfragen nicht annehmen und die Verbindung zwischen Messstation und Kontrollzentrum ist unterbrochen. Die Verbindungsunterbrechung tritt bereits den Bruchteil einer Sekunde nachdem der Angriff gestartet wurde auf und ist somit deutlich schneller als bei dem Bundle Flooding Angriff.

Diese Art Denial of Service Attacke zeichnet sich besonders dadurch aus, dass sie vollkommen vom ION-Netzwerk unabhängig ist, also kein bereits kompromittiertes System innerhalb des Netzwerkes benötigt wird, um erfolgreich die Verbindung zu stören. Ein Angreifer mit ausreichend leistungsstarker Hardware könnte damit sehr einfach sämtlichen Traffic in kritischen Systemen zum Erliegen bringen, sofern diese TCP als Tansportprotokoll nutzen.

Die Visualisierung der Auswirkungen des Angriffes in unserer Netzwerksimulation erfolgt wieder durch ein Shell-Skript, welches das Ergebnis des Pings von Messstation zum Kontrollzentrum in einer Konsole ausgibt.

6 Fazit und Ausblick

- kurze Zusammenfassung
- welche Anforderungen wurden erfüllt?
- was könnte verbessert werden?
- warum konnten verschiedene Anforderungen nicht erfüllt werden (hoffentlich nicht benötigt)

Literatur

[1] Gokberk Yaltirakli. „Slowloris“. In: *github.com* (2015). URL: `https://github.com/gkbrk/slowloris`.