

IT-Sicherheits-Praktikum

Angreifer-Gruppe

Cuong Vo Ta, Max Mischinger, Florian Reiner



Inhalt

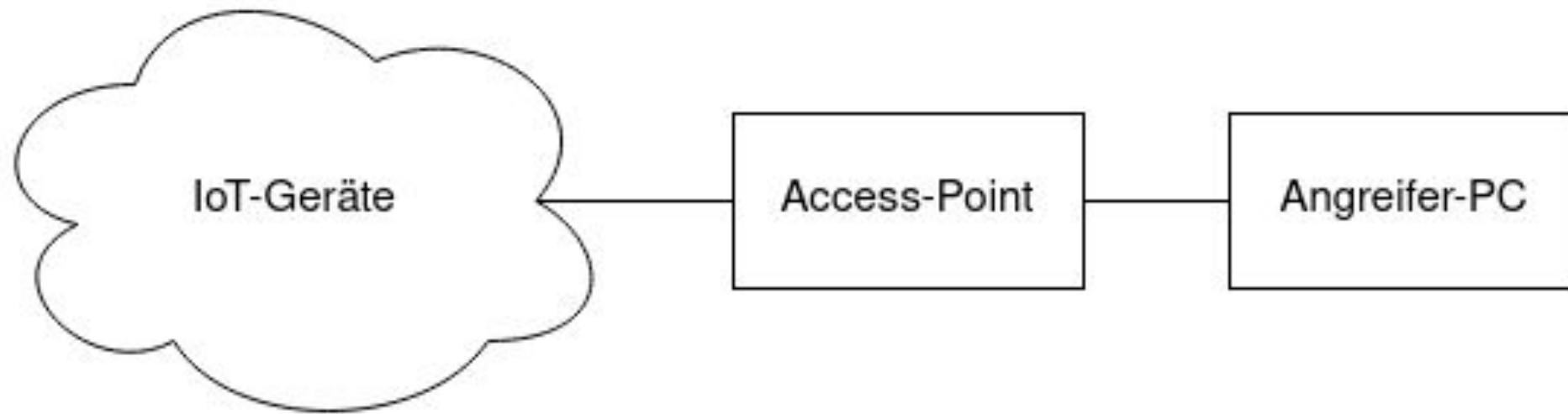
1. Projektziel
2. Aufbau
3. NMAP / Scanner
4. Metasploit
5. Angriffe
6. Fazit & Ausblick

Projektziel

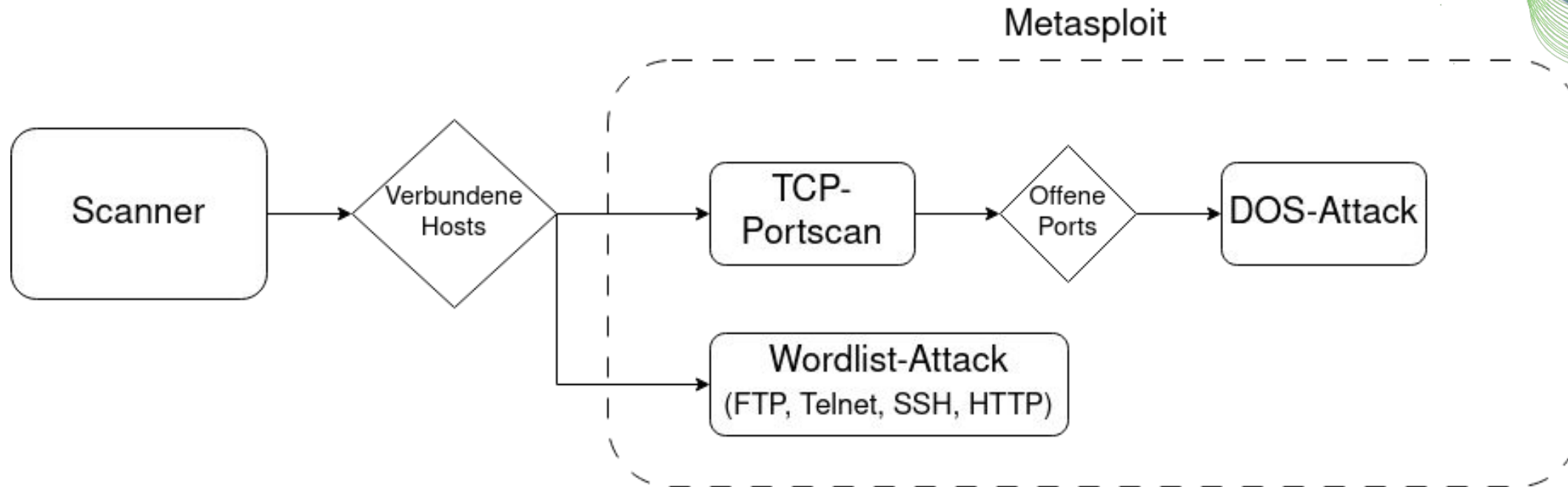
- Gemeinsames Hauptziel aller Praktikumsgruppen:
 - “Aufbau einer Toolchain, die den Netzwerkverkehr von Smart Home-Komponenten automatisch aufzeichnet, labelt und klassifiziert”
- **Unser Ziel als Angreifer-Gruppe:**
 - Automatisierte Angriffe
 - Versenden von Paketen mit bestimmten Payload, um Netzwerkpakete Angriffen zuordnen zu können
 - Integration in Infrastruktur



Aufbau

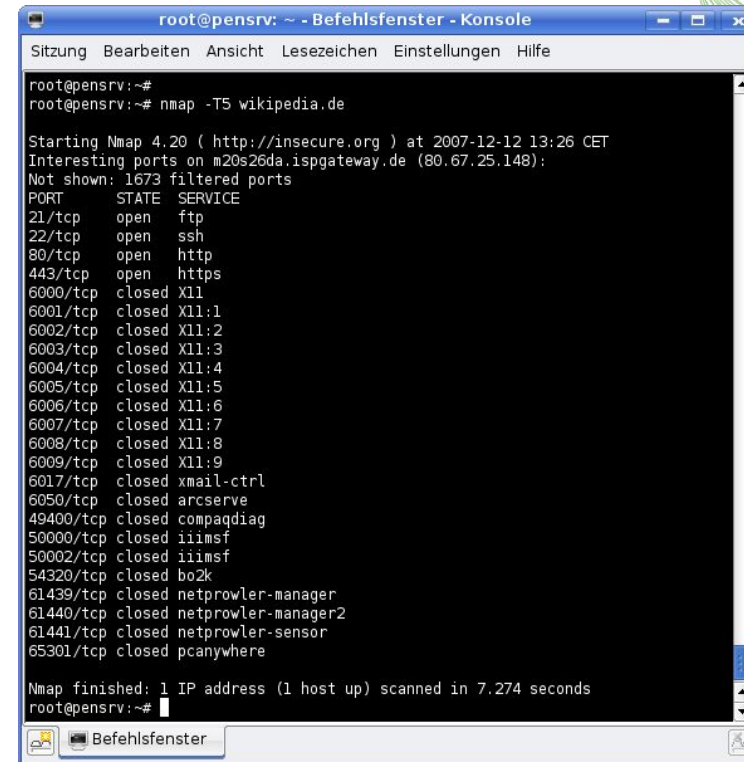


Aufbau



NMAP

- Netwerktool zum finden und analysieren von Hosts:
 - Beinhaltet verschiedene Funktionen zum scannen von Netzwerken
 - Hauptsächlich für Portscans benutzt
- NMAP in unserem Projekt:
 - Teil des Scanners
 - Erfasst alle Hosts im Testnetzwerk
 - Findet offene Ports für Angriffe



```
root@pensrv: ~ - Befehlsfenster - Konsole
Sitzung Bearbeiten Ansicht Lesezeichen Einstellungen Hilfe

root@pensrv:~#
root@pensrv:~# nmap -T5 wikipedia.de

Starting Nmap 4.20 ( http://insecure.org ) at 2007-12-12 13:26 CET
Interesting ports on m20s26da.ispgateway.de (80.67.25.148):
Not shown: 1673 filtered ports
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
80/tcp    open  http
443/tcp   open  https
6000/tcp  closed X11
6001/tcp  closed X11:1
6002/tcp  closed X11:2
6003/tcp  closed X11:3
6004/tcp  closed X11:4
6005/tcp  closed X11:5
6006/tcp  closed X11:6
6007/tcp  closed X11:7
6008/tcp  closed X11:8
6009/tcp  closed X11:9
6017/tcp  closed xmail-ctrl
6050/tcp  closed arcserve
49400/tcp closed compaqdiag
50000/tcp closed iiimsf
50002/tcp closed iiimsf
54320/tcp closed bo2k
61439/tcp closed netproowler-manager
61440/tcp closed netproowler-manager2
61441/tcp closed netproowler-sensor
65301/tcp closed pcanywhere

Nmap finished: 1 IP address (1 host up) scanned in 7.274 seconds
root@pensrv:~#
```

Scanner

- Erfasst und verwaltet Hosts
- Hosts haben bestimmte Eigenschaften, die durch Scanner populiert werden
- Informationen zu Hosts können in Angriffen verwendet werden
- Einheitliches Interface für Erweiterungen

```
class Host:
    def __init__(self, ip: str, mac: str = "", ports: Optional[list[int]] = None):
        self.ip: str = ip
        self.mac: str = mac
        self.ports = [] if ports is None else ports
        self.filtered_ports = {"all": self.ports}
        self.filter_needs_update = False
        self.new_ports = []
```

Scanner (Ablauf)

- Erkennt bei Initialisierung angeschlossene Netzwerke
- Führt Scans durch um Hosts zu ermitteln
- Sucht nach offenen Ports auf den Hosts
- Nach Initialisierung, vollst. Liste aller Hosts mit dazugehörigen Ports
- Offene Ports auf einem Host können per Filter einem Dienst zugeordnet werden

Scanner (Telnet-Beispiel: Filter + Bruteforce)

- Scanner kann um Filter erweitert werden
- Implementierung von Attacken und Filtern kann genauere Aufzeichnung ermöglichen

```
class TelnetFilter(Filter):
    name = "telnet"

    def __init__(self, ip: str, ports: list[int], threads=5):
        super(TelnetFilter, self).__init__()
        self.ip = ip
        self.ports = ports
        self.threads = threads
        self.executor: Optional[ThreadPoolExecutor] = None
        self.filtered_ports = []

    def filter_ports(self):
        self.executor = ThreadPoolExecutor(self.threads)
        futures = [self.executor.submit(connect_and_consume_login, self.ip, port) for port in self.ports]
        for future in concurrent.futures.as_completed(futures):
            if future.result() > 0:
                self.filtered_ports.append(future.result())
        self.executor.shutdown(wait=True, cancel_futures=False)
        return self.filtered_ports
```

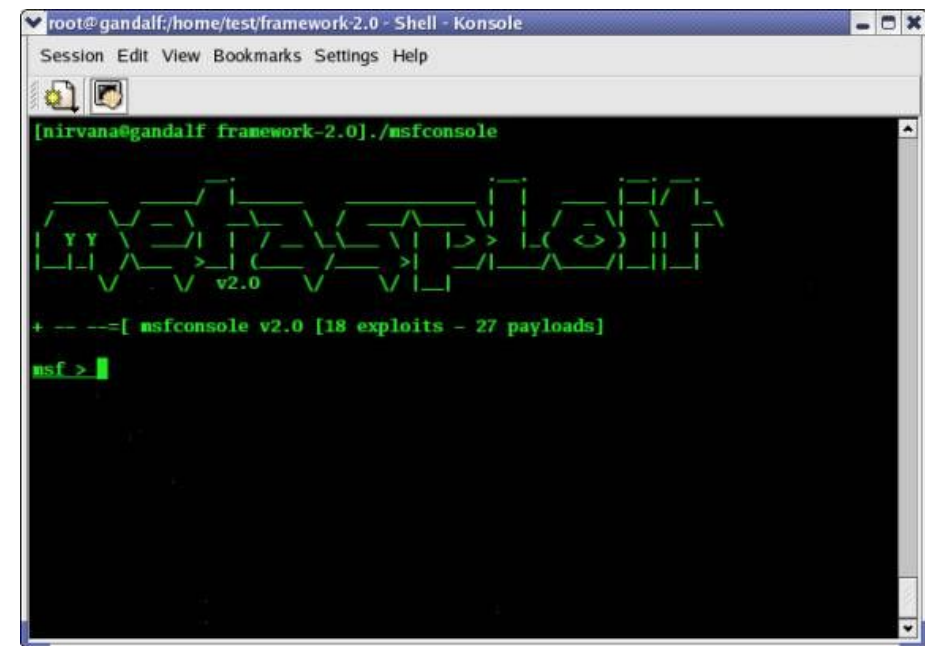
```
class Filter:
    name = "generic"

    def __init__(self):
        pass

    def filter_ports(self):
        pass
```

Metasploit

- IT-Sicherheits-Projekt entwickelt von Rapid7
- Bietet Informationen über Sicherheitslücken und Penetrationstests
- Teilprojekt **Metasploit-Framework**
 - Werkzeug zur Entwicklung und Ausführung von Exploits
 - Entwickelt in Ruby
 - Vorinstalliert auf Kali Linux
 - Version 6.2.37 umfasst 2278 Exploits

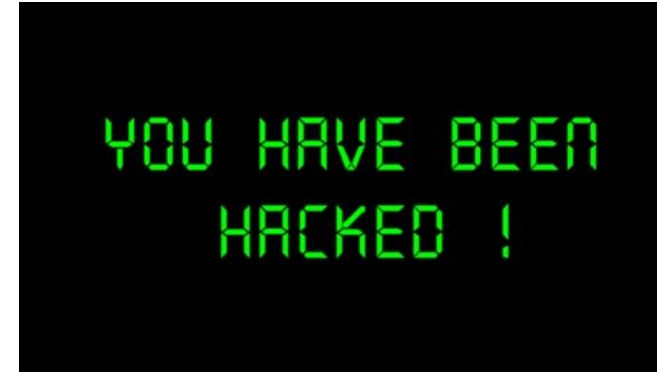


The screenshot shows a terminal window titled "root@gandalf:/home/test/framework-2.0 - Shell - Konsole". The prompt is "[nirvana@gandalf framework-2.0] ./msfconsole". The console displays a large ASCII art logo for the Metasploit Framework, version 2.0. Below the logo, it shows the command "+ --[msfconsole v2.0 [18 exploits - 27 payloads]" and the prompt "msf >".

Metasploit

Funktionsweise:

1. Exploit auswählen und konfigurieren
 - a. **Exploit** (Buffer Overflow, Code Injection, ...)
 - b. **Auxiliary** (Scanner, DoS-Attacks, ...)
 - c. **Post-Exploitation** (Hash Dumps, ...)
 - d. **Payload**
2. Payload wählen und konfigurieren
 - a. Payload ist Code, der auf dem Zielrechner bei erfolgreichem Einbruch ausgeführt
3. Ausführung des Exploits
4. Weiteres Vordringen auf dem Zielrechner mit weiteren Payloads



Metasploit - Demo

Angriffe - Demo

Fazit & Ausblick

- Modulares Run-Skript in Python
- Metasploit universelles Tool für viele verschiedene Angriffe
- Angriffe funktionieren und können von der AccessPoint-Gruppe gelabelt werden
- Neue Funktionalitäten können als Module hinzugefügt werden
 - Beispielsweise Angriffe, die spezifische Geräte angreifen
 - Portfilter für den Scanner (Interface bereits vorhanden)



Danke für die Aufmerksamkeit

Motivation und Projektziel

- Sicherheit von IoT: "Internet of Things" , Smart Home
 - Viele alltägliche Geräte sind mit dem Internet verbunden und können mit Handy gesteuert werden
 - Mögliche Gefahren?
 - Gerätezustand schlecht beobachtbar
 - Keine Benutzeroberfläche
 - Gerät steckt jahrelang am Netz
- Geräte sind ein leichtes Ziel für Angriffe
- Geräte könnten schon Bestandteil eines Botnetzes sein

