

Advanced Exploiting

tk, tk@trapkit.de

Stand der Dinge

- Schutzmechanismen werden ausgereifter (Netz/Host)
- Trend zu mehr Komplexität (→ CP AI/WI, Parser, ...)

Logische Konsequenzen für Angreifer:

- Notwendigkeit zur Verbesserung der Angriffstechniken (Bypass)
- Ausnutzung von Schwachstellen in Sicherheitsprodukten (Aufweichen der Config)

Überblick

- **Fokus:** Zusammenstellung einiger **Bypass-**Techniken und Beispiele
- Rückblick
- Heute
- Ausblick

Nr. 1 Schwachstellen

Was wird ausgenutzt?

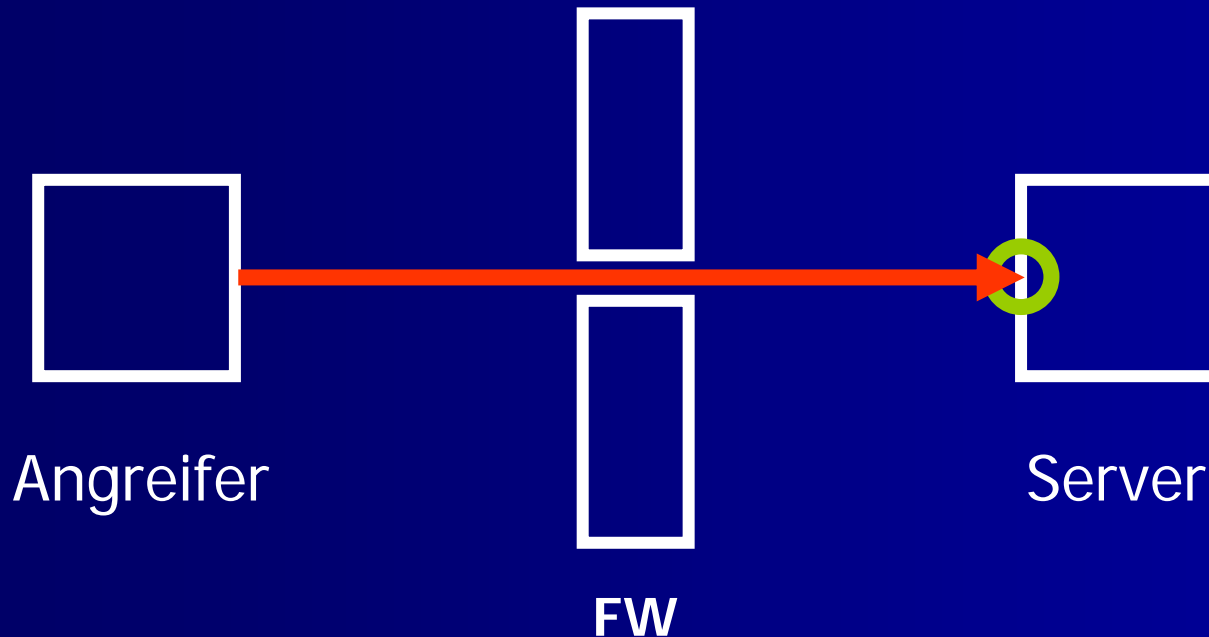
MCV

- **Nr. 1 Ursache** für erfolgreiche Systemeinträge: **Memory Corruption Vulnerabilities (MCV)**
- **Beispiele:** Buffer Overflows, Format-String-Schwachstellen, Integer Overflows, ...

Funktionsweise

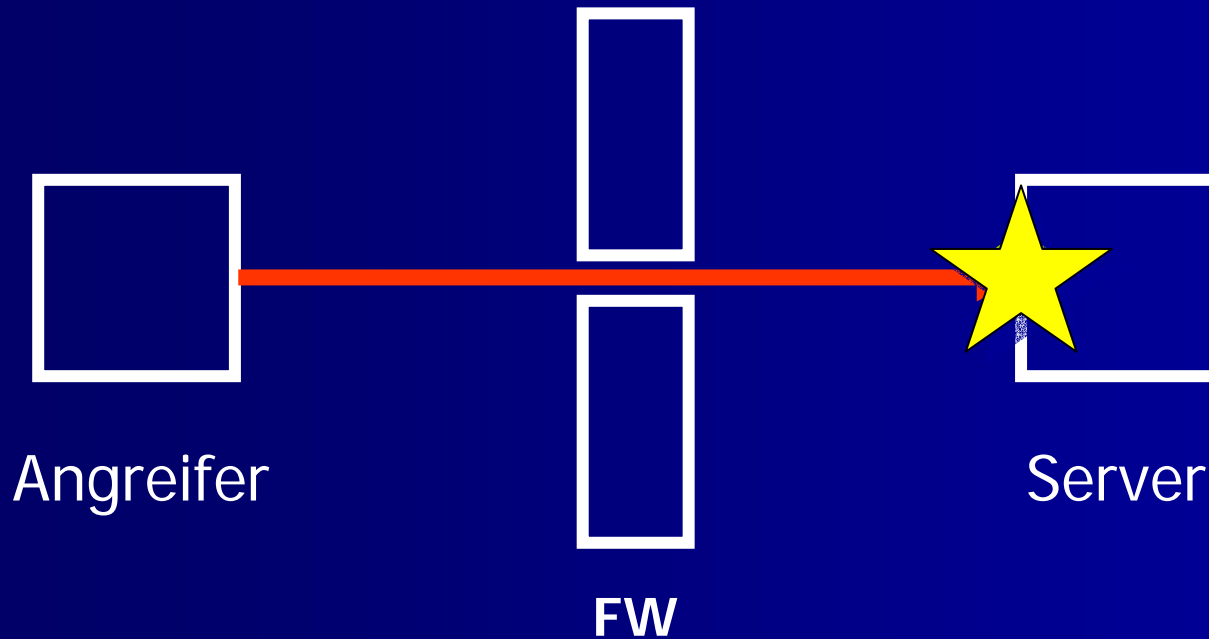
- Verständnis der „Funktionsweise“ der Schwachstellen wird vorausgesetzt ;)
- **Ergebnis:**
 - Kontrolle über den Programmfluss des fehlerhaften Prozesses
 - Ausführung von (Malicious) Code
- Entsprechende Exploits bestehen aus zwei Komponenten:
 - Injection Vector
 - Payload

Injection Vector



Injection Vector: Netzwerkkommunikation, Nachbildung des Applikationsprotokolls, Triggern/Ausnutzung des Fehlers, ...

Payload



Payload: Der auszuführende (Malicious) Code, wird im Kontext des fehlerhaften Prozesses ausgeführt.

MCV

- **(ein) Ergebnis:** Ausführung von (Malicious) Code

- Mehr Informationen?

→ „Buffer Overflows und Format-String-Schwachstellen“ (ISBN: 3-89864-192-9)

Advanced Exploiting

- Robuste Exploits
- Fehlerfreie, exakte Implementation des Injection Vectors (Protokollnachbildung etc.)
- Advanced Payloads
- ...

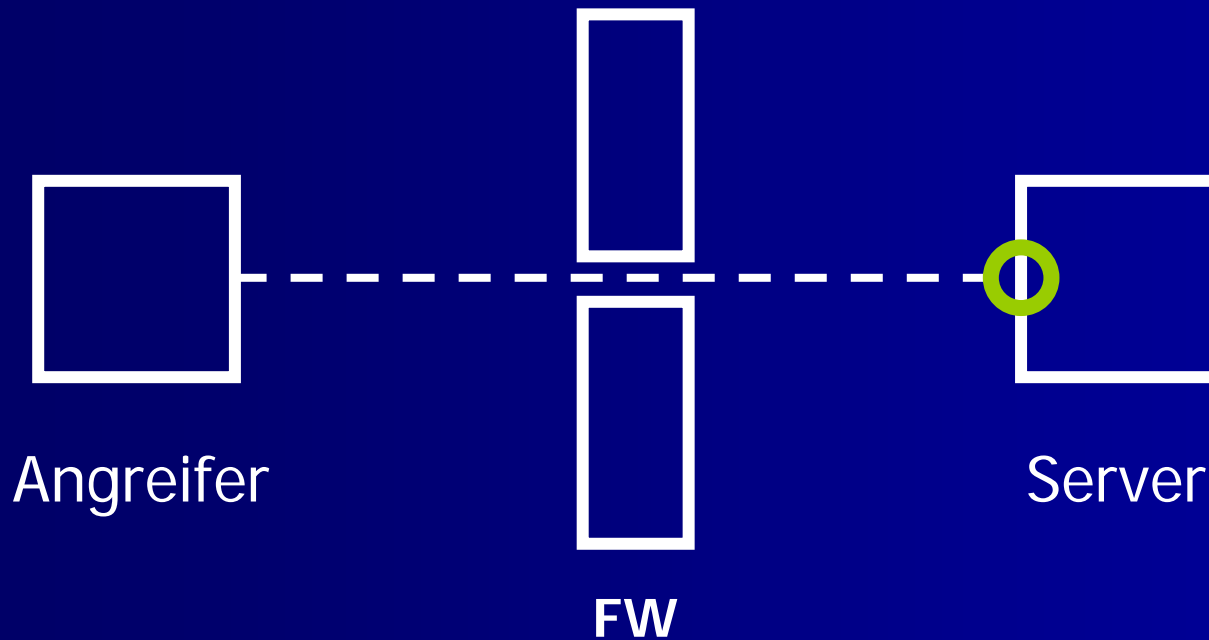
Payload

- Verschiedene denkbare Payloads
 - Ist es eine lokale oder eine remote Schwachstelle?
 - Was will der Angreifer erreichen?
 - ...
- Thema dieses Vortrags: Payloads für remote Exploits

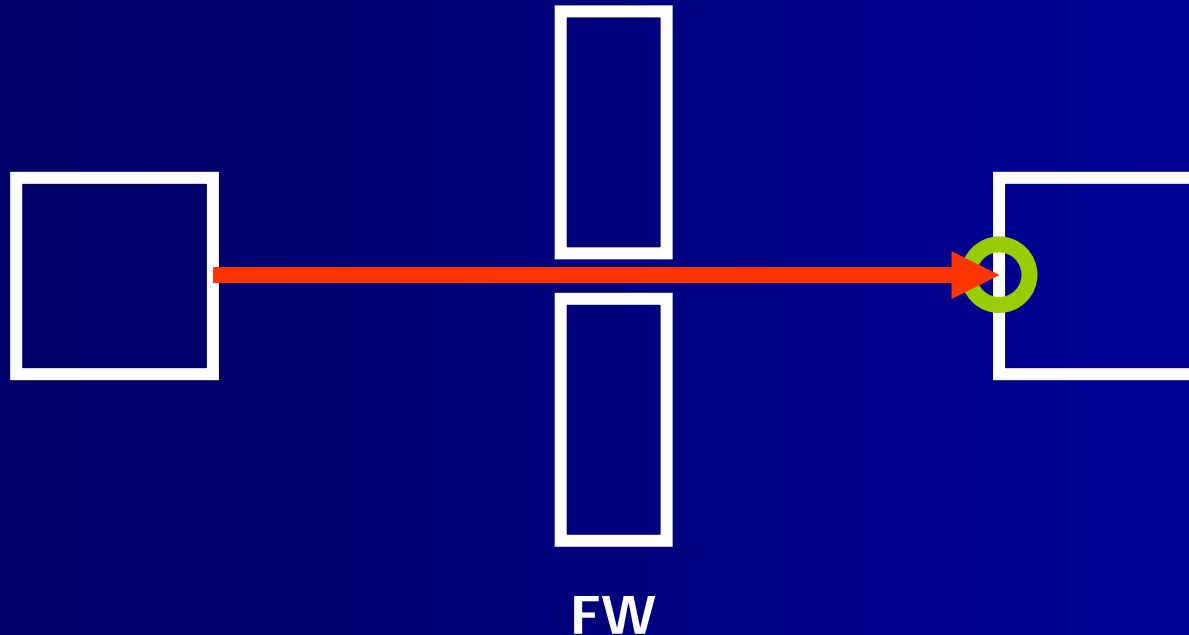
Payload – Remote Exploits

- Old School bzw. PoC Payloads:
 - Bind (a new) Port, Backconnect/Reverse Connect
- Klassiker
- **Probleme** mit stateful perimeter Firewalls á la Check Point, Netscreen, Cisco PIX, SEF, ...

Payload – Bind (a new) Port

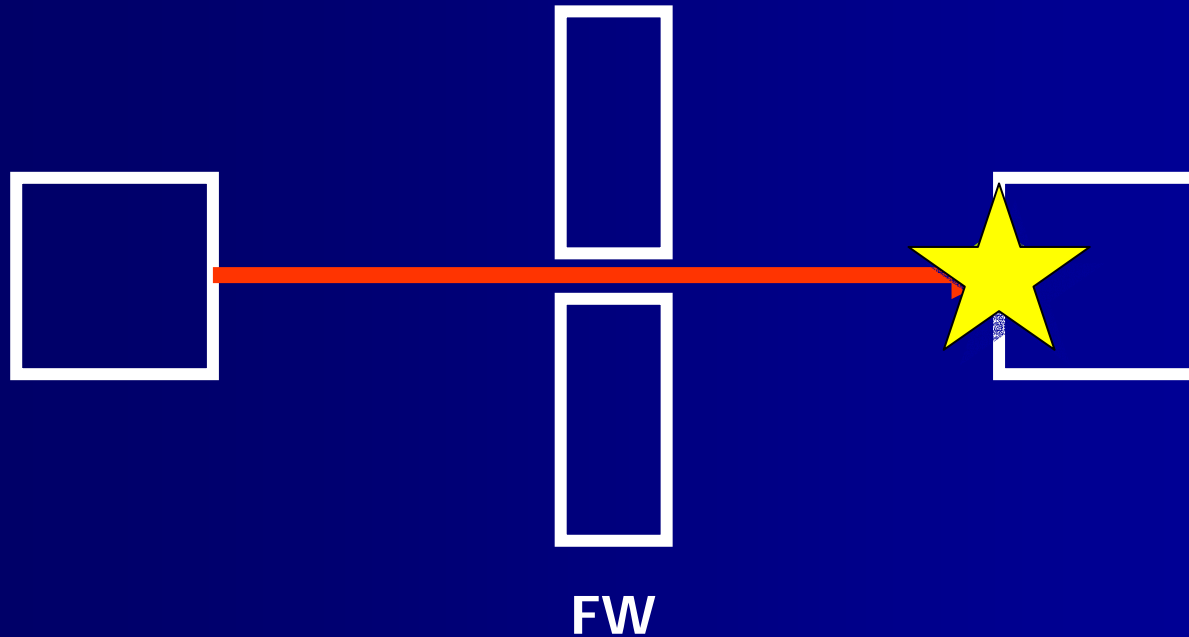


Payload – Bind (a new) Port



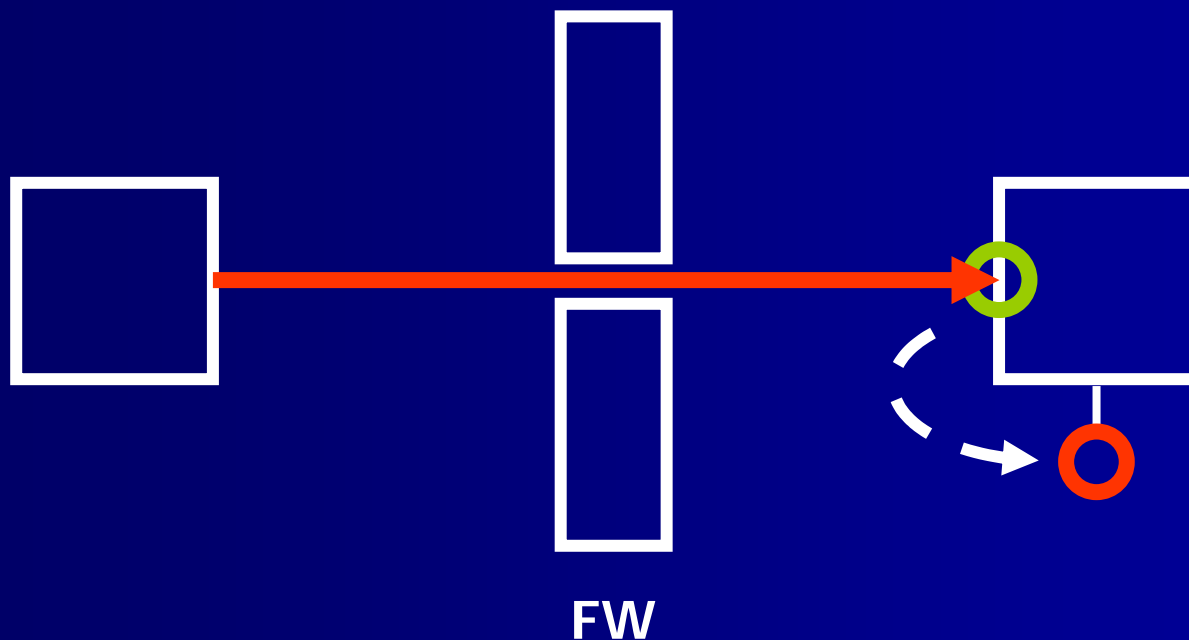
Verbindung zu dem Server Dienst und gezielte Ausnutzung der Schwachstelle (IV) ...

Payload – Bind (a new) Port



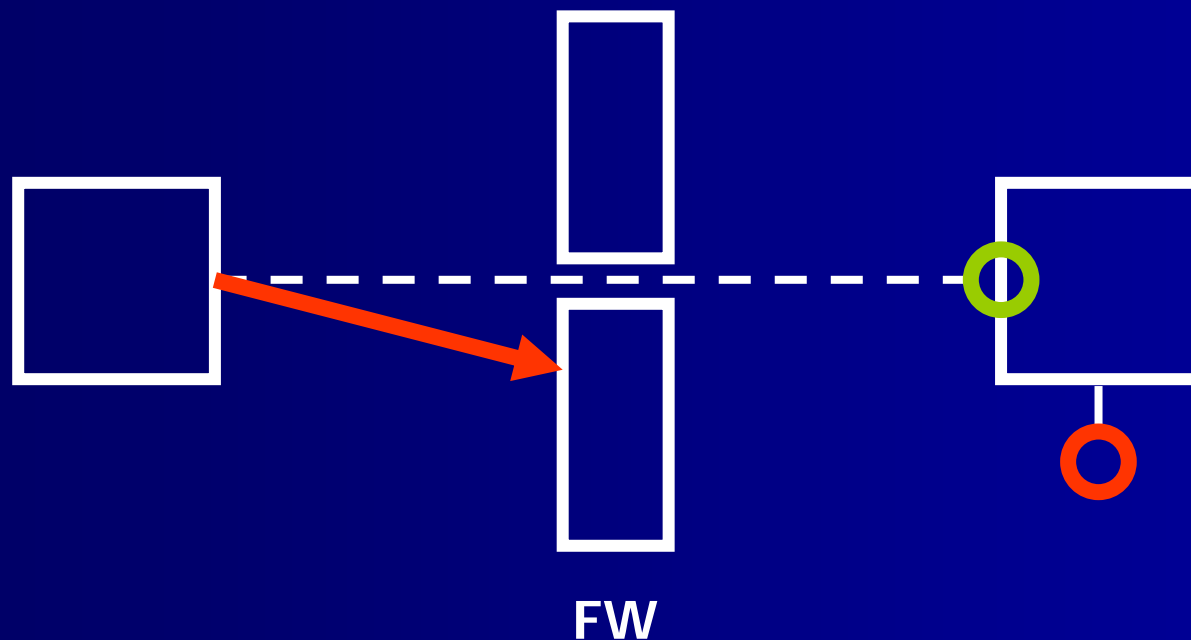
Möglichkeit zur Ausführung beliebigen Codes im Kontext des ausgenutzten Prozesses ...

Payload – Bind (a new) Port



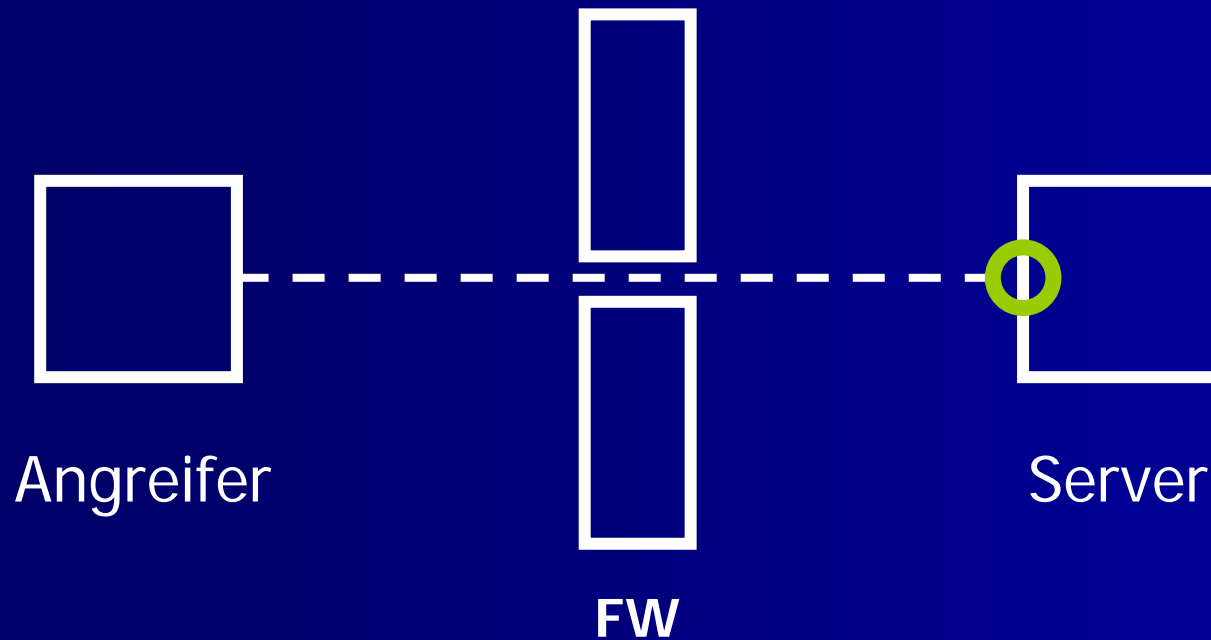
Öffnet einen neuen Netzwerksocket z.B. Port 6666
und bindet daran die Shell/cmd.exe

Payload – Bind (a new) Port

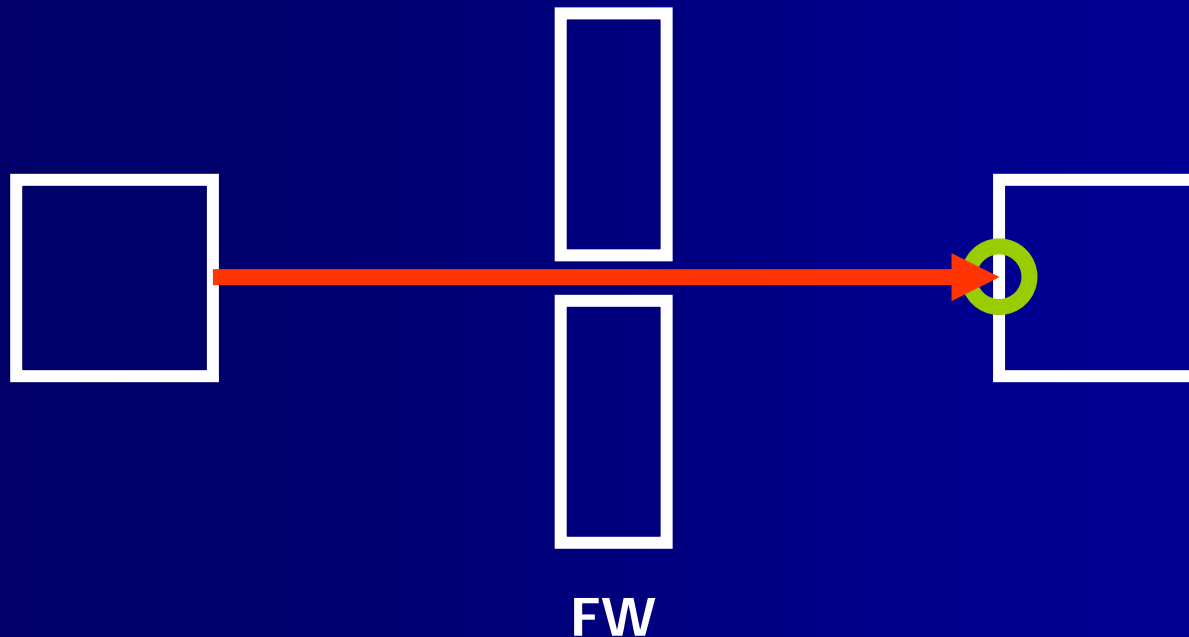


Verbindung zur Backdoor (hier Port 6666) wird von der Firewall (hoffentlich ;) unterbunden!

Payload – Backconnect

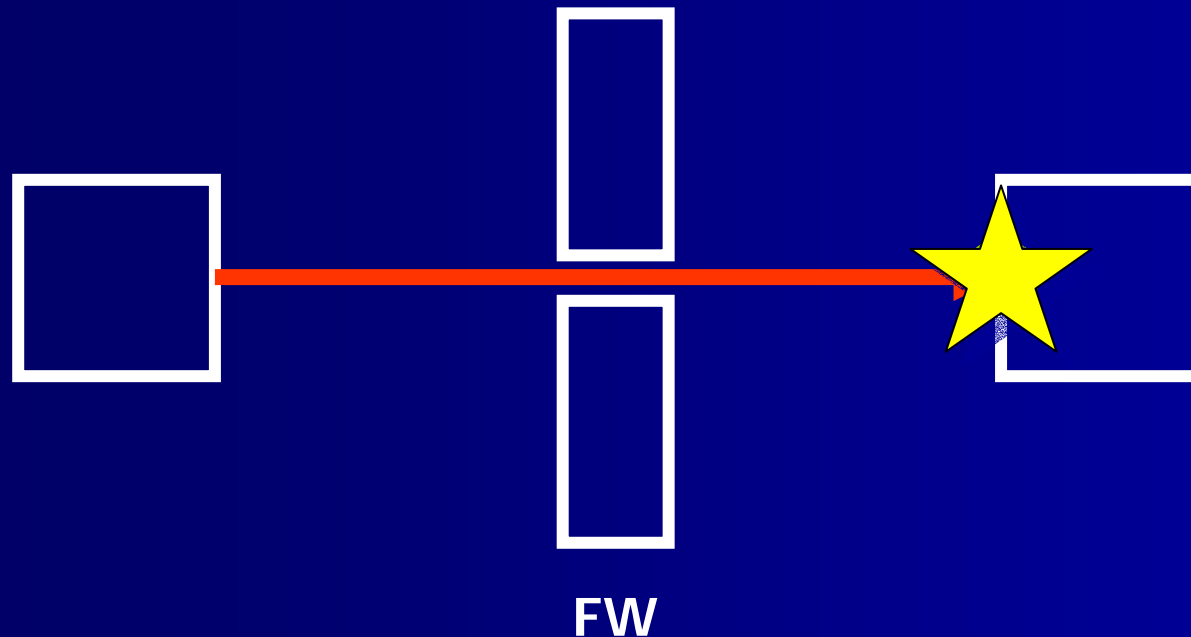


Payload – Backconnect



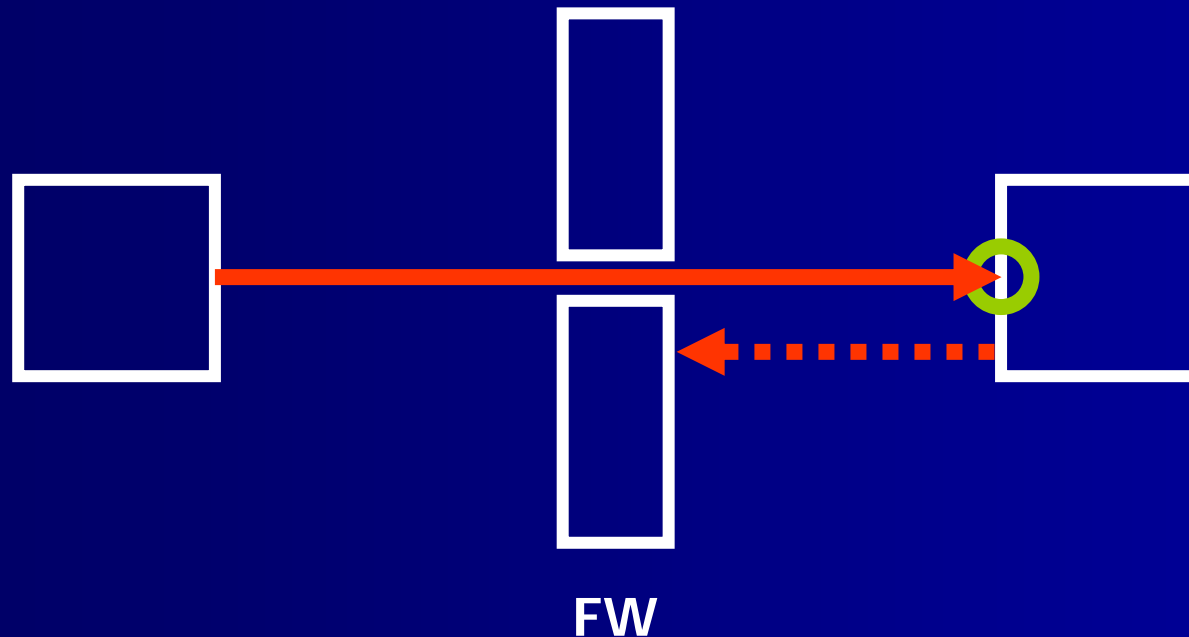
Verbindung zu dem Server Dienst und gezielte Ausnutzung der Schwachstelle (IV) ...

Payload – Backconnect



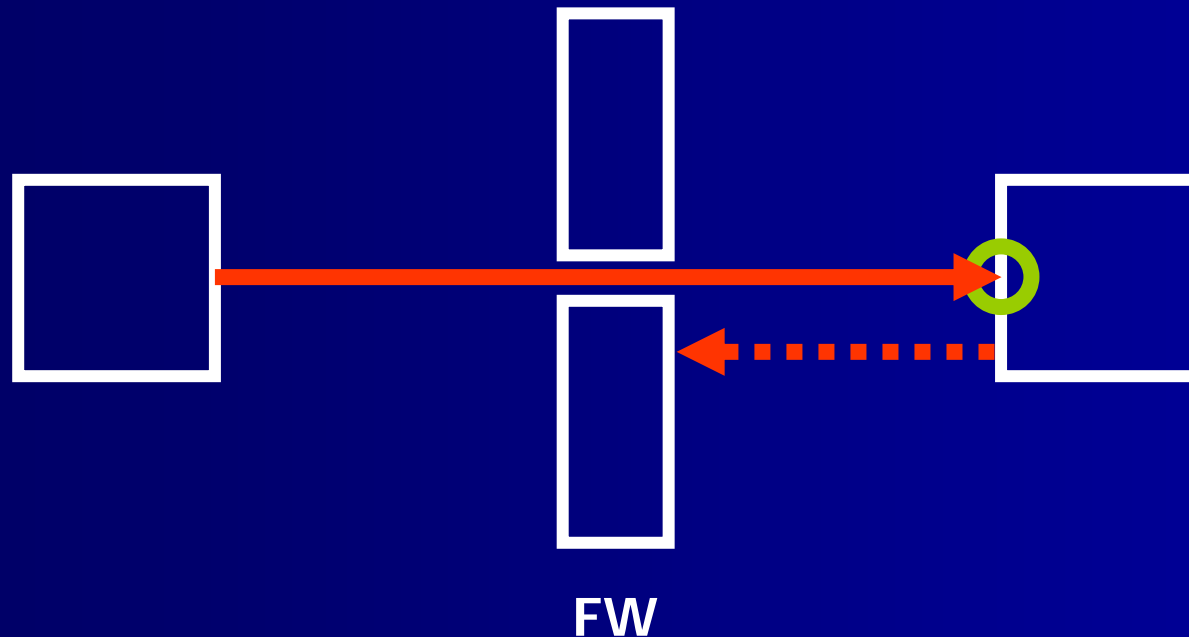
Möglichkeit zur Ausführung beliebigen Codes im Kontext des ausgenutzten Prozesses ...

Payload – Backconnect



Öffne einen neuen Netzwerksocket und mache eine Rückverbindung. Falls diese klappt, binde die Shell/cmd.exe an diesen Socket.

Payload – Backconnect



Diese Rückverbindung wird (hoffentlich) wiederum durch die Firewall unterbunden!

Payload – PoC

- Bind (a new) Port, Backconnect/Reverse Connect
- Proof of Concept Payloads
- Sollten in **keiner** Enterprise-Umgebung erfolgreich funktionieren!
- **Problem:** Stateful Firewalling

Advanced Payloads

Bypass

Payload – Anti stateful perimeter FW

- **Problem:** stateful perimeter Firewalls

- **Mögliche Lösungen:**

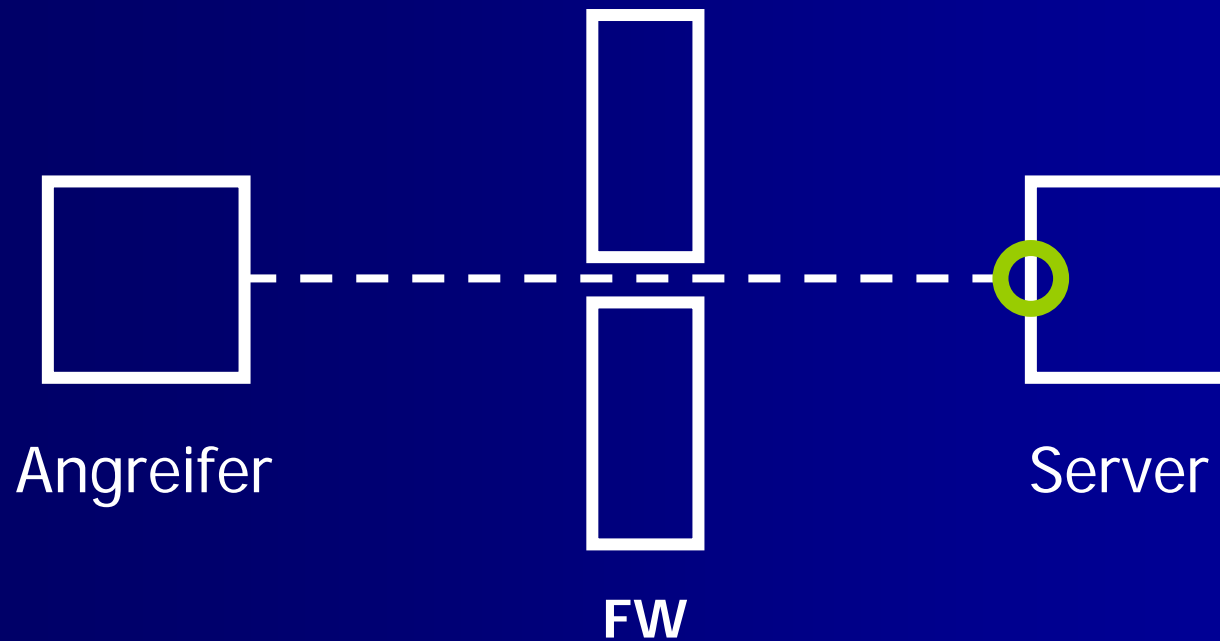
- Port Reuse Techniken (Win32)

- Port Reuse, Port Rebind

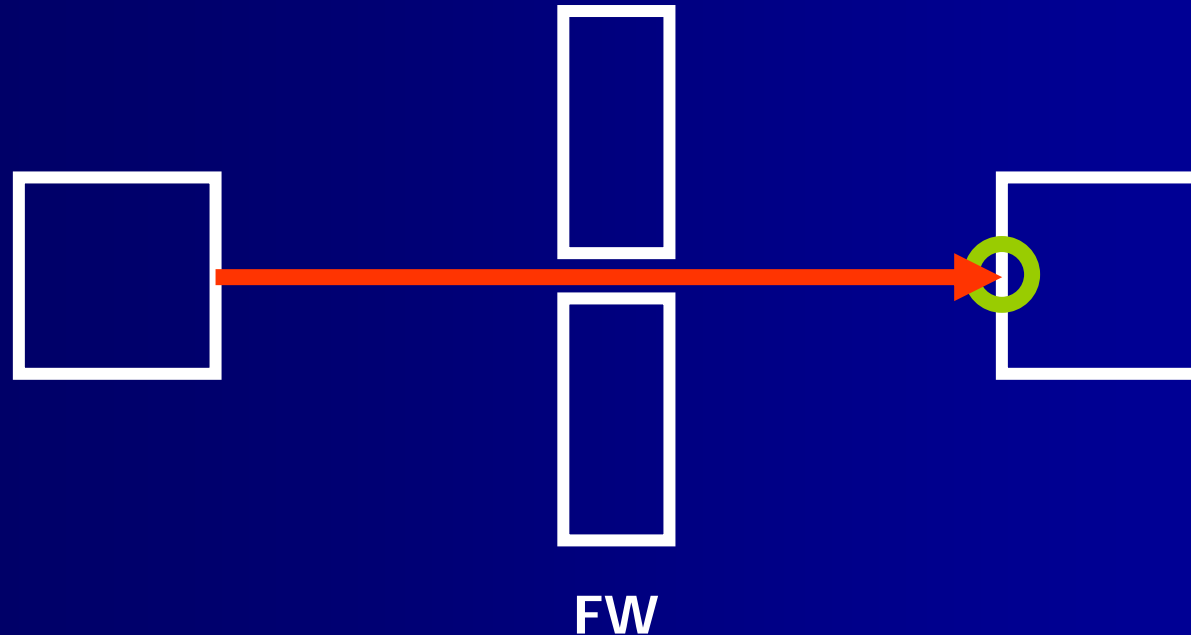
- Socket Reuse Techniken (uni)

- getpeername, ALIAScode, GOcode, ...

Payload – getpeername

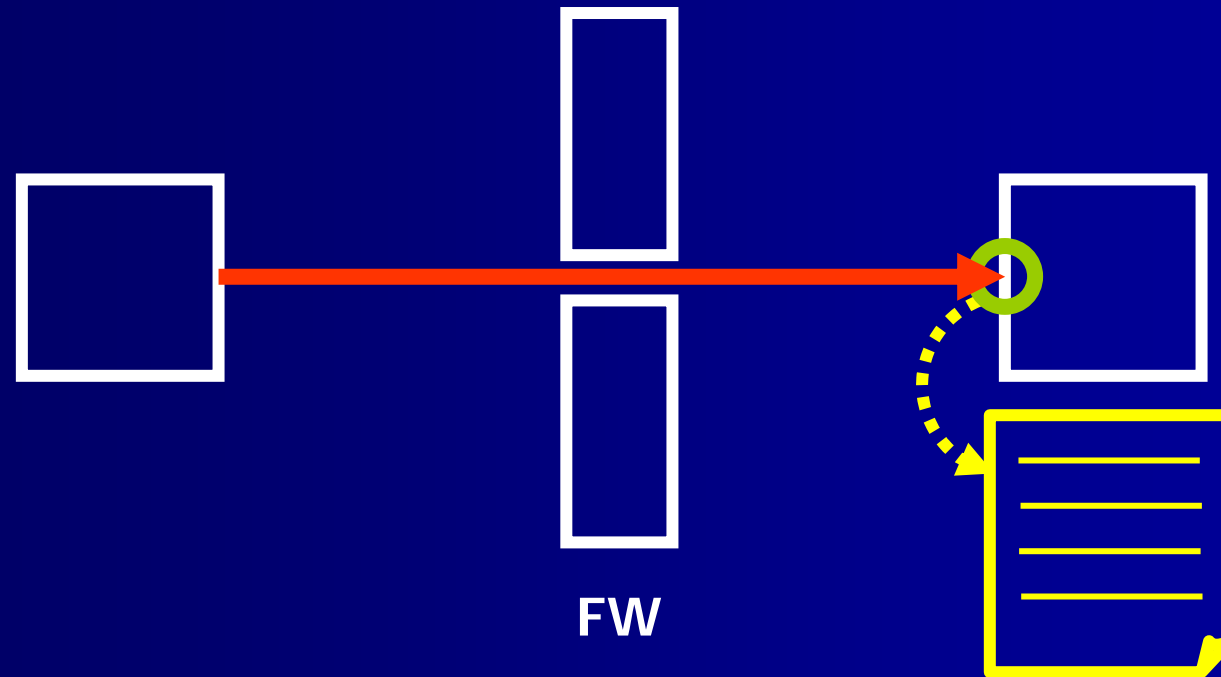


Payload – getpeername



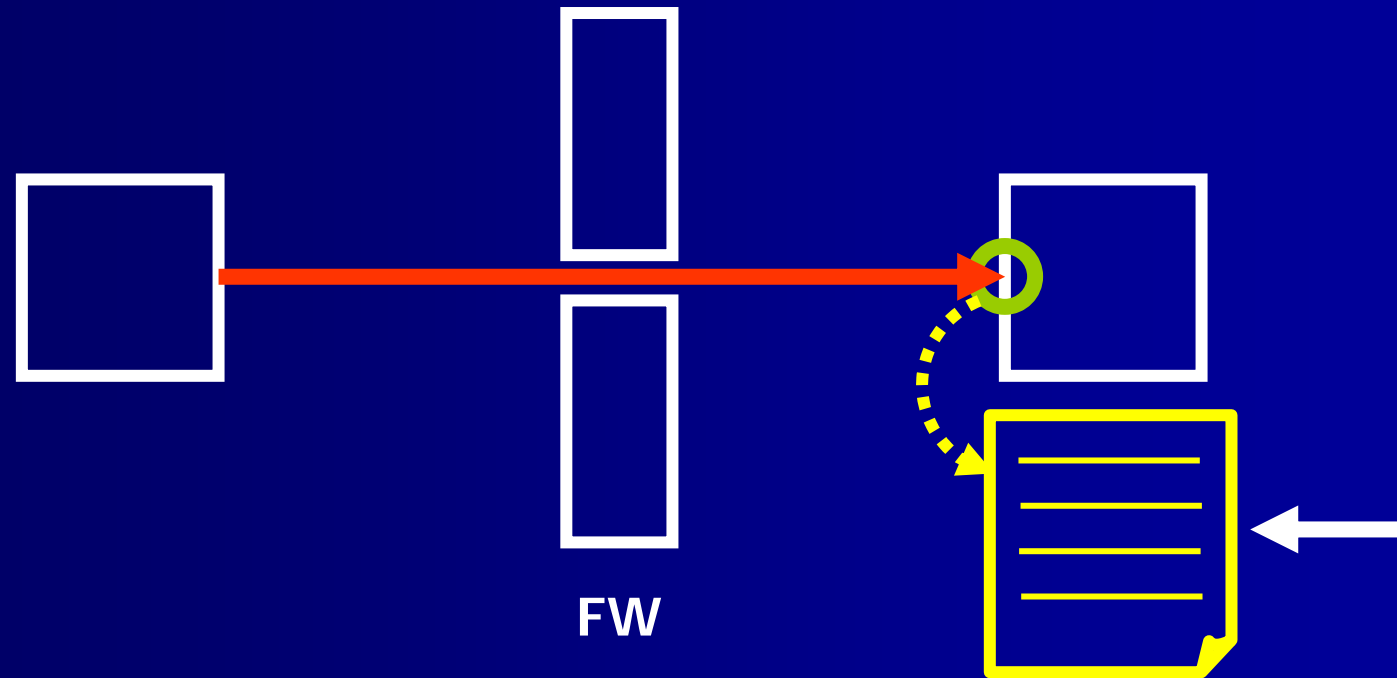
Möglichkeit zur Ausführung beliebigen Codes im Kontext des ausgenutzten Prozesses (IV) ...

Payload – getpeername



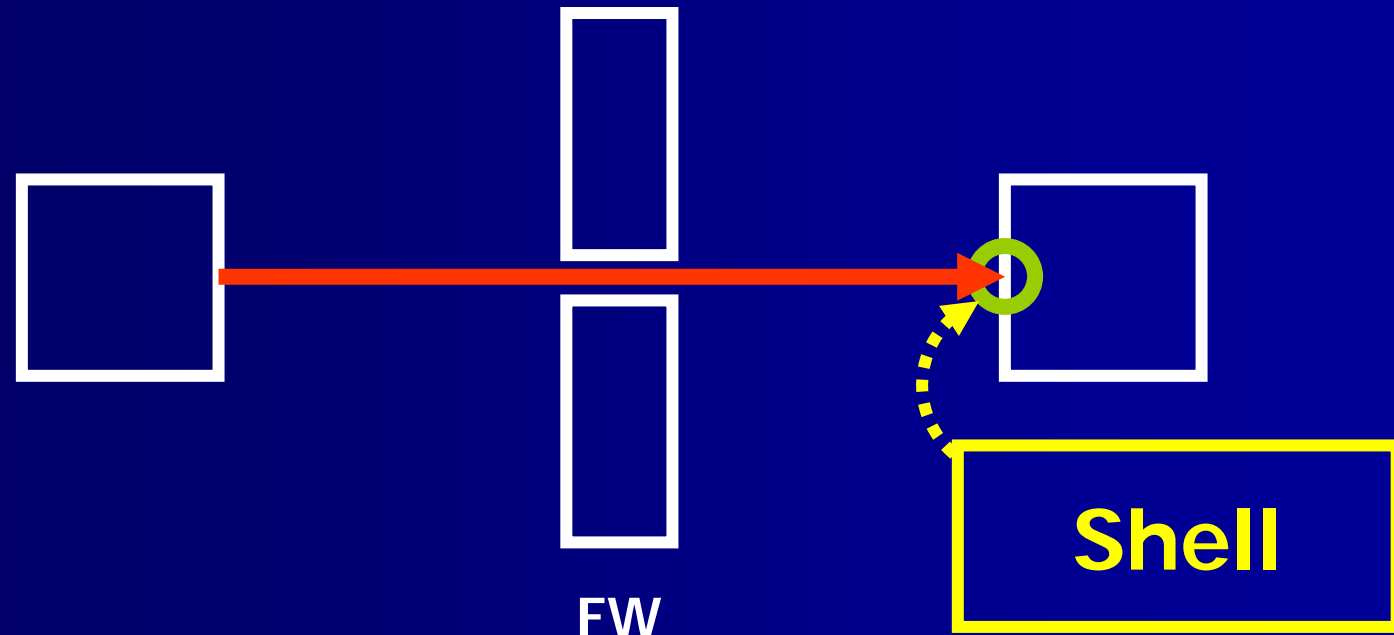
1. Durchsuche den per-process descriptor table (alle fd's des ausgenutzten Prozesses)

Payload – getpeername



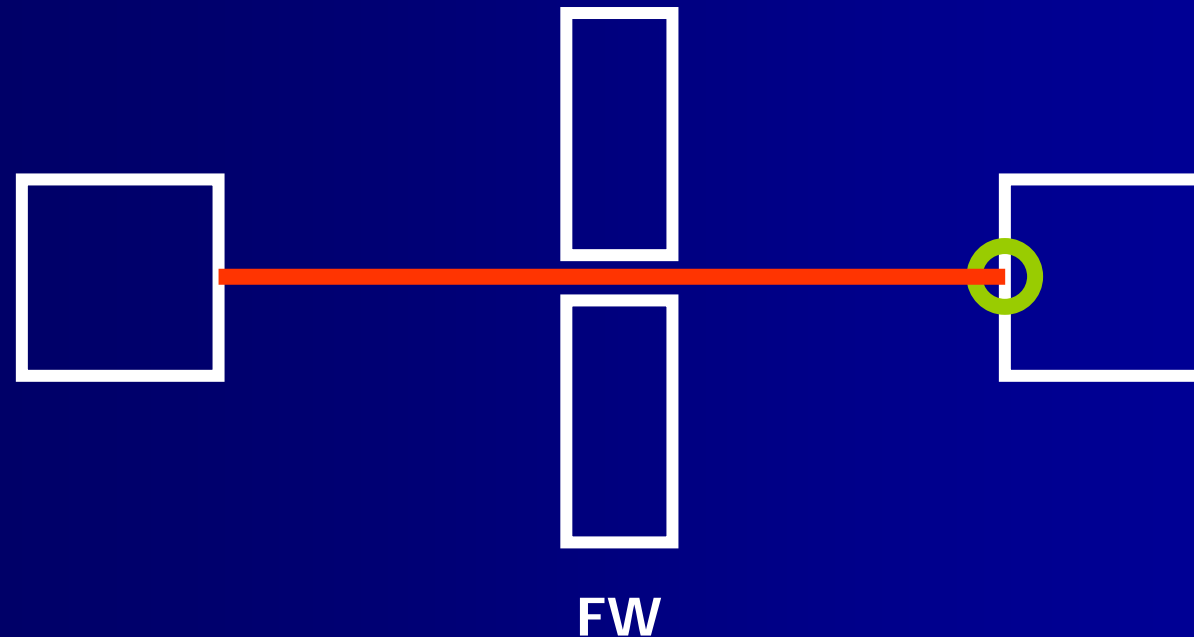
2. Suche den socket descriptor mit meinem TCP Source Port heraus

Payload – getpeername



3. Dupliziere `stdin`, `stdout` und `stderr`
4. Öffne die Shell/cmd.exe

Payload – getpeername

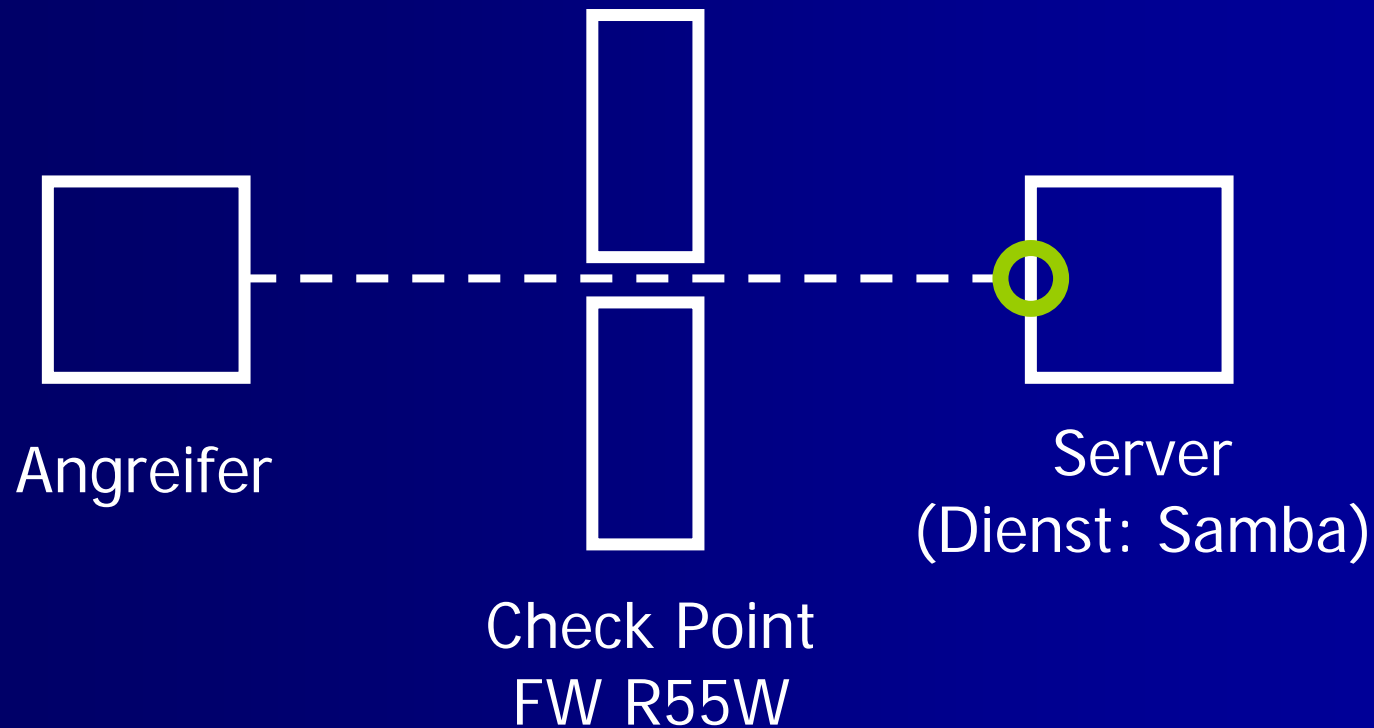


Ergebnis: Transparente Weiternutzung des Sockets → Anti stateful Firewall Technik

Demo 1

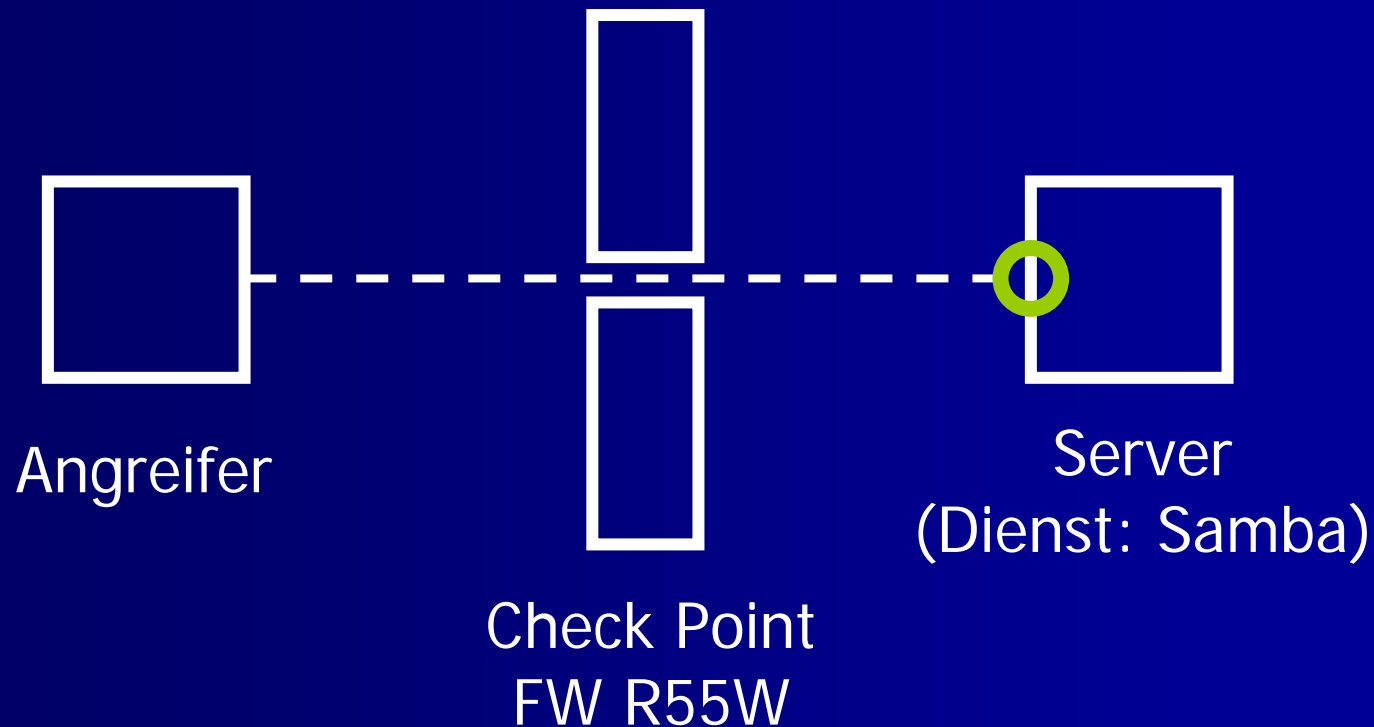
Socket reuse (getpeername)

Demo 1 – Aufbau



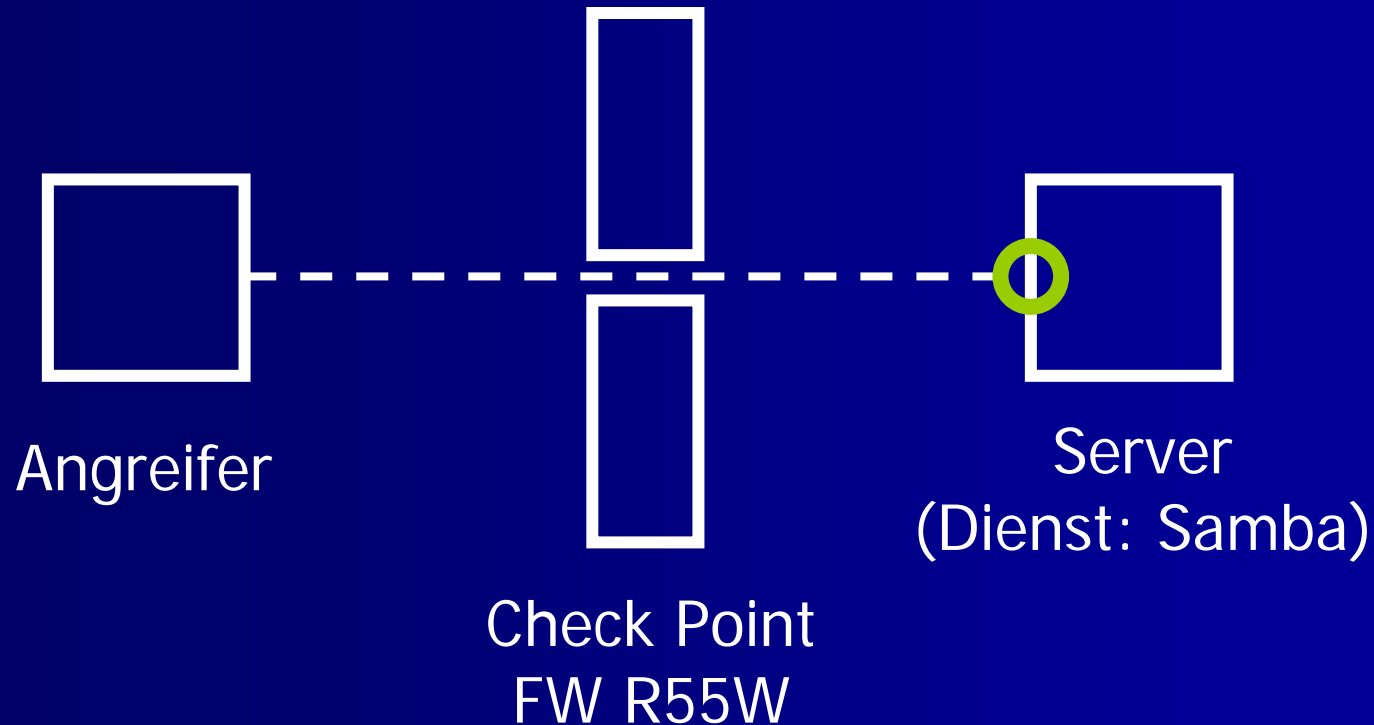
Die Firewall blockt alles außer Port 139 eingehend.

Demo 1 – Aufbau



Der Samba Dienst beinhaltet eine Buffer Overflow Schwachstelle (CAN-2003-0201).

Demo 1 – Aufbau



Ausnutzung dieses Overflows mittels Bind Port und Socket Reuse (getpeername) Payloads.

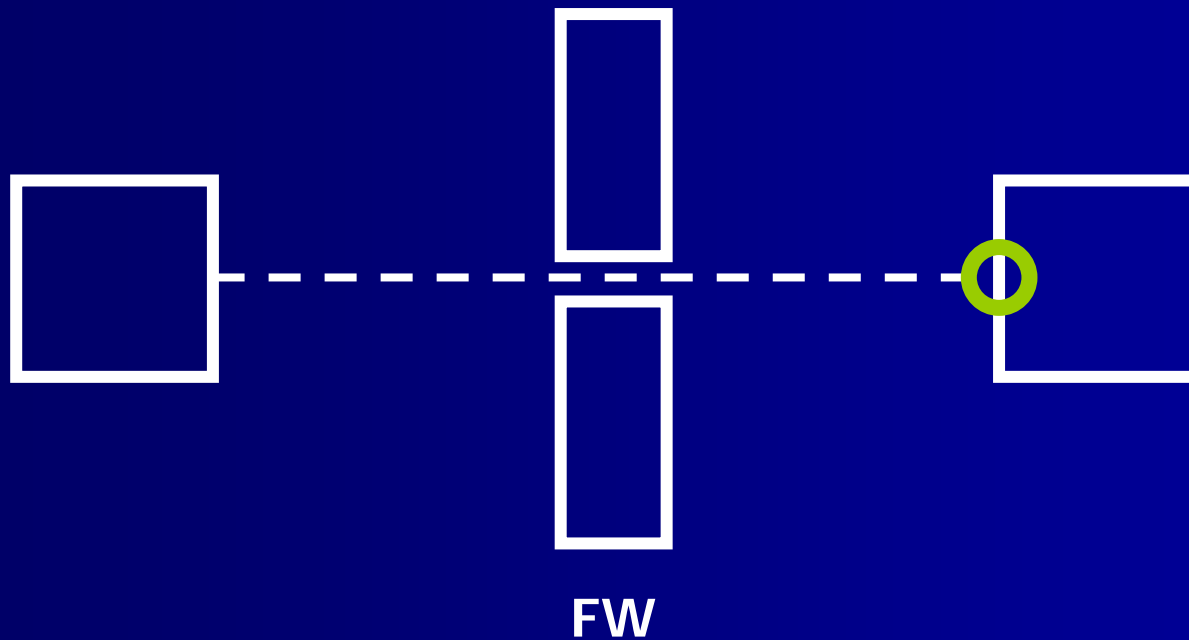
Payload – Anti stateful perimeter FW

- **Vorteile von getpeername:** kleiner Code, einfach zu implementieren, robust
 - **Nachteile von getpeername:** Proxies, NAT
- **Real Life:** ALIAScode, GOcode ...

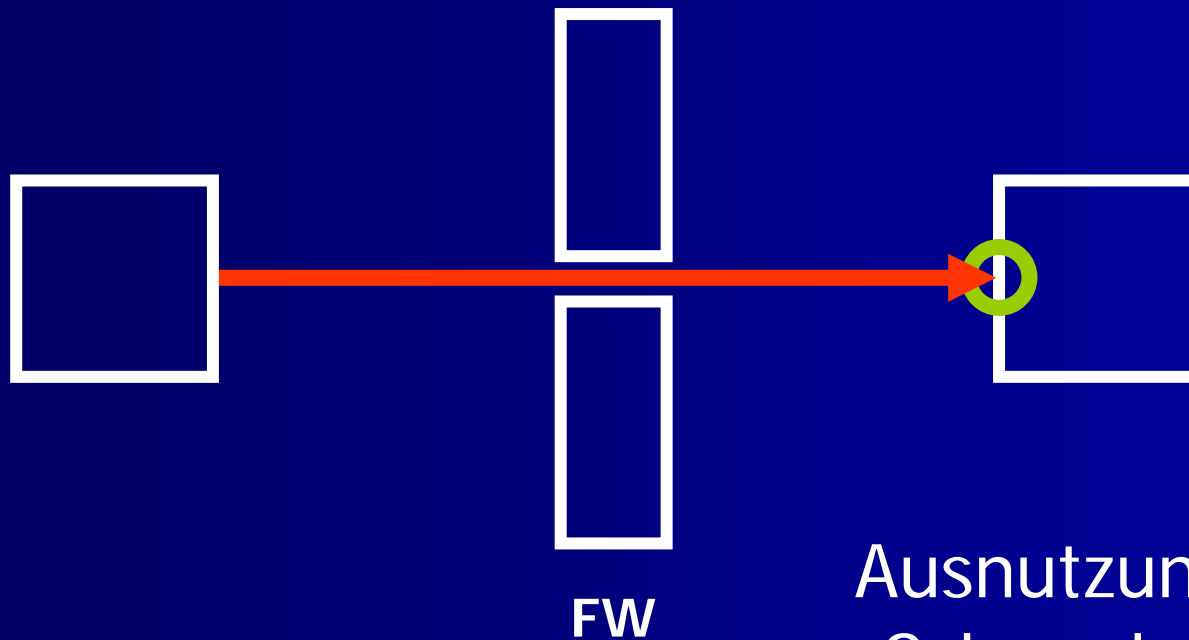
Payload – (Deep) Protocol/Packet Inspection

- **Nächste Hürde:** (Deep) Protocol/Packet Inspection
 - Firewalls (AI, WI), NIDS/NIPS
 - Prüfung des verwendeten Protokolls
 - RFC Konformität
 - ...
- Kommunikation mit Shell wird zunächst unterbunden

Payload – (Deep) Protocol/Packet Inspection

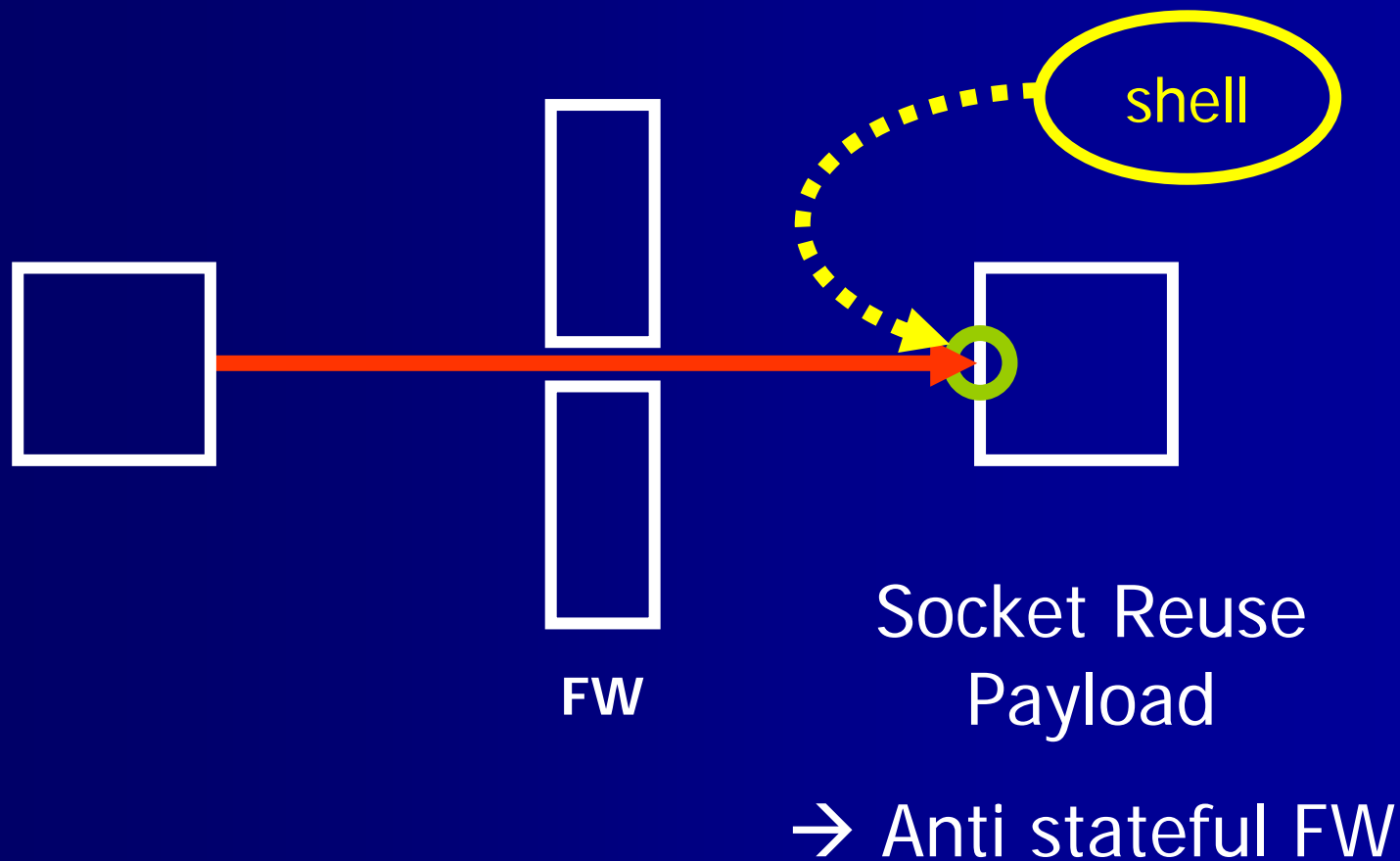


Payload – (Deep) Protocol/Packet Inspection

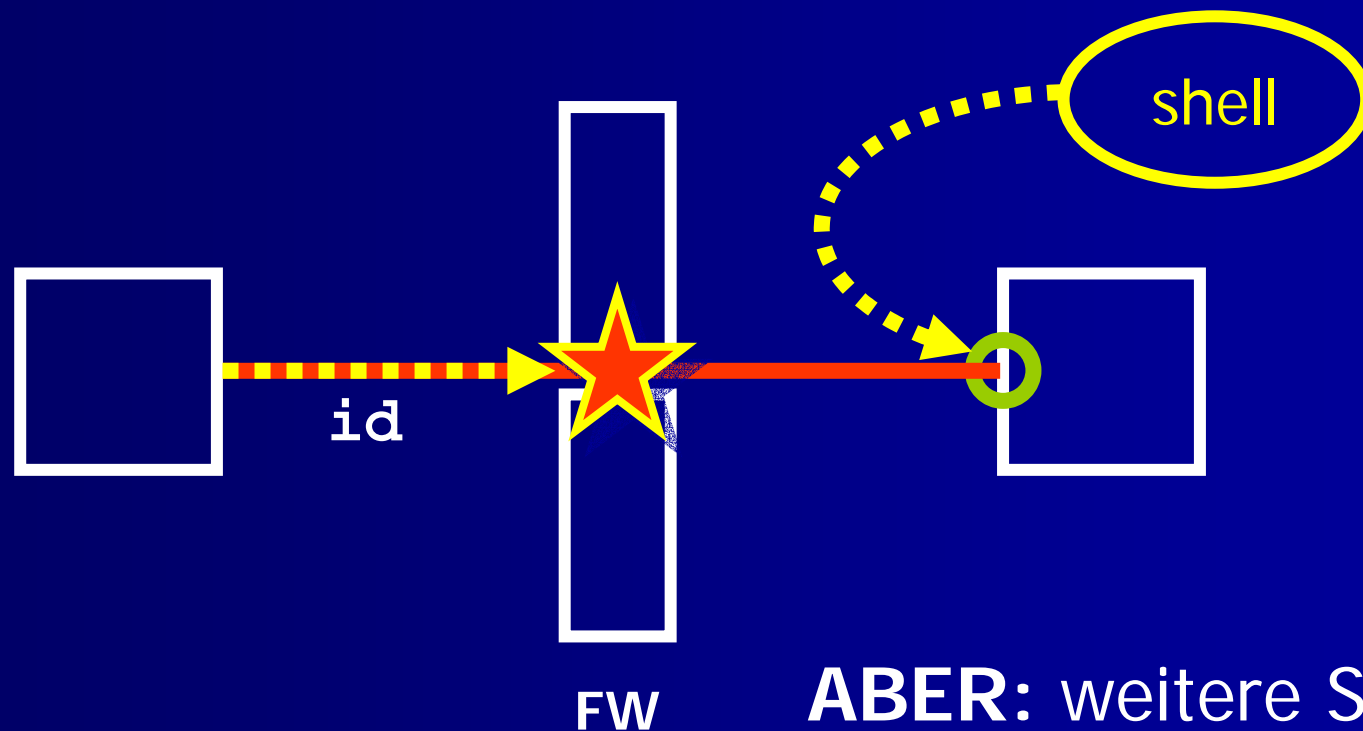


Ausnutzung der
Schwachstelle
funktioniert!

Payload – (Deep) Protocol/Packet Inspection



Payload – (Deep) Protocol/Packet Inspection

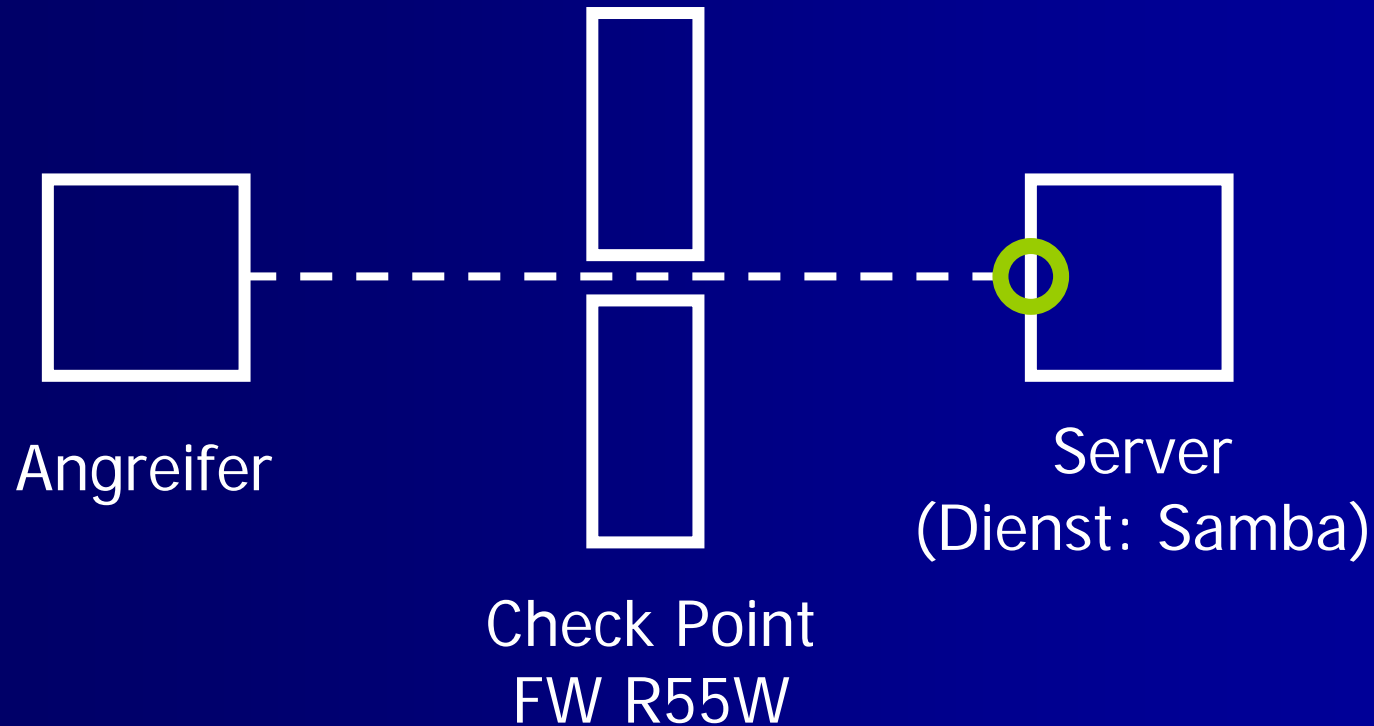


ABER: weitere Shell-Kommunikation wird geblockt

Demo 2

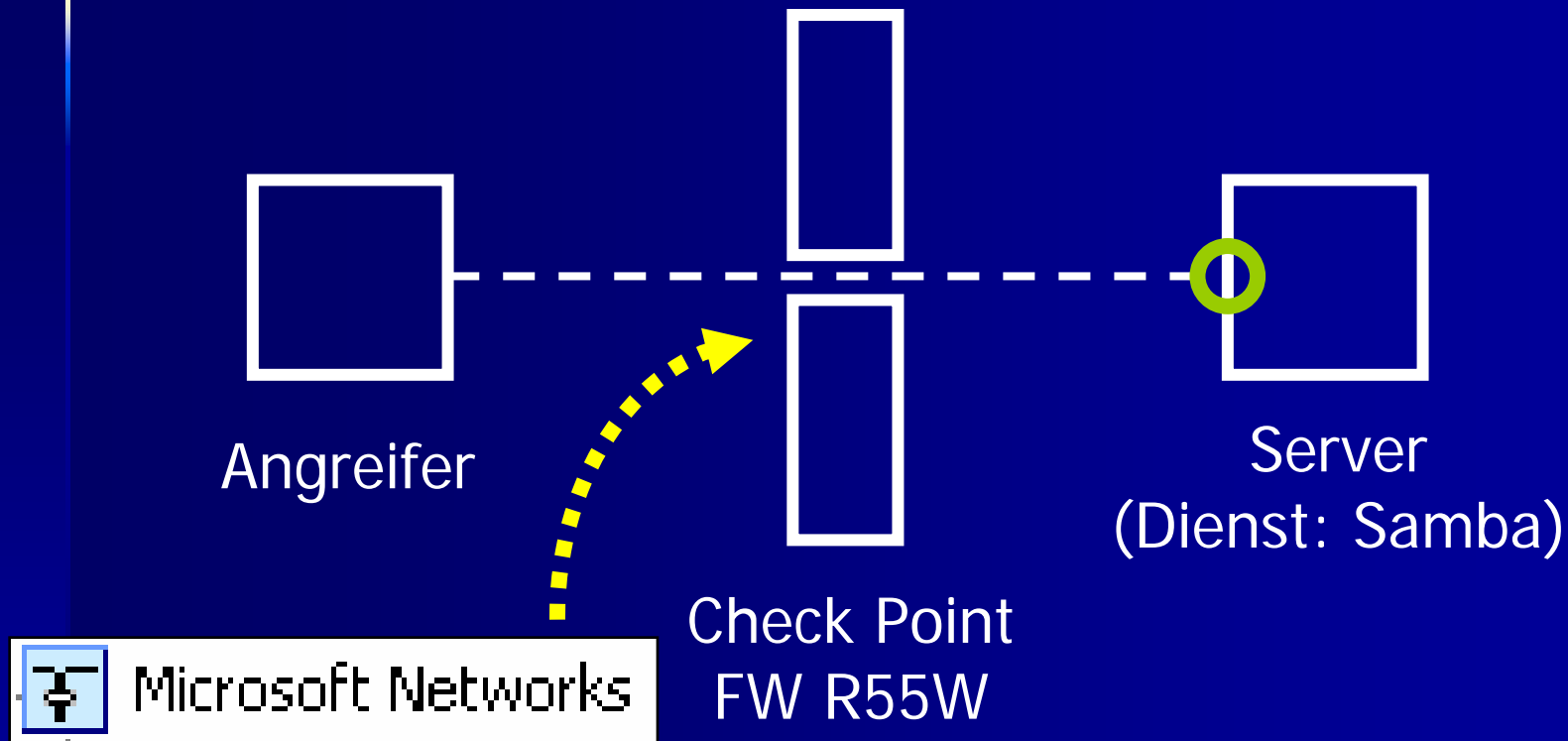
(Deep) Protocol/Packet Inspection

Demo 2 – Aufbau



Die Firewall blockt alles außer Port 139 eingehend.

Demo 2 – Aufbau

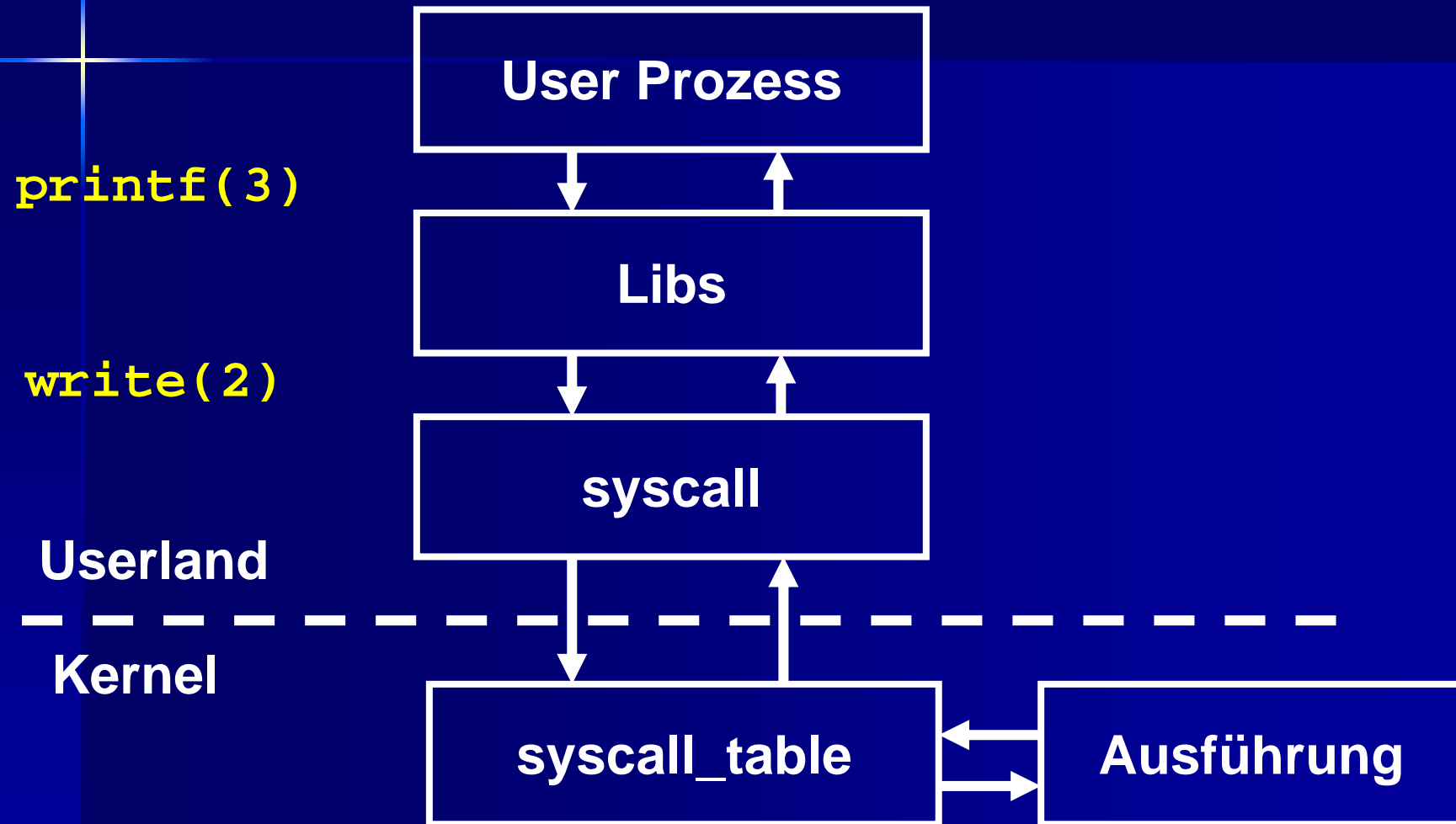


AI Funktionalität der Check Point wird aktiviert.

Payload – Protocol Tunneling

- **Problem:** Protokollanalyse
- **Lösung:** Kapselung der Kommunikation in das entsprechende Protokoll
 - Klassischer Shellcode somit erstmal unbrauchbar!
 - **Mögliche Techniken:** ALIAScode („normale“ Shell-Kommunikation weiterhin möglich ;), Syscall Redirection, Syscall Proxying

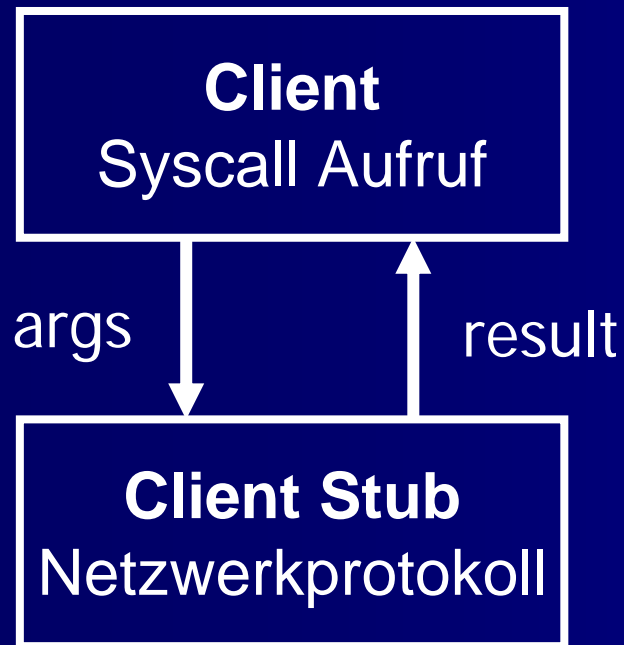
Syscalls



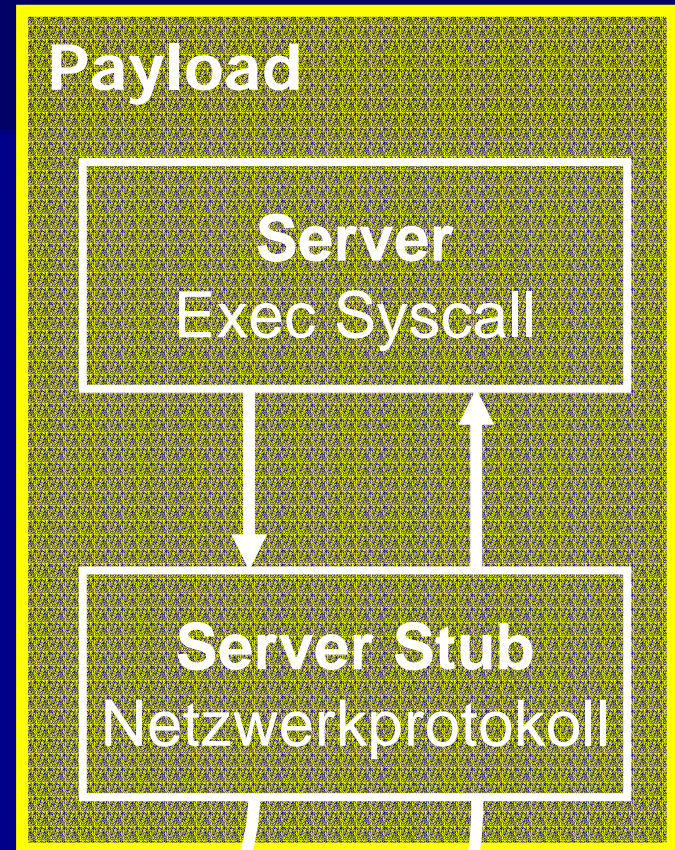
Syscall Redirection/ Proxying

- Payload, der eine Remote-Ausführung von Syscalls erlaubt
- RPC like
- Unterteilung in Client und Server

Syscall Redirection/ Proxying



Netzwerk



Syscall Redirection/ Proxying

- (schlanker) Server – (dicker) Client
- **Ein Vorteil:** Völlige Kontrolle über das verwendete Netzwerkprotokoll!
 - Protocol Tunneling
- Anti (Deep) Protocol/Packet Inspection

Gen1 Zecke

Grundfunktionalität

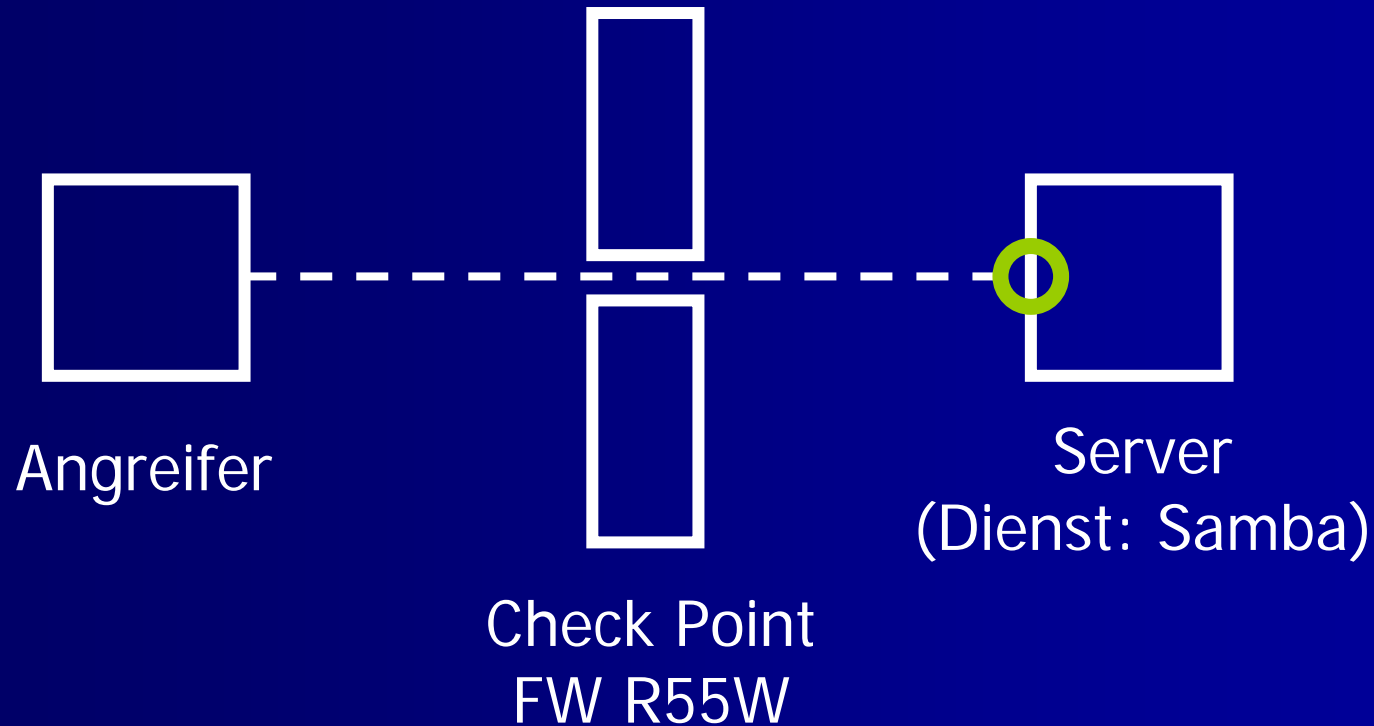
Gen1 Zecke

- Konkrete Syscall Redirection Implementation
- Momentan unterstützte Protokolle: SMB/CIFS, HTTP(s)
- Ermöglicht u.a. die gezielte Umgehung von stateful FWs und (Deep) Protocol/Packet Inspection

Demo 3

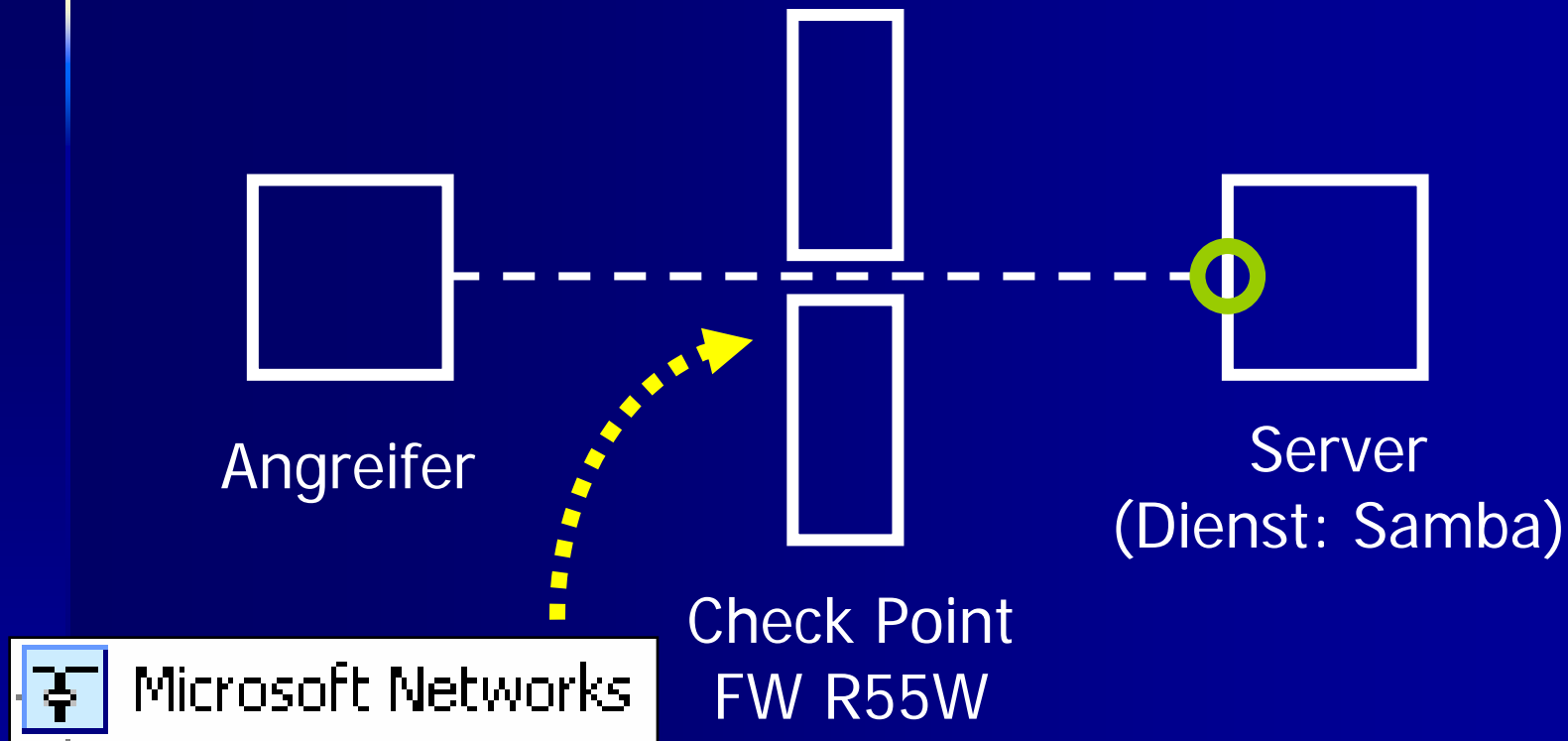
Gen1 Zecke

Demo 3 – Aufbau



Die Firewall blockt alles außer Port 139 eingehend.

Demo 3 – Aufbau



AI Funktionalität der Check Point wird aktiviert.

Gen1 Zecke

Advanced Features

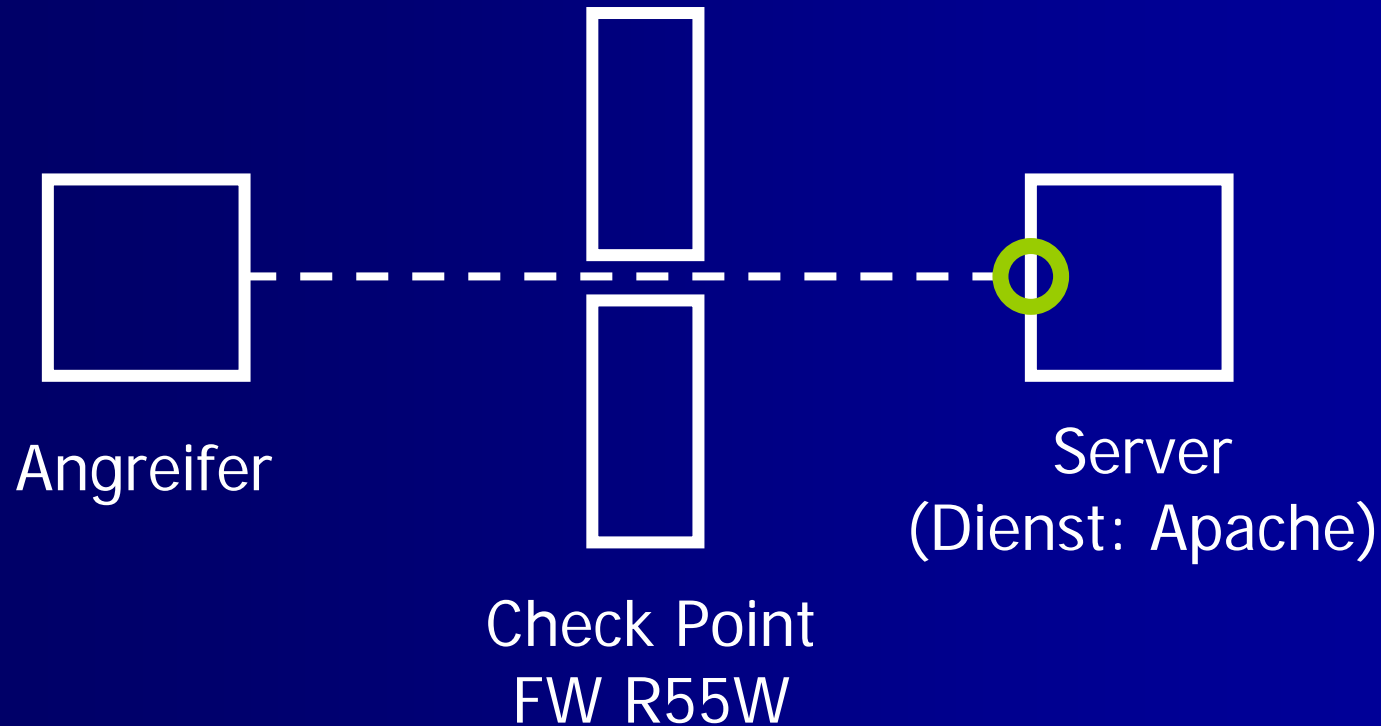
Weitere Widrigkeiten ...

- Neben stateful Firewalling und (Deep) Protocol/Packet Inspection gibt es noch weitere Widrigkeiten für einen Angreifer
 - Privilege Escalation
 - Download/Upload von (beliebigen) Dateien
 - Host Hopping
 - Forensic, Honeypots etc.
- Die Gen1 Zecke besitzt einige Features, um diesen Problemen zu begegnen

Demo 4

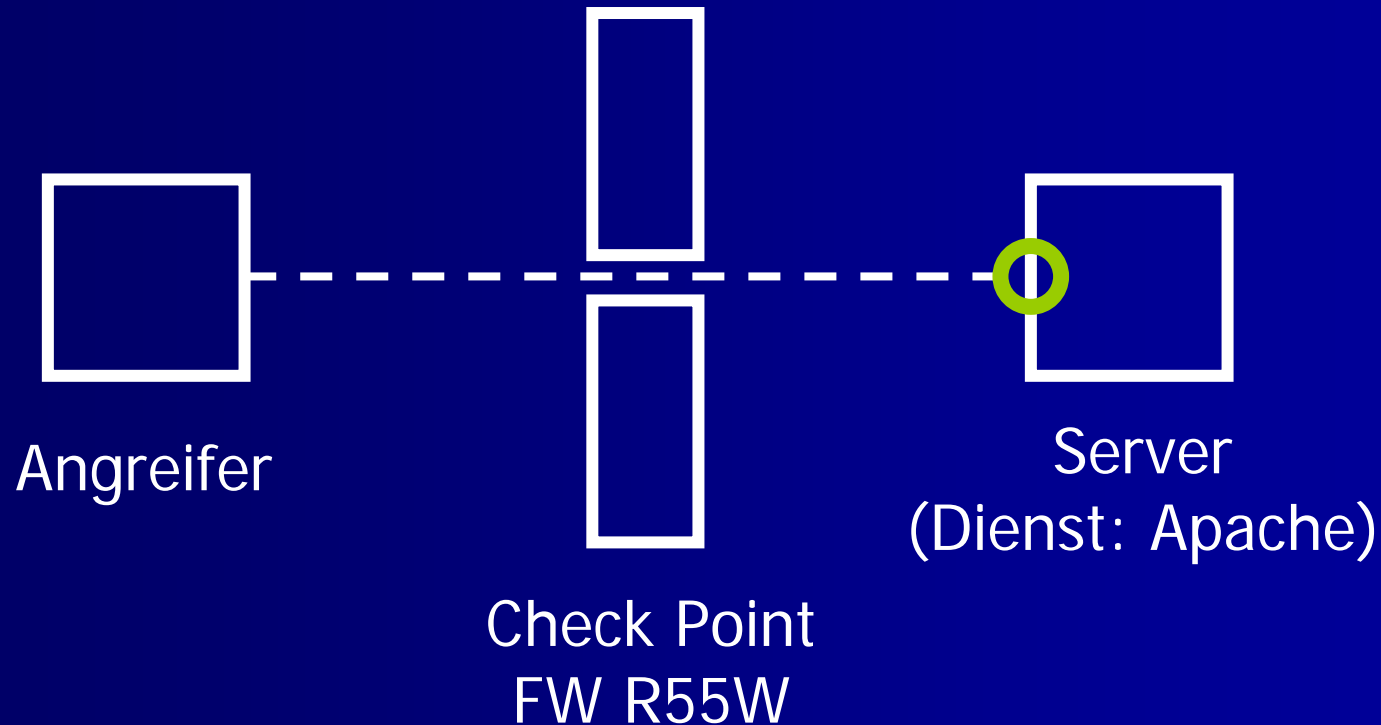
Gen1 Zecke Advanced Features

Demo 4 – Aufbau



Die Firewall blockt alles außer Port 80/443 eingehend.

Demo 4 – Aufbau



Der Apache Dienst beinhaltet eine Buffer Overflow Schwachstelle (CAN-2002-0656).

Zusammenfassung – Gen1 Zecke

- Erfolgreiche Umgehung von:
 - Stateful Inspection
 - (Deep) Protocol/Packet Inspection
 - Vielen Forensic/Honeypot Techniken
 - Dedizierten Filtern (z.B. Check Points Malicious Code Protector → Ø Day Knowledge ;)

Zusammenfassung – Gen1 Zecke

- Erleichterung von:
 - Privilege Escalation
 - Datei Download/Upload
 - Host Hopping

Code Veröffentlichung

- Gen1 Zecke Code nicht public
- Kein buntes „klick ich mal hier, klick ich mal da“ Security-Rockstar-Pen-Test-Produkt™
- FW/NI[D,P]S bypass, Anti Forensics

Ausblick

Gen1 Zecke

- Implementierung weiterer Protokolle (SMTP, DNS, ...)
- Implementierung weiterer remote sowie local Exploits
- Portierung auf weitere Plattformen
- Umgehung weiterer dedizierter Filter (Netz/Host)
- Advanced Honeypot Detection, Anti Forensic Features
- Weitere Verbesserungen
 - Vorschläge und Feature Requests sind erwünscht (→ Roundtables ;)

Fragen ?