Packet Crafting

Objectives

- Introduce and explore the reasons for crafting (injecting) packets
- Leverage the scapy framework to constructed crafted packets

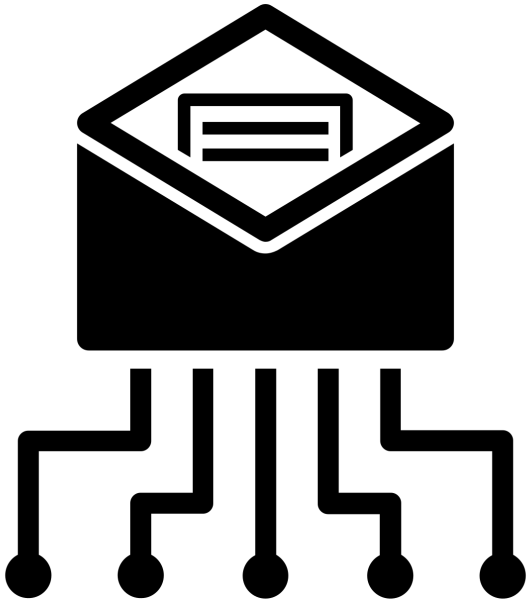
References

- https://scapy.net
- Phrack Volume 7, Issue 49
- RFCs 4987, 3833

What is Packet Crafting?

- Creating network packets with specific characteristics to test or exploit vulnerabilities in systems.

- Step to forging a packet:
  - Create a raw socket
  - Create packet headers for delivery
  - Add data
  - Compute checksums
  - Send the packet to raw socket

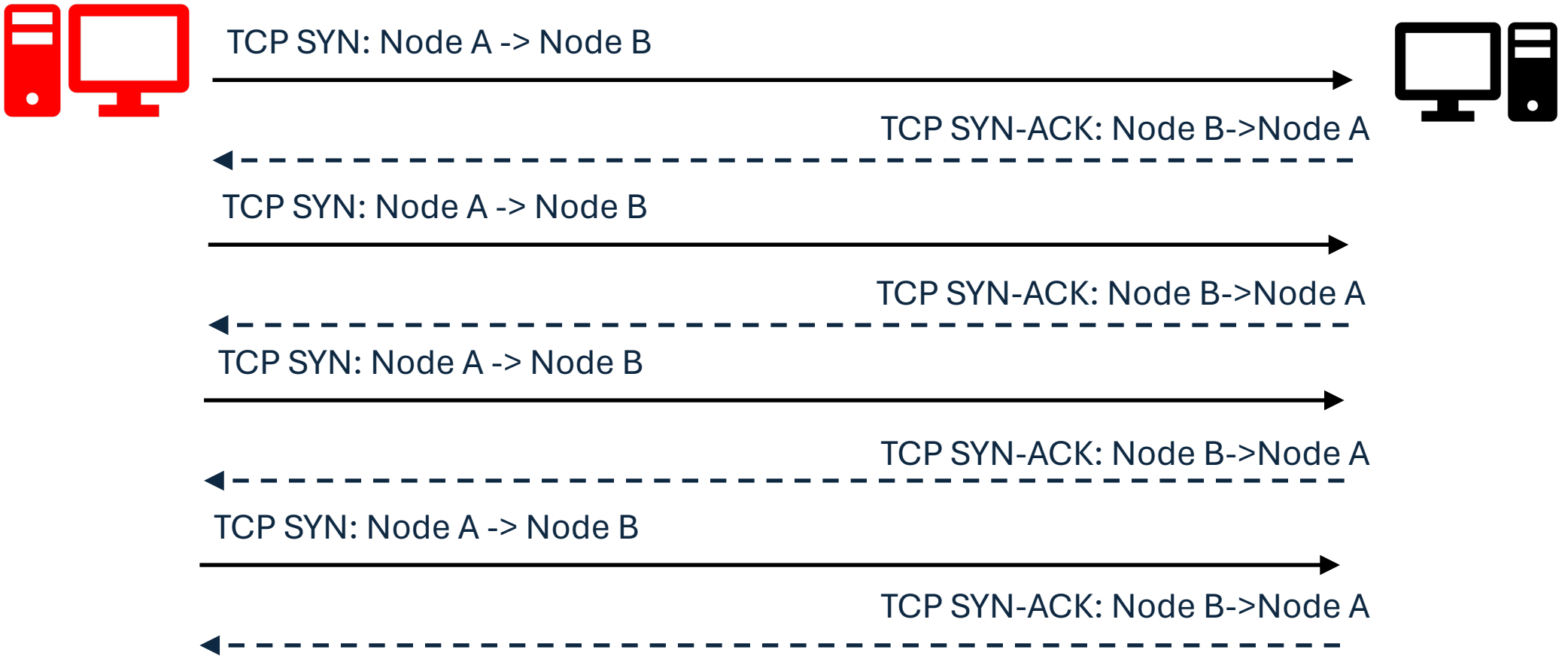


Why Would Want to Inject Packets

- Spoofing for hijack attacks (dns, arp spoofing)
- Denial of service attacks (syn flooding, reflective dns)
- Covert channels to exfiltrate data (covert icmp, dns)

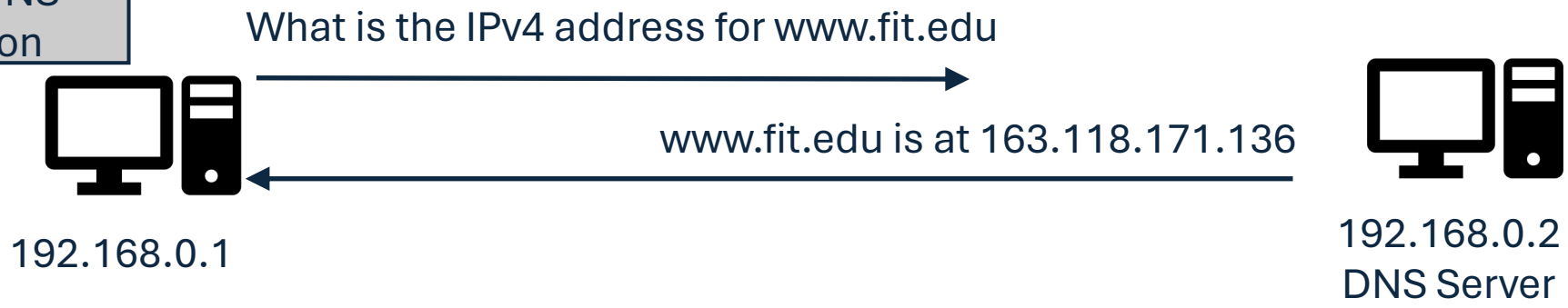
TCP Syn Flooding

The SYN flooding attack is a denial-of-service method affecting hosts that run TCP server processes. The attack takes advantage of the state retention TCP performs for some time after receiving a SYN segment to a port that has been put into the LISTEN state. The basic idea is to exploit this behavior by causing a host to retain enough state for bogus half-connections that there are no resources left to establish new legitimate connections. (RFC 4987)

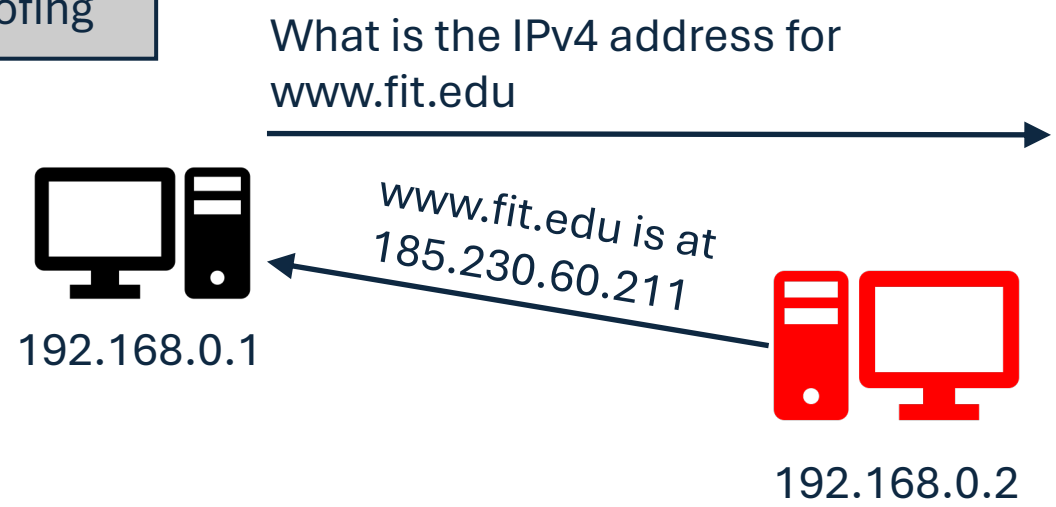


DNS-Spoofing (DNS Cache Poisoning)

Normal DNS Resolution



DNS Spoofing



Some of the simplest threats against DNS are various forms of packet interception: monkey-in-the-middle attacks, eavesdropping on requests combined with spoofed responses that beat the real response back to the resolver, and so forth. In any of these scenarios, the attacker can simply tell either party (usually the resolver) whatever it want that party to believe. While packet interception attacks are far from unique to DNS, DNS's usual behavior of sending an entire query or response in a single unsigned, unencrypted UDP packet makes these attacks particularly easy for any bad guy with the ability to intercept packets on a shared or transit network. (RFC 3833)

192.168.0.2
DNS Server

Covert Channels

```
Ord converts a character ‘A’
to a numeric representation

>>> x =  ord(‘A’)
>>> x
41
```

```
                        Volume Seven, Issue Forty-Nine

                                File 06 of 16

                              [ Project Loki ]

                          whitepaper by daemon9 AKA route
                         sourcecode by daemon9 && alhambra
                              for Phrack Magazine
                         August 1996 Guild Productions, kid

                comments to route@infonexus.com/alhambra@infonexus.com


                    --[ Introduction ]--


        Ping traffic is ubiquitous to almost every TCP/IP based network and
subnetwork.  It has a standard packet format recognized by every IP-speaking
router and is used universally for network management, testing, and
measurement.  As such, many firewalls and networks consider ping traffic
to be benign and will allow it to pass through, unmolested.  This project
explores why that practice can be insecure.  Ignoring the obvious threat of
the done-to-death denial of service attack, use of ping traffic can open up
covert channels through the networks in which it is allowed.

        Loki, Norse God of deceit and trickery, the 'Lord of Misrule' was
well known for his subversive behavior.  Inversion and reversal of all sorts
was typical for him.  Due to it's clandestine nature, we chose to name this
project after him.

        The Loki Project consists of a whitepaper covering this covert channel
in detail.  The sourcecode is not for distribution at this time.


                    --[ Overview  ]--


        This whitepaper is intended as a complete description of the covert
channel that exists in networks that allow ping traffic (hereon referred to
in the more general sense of ICMP_ECHO traffic --see below) to pass.  It is
organized into sections:

        Section I.      ICMP Background Info and the Ping Program
        Section II.     Basic Firewall Theory and Covert Channels
        Section III.    The Loki Premise
        Section IV.     Discussion, Detection, and Prevention
        Section V.      References

(Note that readers unfamiliar with the TCP/IP protocol suite may wish to first
read ftp://ftp.infonexus.com/pub/Philes/NetTech/TCP-IP/tcipIp.intro.txt.gz)
```

Phrack Volume 7, Issue 49

Embed data inside ICMP echo requests

```
>>> pkt = IP()/ICMP()/Raw(load="SENDFLAG")
```

```
>>> pkt = IP(id=ord('S'))/ICMP()
>>> pkt = IP(id=ord('E'))/ICMP()
>>> pkt = IP(id=ord('N'))/ICMP()
>>> pkt = IP(id=ord('D'))/ICMP()
>>> pkt = IP(id=ord('F'))/ICMP()
>>> pkt = IP(id=ord('L'))/ICMP()
>>> pkt = IP(id=ord('A'))/ICMP()
>>> pkt = IP(id=ord('G'))/ICMP()
```

Packet Injection: ICMP Ping Packet

```
from scapy.all import *
pkt = IP(src="192.168.1.20",dst="192.168.1.7")/ICMP(type=8)
send(pkt)
```

- Step to forging a ping packet:
  - Create a raw socket
  - Create an IP header
  - Set the source and destination
  - Create an ICMP header

    Set the type of ICMP packet

    Send the packet to raw socket

Packet Injection: TCP Packet

Scapy	Field	Explanation
sport	Source Port	Identifies the TCP port sending the packet
dport	Destination Port	Identifies TCP port receiving the packet
seq	Sequence Number	Used for ordering packets
ack	Acknowledgement Number	Used for acknowledging received packets
dataofs	Data Offset	Explains where the data begins
flags	TCP Flags	Used to managed the state of the connection
chksum	Checksum	Used for verifying packet integrity
window	Window	Used to manage size of the connection
urgptr	Urgent Pointer	Indicates should be processed immediately
options	options	

```
pkt = IP(dst= dst="192.168.1.2")/TCP(dport=31337)
# send a TCP packet to port 31337 and IP address 192.168.1.2
```

Common TCP Ports

Port	Purpose
20, 21	File Transfer Protocol (FTP)
22	Secure Socket Shell (SSH)
23	Telnet
25	Simple Mail Transfer Protocol (SMTP)
53	Domain Name System (DNS)
80	Hypertext Transfer Protocol (HTTP)
445	Server Message Block (SMB)
3389	Remote Desktop Protocol (RDP)
5900	Virtual Network Computing (VNC)

Packet Injection: UDP Packet

Scapy	Field	Explanation
sport	Source Port	Identifies the TCP port sending the packet
dport	Destination Port	Identifies TCP port receiving the packet
len	Length	Used for ordering packets
chksum	Checksum	Used for verifying packet integrity

```
pkt = IP(dst= dst="192.168.1.3")/UDP(dport=1337)
# send a TCP packet to port 1337 and IP address 192.168.1.3
```

Common UDP Ports

Port	Purpose
53	Domain Name System (DNS)
69	Trivial File Transfer Protocol (TFTP)
123	Network Time Protocol (NTP)
514	System Logging (Syslog)
1900	Simple Service Discovery Protocol (SSDP)