Hüseyin Bozkurt

Die Dokumentation des 1.Sprints:

20.1.2015-20.2.2015

Sprint #1

Erik Brändli, Hüseyin Bozkurt

Contents

[1) Changelog 1](#_Toc422322930)

[2) Sprintdaten 2](#_Toc422322931)

[4) Kommentare 3](#_Toc422322932)

[3](#_Toc422322933)

[Globale Anwendung 9](#_Toc422322934)

[Entscheidungsgrundlagen 9](#_Toc422322935)

# 1) Changelog

|  |  |  |  |
| --- | --- | --- | --- |
|  | *Datum* | *Name* | *Unterschrift* |
| Erstellt | 16.06.15 | Hüseyin Bozkurt |  |
| Geprüft | 17.06.15 | Erik Brändli |  |
| Freigegeben |  |  |  |

# 2) Sprintdaten

Zu finden auf taiga.io: <https://tree.taiga.io/project/selfmade-secX/taskboard/main-sprint-1-1>

Im Zeitraum dieses Sprints wurden folgende Userstories bearbeitet: 1,3,5,6,17.

Dauer: 4 Wochen.

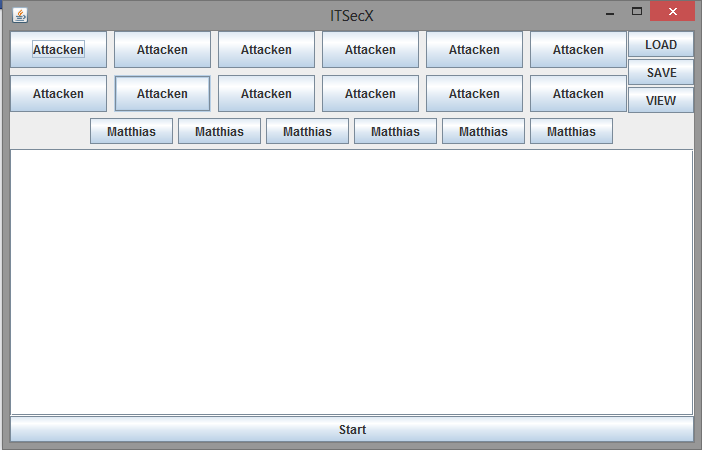
# 4) Kommentare

**US#1 Fertigstellung:**

# C:\Users\Hüseyin\Desktop\k-lernen9-tcp-client-server.gif

Wie im Evaluationssprint vorgestellt, die Server-Client Verbindung mittels Sockets.

**#3 GUI-Realisierung:**



Dies war die erste implementierte GUI. Buttons in verscheidenen Panels, geschmückt mit einem JtextField als LiveLog. Die höchste Priorität der Graphischen Oberfläche wurde in der Evaluationsphase des Projektes festgelegt. Dies besagte, dass die Oberfläche möglichst einfach zu bedienen und selbsterklärend gestaltet wären soll. Die Buttons wurden dann später durch eine Drop-Down Liste vereinfacht und die Buttons mit der Inschrift "Matthias" sollen die Module und Konfigurationsdateien sein, die man durch den Load Button eingefügt hat. Ebenfalls sollten selbstentwickelte Attacken in das Programm einbezogen werden können. Durch den Button "Save" sollte der Inhalt des Livelogs exportiert werden und in einer .txt Datei gespeichert werden. Der Button "View" war für das erneute ansehen und weiterführen einer Attacke gedacht, nach einem Stromausfall oder ähnlichem. Die Funktion dahinter sollte den Ordner für alle gespeicherten Konfigurationsdateien öffnen.

**Code:**

package Bozkurt;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import javax.swing.\*;

public class GUI extends JFrame {

//-----Attribute-----

JButton[][] jb = new JButton[2][6];

JButton[] unten = new JButton[6];

JButton[] conf = new JButton[3];

JTextField log = new JTextField();

JButton start = new JButton("Start");

private al al1;

//-----Methoden------

public GUI() {

JPanel nord = new JPanel(new BorderLayout());

//JPanel flow1 = new JPanel(new FlowLayout());

//JPanel flow2 = new JPanel(new FlowLayout());

JPanel flow3 = new JPanel(new FlowLayout());

JPanel grid = new JPanel(new GridLayout(3,1,2,2));

JPanel gridn = new JPanel(new GridLayout(2,6,7,7));

for(int i = 0; i < jb.length; i++) {

for(int a = 0 ; a < jb[i].length;a++){

jb[i][a]= new JButton("Attacken");

jb[i][a].setBounds(6,6,6,6);

jb[i][a].addActionListener(this.al1);

unten[a] = new JButton("Matthias");

unten[a].addActionListener(this.al1);

gridn.add(jb[i][a]);

// if(i == 1){

// flow2.add(jb[i][a]);

// }else{

// flow1.add(jb[0][a]);

// }

}

}

for(int i = 0; i < unten.length; i++) {

flow3.add(unten[i]);

}

conf[0] = new JButton("LOAD");

conf[1] = new JButton("SAVE");

conf[2] = new JButton("VIEW");

for(int i = 0; i < conf.length; i++) {

grid.add(conf[i]);

}

//nord.add(flow1,BorderLayout.NORTH);

nord.add(gridn,BorderLayout.CENTER);

nord.add(flow3,BorderLayout.SOUTH);

nord.add(grid, BorderLayout.EAST);

this.setLayout(new BorderLayout());

this.add(nord, BorderLayout.NORTH);

this.add(log);

this.add(start, BorderLayout.SOUTH);

this.setTitle("ITSecX");

this.setSize(700,450);

this.setVisible(true);

}

public static void main(String[] args) {

new GUI();

}

public class al implements ActionListener{

@Override

public void actionPerformed(ActionEvent arg0) {

// TODO Auto-generated method stub

}

}

}

**#6: Entwicklung einer Attacke (metasploit)**

Das Metasploit-Projekt ist ein freies Open-Source-Projekt zur Computersicherheit, das Informationen über Sicherheitslücken bietet und bei Penetrationstests sowie der Entwicklung von IDS-Signaturen eingesetzt werden kann. Das bekannteste Teilprojekt ist das Metasploit Framework, ein Werkzeug zur Entwicklung und Ausführung von Exploits gegen verteilte Zielrechner. Andere wichtige Teilprojekte sind das Shellcode-Archiv und Forschung im Bereich der IT-Sicherheit.

Wie vergleichbare kommerzielle Lösungen, so z. B. Canvas (von Immunity) oder Core Impact (von Core Security Technology), kann Metasploit von Administratoren eingesetzt werden, um die Schwachstellen von Computersystemen zu prüfen und diese bei Bedarf zu schließen. Andererseits kann es auch missbraucht werden, um in andere Systeme einzubrechen. Während der beschriebene Einsatz durch einen Administrator in seinem eigenen Netzwerk nicht nur legitim, sondern auch legal ist, erfüllt ein Einsatz ohne ausdrückliche Erlaubnis bei Fremdsystemen verschiedene Tatbestände der Computerkriminalität.

Da es nicht möglich ist, dass die interaktive Shell nur mit einem Befehl arbeitet (metasploit verlangt eine aktive Sitzung), konnten wir dieses Kapitel nicht in unser Projekt einbeziehen.

**#17: Verschlüsselung der Übertragung**

Secure Shell oder SSH bezeichnet sowohl ein Netzwerkprotokoll als auch entsprechende Programme, mit deren Hilfe man auf eine sichere Art und Weise eine verschlüsselte Netzwerkverbindung mit einem entfernten Gerät herstellen kann. Häufig wird diese Methode verwendet, um lokal eine entfernte Kommandozeile verfügbar zu machen, das heißt, auf einer lokalen Konsole werden die Ausgaben der entfernten Konsole ausgegeben und die lokalen Tastatureingaben werden an den entfernten Rechner gesendet. Genutzt werden kann dies beispielsweise zur Fernwartung eines in einem entfernten Rechenzentrum stehenden Servers. Die neuere Protokoll-Version SSH-2 bietet weitere Funktionen wie Datenübertragung per SFTP.

Nachdem einige Schwachstellen in der Integritätsprüfung von SSH-1 bekannt geworden waren, wurde 1996 mit SSH-2 eine überarbeitete Version des Protokolls entwickelt. Sie ist inkompatibel zu SSH-1. Dabei wurde unter anderem das Protokoll in verschiedene Einzelteile aufgegliedert und somit die Verwendung sicherer Verschlüsselungs- und Authentifikations-Algorithmen ermöglicht. Damit wurde die Schwachstelle beseitigt. Derzeit gilt das Protokoll als sicher.

## Globale Anwendung

SSH ermöglicht eine sichere, authentifizierte und verschlüsselte Verbindung zwischen zwei Rechnern über ein unsicheres Netzwerk. Dadurch dient es unter anderem als Ersatz für die Vorgänger rlogin, telnet und rsh; diese übertragen jeglichen Netzverkehr, darunter auch die Passwörter, unverschlüsselt.  
SSH ermöglicht eine sichere, authentifizierte und verschlüsselte Verbindung zwischen zwei Rechnern über ein unsicheres Netzwerk. Dadurch dient es unter anderem als Ersatz für die Vorgänger rlogin, telnet und rsh; diese übertragen jeglichen Netzverkehr, darunter auch die Passwörter, unverschlüsselt.

Das ursprüngliche Anwendungsgebiet ist das Anmelden an entfernten Rechnern über ein Netzwerk (meistens das Internet), doch insbesondere SSH-2 ist nicht nur auf Terminalfunktionen beschränkt.

* SFTP und SCP bieten kryptographisch sicherere Alternativen zu FTP und RCP.
* X11 kann über SSH transportiert und somit gesichert werden.
* Über SSH können beliebige TCP/IP-Verbindungen getunnelt werden (Portweiterleitung); dabei wird jeweils ein einzelner Port von einem entfernten Server auf den Client weitergeleitet oder umgekehrt. So kann etwa eine ansonsten unverschlüsselte VNC-Verbindung abgesichert werden.
* Ein SSH-Client kann sich wie ein SOCKS-Server verhalten und ermöglicht somit einen automatisierten Zugriff auf entfernte Rechner durch den SSH-Tunnel, etwa zum Umgehen einer Firewall.
* Über SSHFS kann ein entferntes Dateisystem auf dem lokalen Rechner gemountet werden.
* Mit „ssh-keyscan“ kann der öffentliche Schlüssel eines entfernten Rechners ausgelesen werden. Damit kann man unter Zuhilfenahme des zugehörigen öffentlichen Schlüssels zum Beispiel feststellen, ob die IP-Adresse und/oder der DNS-Eintrag eines SSH-Servers manipuliert worden ist

1. Secure System Administration (Sichere Systemverwaltung) zur Absicherung der Fernverwaltung von Servern. Ersetzt telnet, rlogin etc.
2. Secure Application Tunneling (Sicheres Tunneln) zum transparenten Schutz TCP/IP-basierender Anwendungen als „End-to-End-Security“.
3. Secure Remote Command Execution (Sichere Ausführung von Kommandos) zur Ausführung einzelner Kommandos auf einem anderen Rechner. Dabei werden stdin, stdout und stderr transparent weitergeleitet.

## Entscheidungsgrundlagen

SSH verschlüsselt den gesamten Datenverkehr, login als auch die übertragenen Daten

Es ist momentan als Stable und als sicher eingestuft. Außerdem bittet SSH die tunnel-technologie an was ermöglicht, das alle Daten durchs SSH-Protokoll geschleust werden.