INF473X-2020
Participants
Badges
Competencies
Grades
General
Welcome to MODAL-HACK
Misc Ressources
Tutorial 1: Programming in C (5 points)
Tutorial 2: Network Programming in C (5 points)
Tutorial 3: Concurrent Programming in C (5 points)
Tutorial 4: Sniffing and IP Spoofing (5 points)
μ-challenge: Vulnerable Applications (10 points)
ν-challenge: Vulnerable Applications - Advanced (10 points)
Σ-challenge: TCP SYN Flooding (10 points)
δ-challenge: DNS Hi-Jacking (10 points)
τ-challenge: TCP Hi-Jacking (10 points)
Ψ-Challenge
Dashboard
Site home
Calendar
My courses
AMPHI0-2019
SDD371-2019
STG1A-2019
PSC411-2020
INF473X-2020
INF412-2020
LAN413ALL-2020
MAA304T-2020
MIE431-2020
PHY433-2020

INF473X - Modal d'informatique - Cybersecurity - The Hacking Xperience (2020-2021)

τ-challenge: TCP Hi-Jacking

τ-challenge: TCP Hi-Jacking

This challenge has two different parts:

- Part 1 consists of "half-way killing" an ongoing TCP connection
- Part 2 consists of replacing one of the endpoints of a TCP connection, without the other side detecting that this is happening.

The basic idea is quite simple, and consists of:

1. The **attacker** being on the same link as the **victim**, and is able to overhear the traffic between the the **victim** and the **remote endpoint** of the connection;
2. The **attacker** detects that connection establishment is happening (SYN's are being exchanged), and records the chosen initial sequence numbers, from either side
3. The **attacker**, then, sends a TCP segment with the RST flag set, and spoofing the IP address of the **remote endpoint** of the connection, to the **victim**

At this point, the **victim** thinks that the **remote endpoint** has chosen to "reset" the connection - this is, among other things, what the infamous "Great Firewall of China" does, to kill connections to destinations, of which they do not approve.

The **remote endpoint**, however, still thinks that the connection is alive and well - and, consequently, will continue to communicate with the **victim**. Traffic which the **victim** will ignore (since it relates to what it believes to be a by-now-terminated connection) - but which the **attacker** will be able to overhear.

At this point, the **attacker** can switch to spoofing the IP address of the **victim** (as was insisted on in the lectures since the beginning: since the **victim** and the **attacker** are on the same link, they appear as *"topologically connected to the same place in the Internet"*), and effectively has taken over the TCP connection.

The two attacks possible for this attack are, obviously:

- **To prevent someone on the same link as you**, from ever initiating any TCP connection to anywhere. This is a form of "Denial of Service" attack (as you are denying service to someone), albeit not one that is based in ressource exhaustion (as many DoS-attacks generally are assumed to be).
- **To hi-jack and exploit an existing connection**, so as to inject malicious data - or, receive data not intended for you. This requires understanding a little bit about higher-level protocols.

For the latter, the sake of simplicity, let us consider remotely logging in to a computer using telnet (although this easily works with other clear-text protocols such as FTP, also, remote login via telnet probably is the easiest demonstrator).

Remote login via telnet works as follows:

1. A TCP connection is created by the client, and to the remote system
2. The remote system prompts for a login
3. The client provides a login username
4. The remote system prompts for a password
5. The client provides a password
6. The remote system considers the TCP connection to the client authenticated, and will therefore consider everything received over this TCP connection to be "authenticated". In the specific situation of remote login, this essentially means that everything received is accepted and executed as a shell command - such as, for example, 'rm -Rf *' ...

Obviously, an attacker would want to wait until after step 5, before hi-jacking the connection (sending an RST).

While partial credit will be given for just "Part 1" (To prevent someone on the same link as you, from ever initiating any TCP connection), full credit requires actually demonstrating hi-jacking and exploiting an active connection (of your choosing, which might include setting up a Telnet server for the simplest possible attack vector).

For completeness, in part due to this attack, Telnet - and similar "plain text, non-encrypted" connections - generally are not used for remote login. Rather, they are replaced by SSH (where an exchange of public keys take place, required for item 6 and forward in the remote login procedure).

Still, a non-trivial amount of "stuff" (for example, network infrastructure gear) allows remote login over telnet for access to configuration CLIs etc, rendering this a viable attack even to this day.

End