BGP Vulnerability Testing: Separating Fact from FUD v1.1

Sean Convery (sean@cisco.com)
Matthew Franz (mfranz@cisco.com)

Cisco Systems
Critical Infrastructure Assurance Group (CIAG)
http://www.cisco.com/go/ciag

Agenda

- Introduction
- BGP Vulnerability Testing
- Analysis of BGP Best Practices
- “Active” ISP Survey
- Conclusions

If you believe what you read...

- BGP is...highly vulnerable to a variety of attacks due to the lack of a scalable means of verifying the authenticity and authorization of BGP control traffic. - S-BGP Website[1]
- Any outsider can inject believable BGP messages into the communication between BGP peers and thereby inject bogus routing information or break the peer to peer connection. - draft-murphy-bgp-vuln-02.txt[2]
- Outsider sources can also disrupt communications between BGP peers by breaking their TCP connection with spoofed RST packets. - draft-murphy-bgp-protect-01.txt[3]
- The border gateway protocol...is rife with security holes and needs to be replaced, a security consultant warned. - news.com[4]

Research Objectives

- Conduct a systematic analysis of BGP vulnerabilities based on testing of multiple implementations—current assumptions are largely speculative
- Measure the effectiveness of best practices in mitigating likely attacks—in the near term, hardening vendor implementations and applying best practices is all we have
- Collect data on the security posture of real-world routers and BGP implementations

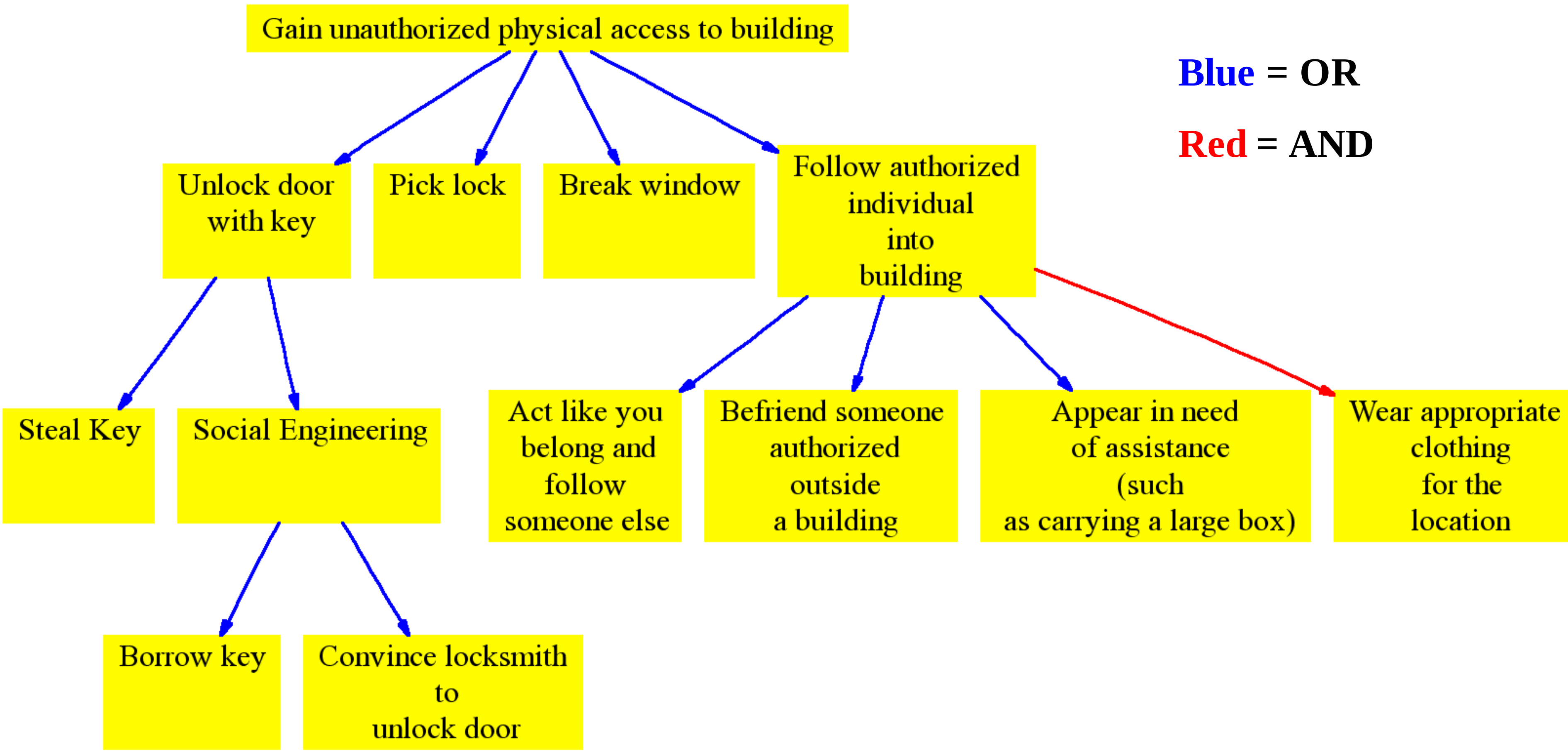
Methodology

- Conduct BGP-relevant TCP attacks
- Evaluate robustness of BGP parsers using fuzz-testing (similar to PROTOS)
- Conduct selected attacks in BGP Attack Tree[6] under the following conditions:
  - Blind Attacker / Non-Blind Attacker / Compromised Router
  - BGP best practices ON and OFF
- Conduct an "Active" survey of ISP best practices
  - Probe Admin ports (22/23/80)
  - Identify Permissive BGP speakers (179)

Vulnerabilities & Vulnerability Disclosure

- Three types of vulns are considered in this talk:
  - Design – does what it is supposed to do
  - Implementation – bug based on coding error
  - Misconfiguration – weak passwords, failure to use security features, block admin ports, etc.
- Vendors have been notified of all implementation flaws
- CERT/CC has been given a set of BGP test cases to distribute to vendors
- No vendors will be identified in this talk

Attack Tree Example (Graphical)

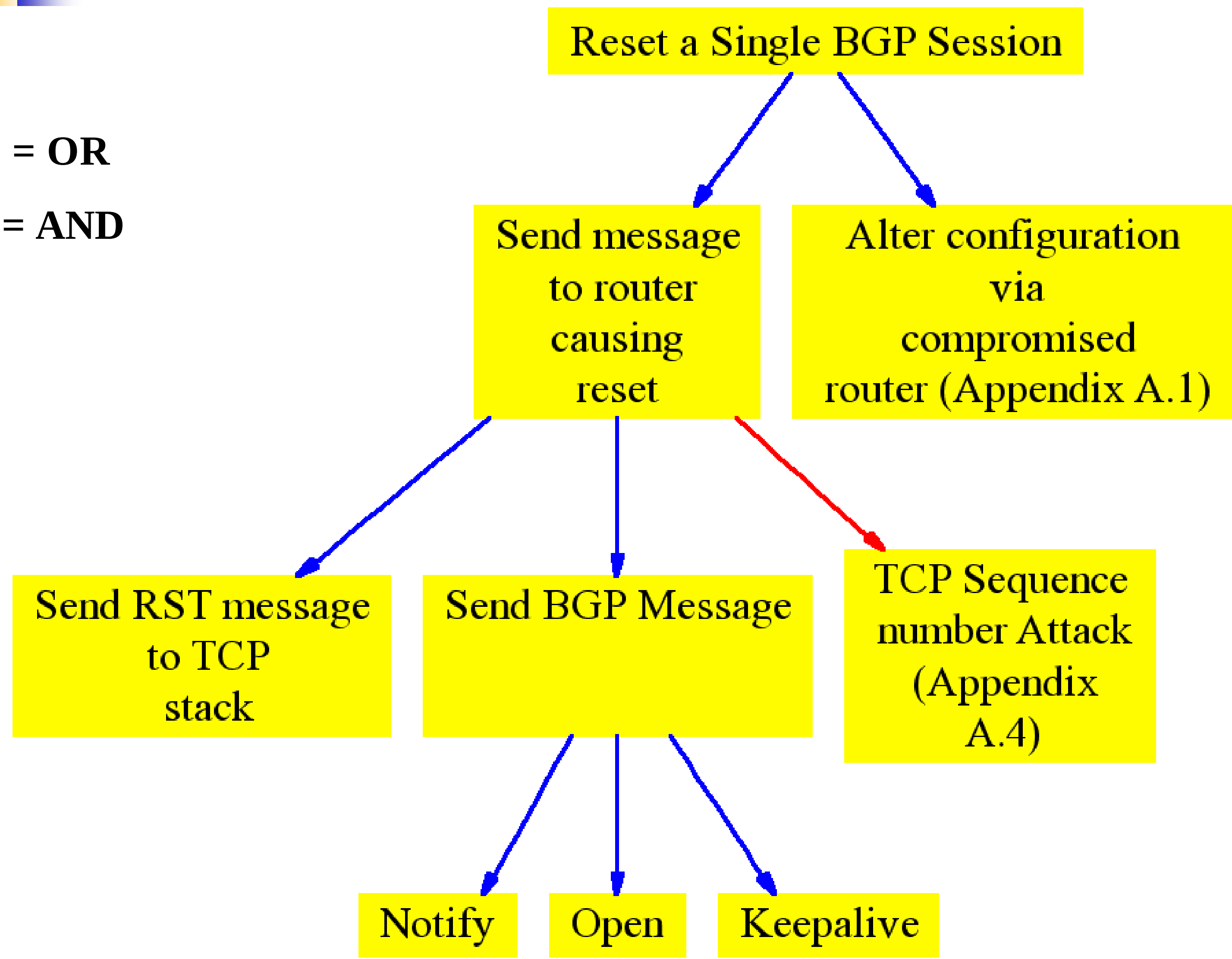


Graphic tree representations are generated from the source attack tree.

Reset a Single BGP Session (Graphical)

Blue = **OR**

Red = **AND**



Building on draft-convery-bgpattack-00.txt[6]

Atomic Goals

- "Compromise" MD5 Auth
- Establish unauth BGP session
- Originate unauth prefix into peer
- Change path pref of a path
- DoS BGP Session
- Spoof BGP Message

Supp. Atomic Goals

- Compromise router
- DoS router
- MITM attack
- TCP Sequence # attack
- Sniff traffic

Attack Scenarios

- Disable critical portions of Internet...
- Disable single-homed AS
- Disable multi-homed AS
- Blackhole traffic

Agenda

- Introduction
- BGP Vulnerability Testing
- Analysis of BGP Best Practices
- "Active" ISP Survey Results
- Conclusions

BGP & TCP Testing

- TCP/BGP Connection Behavior*
- TCP Resource Exhaustion*
- TCP Resets
- MD5 (RFC 2385) Attacks
  - MD5 Dictionary Attack
  - MD5 DoS*
- Update Flooding*
- BGP Route Insertion (TCP Hijack)
- BGP Peer Hijack (ARP Spoofing)
- Malformed BGP Messages*
  - OPEN
  - UPDATE

*Conducted against multiple implementations

Testing BGP Implementations

- Goal: sample the responses of a variety of implementations to known and potential attacks
- 7 different BGP implementations were evaluated using “default” BGP configs
- When present, parenthetical notations in test result slides identify the number of implementations that exhibited that behavior
- Statistics (times/CPU utilization, etc.) were on a lightly loaded test network, so impact of certain attacks is likely to be different (greater)

Tools We Used

- Packet Generation & Injection
  - Hping[7], Nemesis-tcp[8], Netcat[9], Naptha (synsend)[10]
- Bgpcrack*
  - MD5 attacks
- TCP Test Tool (ttt)*
  - Sequence number guessing, MD5 flooding
- Tcphijack*
  - BGP route insertion
- Dsniff[11]
  - ARP spoofing
- Protocol Independent Fuzzer (pif)*
  - Invalid Message Generation
- Pyupdate/Pyopen*
  - Valid message generation
- "Active" ISP Survey Tools*

Some of these new tools available at:
http://www.cisco.com/security_services/ciag/tools

Connection Establishment Tests

- Identify implementation behavior during session establishment—*what is necessary for successful peer negotiation? How far can the attacker get?*
- How much of the message is processed and how far the state can be advanced determines risk and impact of attacks:
  - Initial SYN – SYN flooding
  - Connect() – ESTABLISHED/FIN_WAIT flooding
  - BGP OPEN – Remote Identification/Malformed messages
  - UPDATE – Route insertion/deletion

Connection Establishment (TCP)

- No standard behavior was observed across the implementations we tested
- Results varied, from least permissive (reject quietly) to most permissive (full 3-way handshake)
  - SYN from non-configured peer
    - Silent Drop (1)
    - RST-ACK (3)
    - SYN-ACK (3)
  - Spoofed SYN from configured peer (session est.)
    - RST-ACK (4)
    - SYN-ACK (3)

Connection Establishment (BGP)

- Test Results:
  - OPEN from non-configured peer
    - RST (6)
    - NOTIFICATION: OPEN Message Error/Authentication Failure (1)
  - OPEN from configured peer with invalid AS
    - NOTIFICATION: OPEN Message Error/Authentication Failure (2)
    - NOTIFICATION: OPEN Message Error Bad Peer AS (5)

Connection Establishment (BGP)

- Wildcards
  - Timeouts – delay between session renegotiation (especially after NOTIFICATION)
    - Delay of 1-3 minutes before new connection (4)
    - No timeouts (3)
  - Send OPEN immediately after reaching established state (1)
- No implementation allowed BGP OPENs with the wrong AS or from non-configured peer to reach BGP ESTABLISHED state—as a result, *TCP spoofing is required to inject data*

TCP Resource Exhaustion vs. BGP

- Goal: prevent new BGP sessions from being established or impact existing sessions
- Why: many BGP implementations are tightly integrated with TCP stacks and there may be “collateral damage”
- Should be the easiest to conduct and require the least amount of knowledge and access
  - SYN Flooding
  - ESTABLISHED Flooding
  - FIN_WAIT1 Flooding

SYN Flooding

- Exhaust number of sessions in SYN_RCVD state

```
Attacker# synsend 10.89.168.101 10.89.168.89 1
Randomizing port numbers.
Sending SYN packets.

Victim# netstat -an | grep --tcp
tcp  0 0 10.89.168.101:179 10.89.168.99:4189  SYN_RECV
tcp  0 0 10.89.168.101:179 10.89.168.99:8017  SYN_RECV
tcp  0 0 10.89.168.101:179 10.89.168.99:56477 SYN_RECV
tcp  0 0 10.89.168.101:179 10.89.168.99:41185 SYN_RECV
```

ESTABLISHED Flooding

- Stress peer establishment or overflow socket file descriptors

```
Attacker# synsend 10.89.168.101 10.89.168.89 1
Randomizing port numbers.
Sending SYN packets.

Attacker# srvr -SAa 10.89.168.10

Victim# netstat -an | grep --tcp
tcp  0 0 10.89.168.101:179 10.89.168.99:36601     ESTABLISHED
tcp  0 0 10.89.168.101:179 10.89.168.99:59545     ESTABLISHED
tcp  0 0 10.89.168.101:179 10.89.168.99:49340     ESTABLISHED
```

FIN_WAIT 1 Flooding

- Stress peer deletion or exhaustion of socket file descriptors

```
Attacker# synsend 10.89.168.101 10.89.168.89 1
Randomizing port numbers.
Sending SYN packets.

Attacker# srvr -SAfa 10.89.168.10



Victim# netstat -an | grep --tcp
tcp 0 1 10.89.168.101:179 10.89.168.99:35734      FIN_WAIT1
tcp 0 1 10.89.168.101:179 10.89.168.99:15142      FIN_WAIT1
tcp 0 1 10.89.168.101:179 10.89.168.99:56006      LAST_ACK
tcp 0 1 10.89.168.101:179 10.89.168.99:63718      LAST_ACK
```

TCP Resource Exhaustion vs. BGP Results

- Goal was to just impact TCP and as a result, BGP–we know there are infinite ways to kill a box (saturate links, punt to CPU, fill non-TCP queues, etc.)
- Impact to implementations that SYN/ACK peers (or when spoofed)
  - Up to 5-6 minute delay in BGP session establishment – peers under attack could negotiate outbound sessions with other peers
  - Moderately elevated CPU utilization and latency
  - No impact on existing sessions

TCP Resource Exhaustion Results

- The bottom line
  - An attacker would have to find a way to break the current session and SYN flood both peers (and possibly spoof the src, depending on the implementation) to cause significant impact
  - Implementations that allow state past SYN_RECVD may have issues—but ACLs can mitigate this—blind connect() spoofing is hard

TCP Resets (1/2)

- Various research [12], and [13] have found flaws in some implementations of TCP ISN selection. This should be a solved problem for most implementations though (did not repeat tests).
- Recent research [24] has shown that the TCP window size significantly reduces the problem space to conduct a successful blind attack.
- draft-ietf-tcpm-tcpsecure-00.txt [25] describes new techniques for overcoming vulnerabilities due to the TCP window size in current TCP stacks.
  - The draft outlines an approach to increase their difficulty by implemented a challenge/response between client and server. These improvements have been implemented in shipping code from Cisco and Juniper and are under consideration by several other vendors.

TCP Resets (2/2)

- Blind TCP seq. guessing is operationally impossible with a router using BCPs because with proper RFC 2827[14] filtering—the packet won't even reach the destination
- A successful TCP reset attack would need to be constantly repeated to keep a session down and would need to be duplicated on many routers to cause substantial impact to the Internet's routing tables
  - These attacks are noisy by design as the attacker will likely not know which side is the TCP client vs. server and some amount of guesswork is required, even in traditional TCP stacks
- More research is needed to determine whether blind RSTs (via guessing, even within a narrowed window) will be detected on operational networks (load, logging, etc.) and whether some implementations are more or less vulnerable due to throttling mechanisms or other implementation specific TCP features.

MD5 Dictionary Attack

- All the information needed to compute RFC2385[15] MD5 authentication is present in the packet except the secret itself:
  - TCP Pseudo-header (sIP, dIP, protocol number, segment length)
  - TCP header (w/o options, and 0 checksum)
  - TCP Segment data (if any)
- “Bgpcrack” test tool uses .pcap files and a dictionary file (with permutation definitions) or can increment through all possible passwords using John the Ripper[16]
- Tool can also run in “online” mode by sending a segment repeatedly with different MD5 passwords—allowing remote brute force (similar to Telnet/HTTP attacks)

MD5 Offline Attack (Sample Run)

```
# ./bgpcrack -r md5.pcap -w words port bgp
39 frames have been processed.
There are 7 TCP segments with MD5 signatures.
Using 784 bytes for storage of MD5 data.
Found a match in frame 8.
Password is 'DOMINO'. Bye.

elapsed time = 8 seconds
```

- A permuted version of the above password "D0M1N0" was found in 3.5 hours with no dictionary file as help: "`./john -stdout:6 -incremental | ~/bgpcrack-2.0/bgpcrack -r ~/md5cap3 -w - -n 1 port bgp -R ~/bgpcrack-2.0/rules.ini`"
- Countermeasures: Choose strong passwords: draft-ietf-idr-md5-keys-00.txt[17]

MD5 Testing

- Test Combinations
  - Valid or invalid peer
  - Established or non-established session
  - Valid or invalid password
  - TCP SYN, PSH-ACK, RST
- Two possible results: drop silently or RST
- Implementations that dropped silently had lower CPU impact than those that RST
- Worst attack using MD5—SYN-Flooding from peer if no session established (70%)
  - Dropped to 30-40% if session already established

MD5 Flooding Results

- Order of processing impacts results
  - Some processed MD5 before sequence number resulting in greater CPU impact when flooded
  - Others processed TCP (checked for valid ports, sequence numbers) resulting in lesser impact
- TCP behavior (especially with regard to existing session) impacts results

BGP Update Flooding

- Wrote python script to establish session and continue to add an arbitrary number of routes at will

```
bash-2.05a$ pyupdate 192.168.1.200 100 eth0

Source IP: 192.168.1.101
Connecting to 192.168.1.200 (45 bytes received)
Sending keepalive...
How many routes to send? 10000
Split into 1000 route updates?y
Generating 10000 routes (40000 bytes)
Building UPDATE...
Source IP: 192.168.1.101
Routes: 1000
NLRI: 4000
BGP Length:  4048
```

BGP Update Flooding Results

- Variations among implementations:
  - Rate at which new routes could be processed
  - CPU Utilization and ICMP latency
  - Behavior when route ceiling was hit
    - Will not accept new routes
    - Tears down BGP session
    - Overwrites old routes

BGP Route Insertion (TCP Hijack)

- Assuming the ability to guess the TCP sequence number; routes can be inserted using a single spoofed update message.
- As soon as the real BGP speaker communicates again (keepalive), an ACK storm ensues due to the overlapping sequence numbers.
- In our testing we found that the ACK storm takes about 5 minutes to resolve during which time the spoofed route will remain in the table and be passed to other routers.

BGP Route Insertion (cont.)

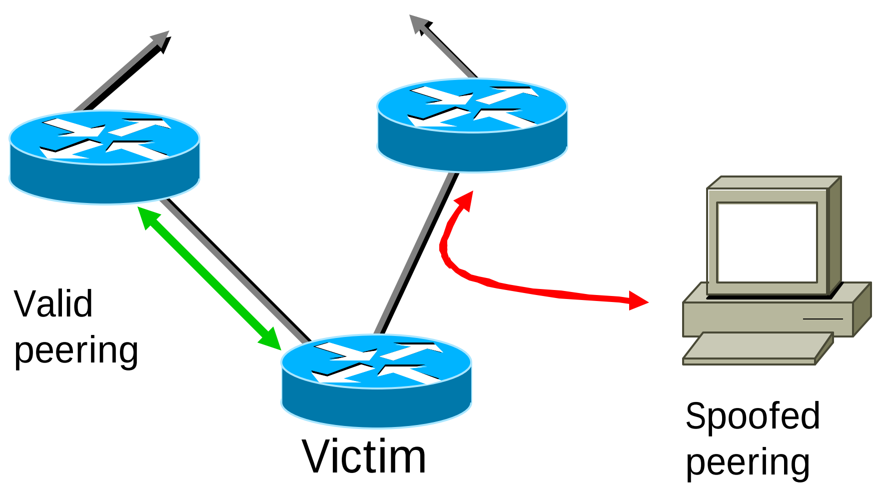
- TCP hijack will insert a binary payload by listening to the sequence numbers on the wire.
- If the attacker stays inline (via ARP or MAC spoofing) the route could stay longer. There may be ways to back-out gracefully without killing the existing session (further research warranted).

```
# ./tcphijack -c 99.0.0.5 -s 99.0.0.3 -p 11041 -P test2.txt
tcphijack: listening on eth0.
pcap expression is 'host 99.0.0.5 and 99.0.0.3 and tcp port
   11041'.
Press Control-C once for status, twice to exit.
We're sync'd to the TCP conversation. Sending Update.
Done.
```

```
5w1d: BGP(0): 99.0.0.5 rcvd UPDATE w/ attr:
nexthop 99.0.0.5, origin i, metric 0, path 5
5w1d: BGP(0): 99.0.0.5 rcvd 7.7.7.0/24
```

BGP Peer Hijack (ARP spoof)

- Using arpspoof an attacker can easily poison the ARP table of a BGP peer and cause the session to be terminated and reestablished with the attacker.
- By spoofing only one peer of the victim both the real BGP speaker and the victim will remain connected. (the victim still peers with other ISPs)



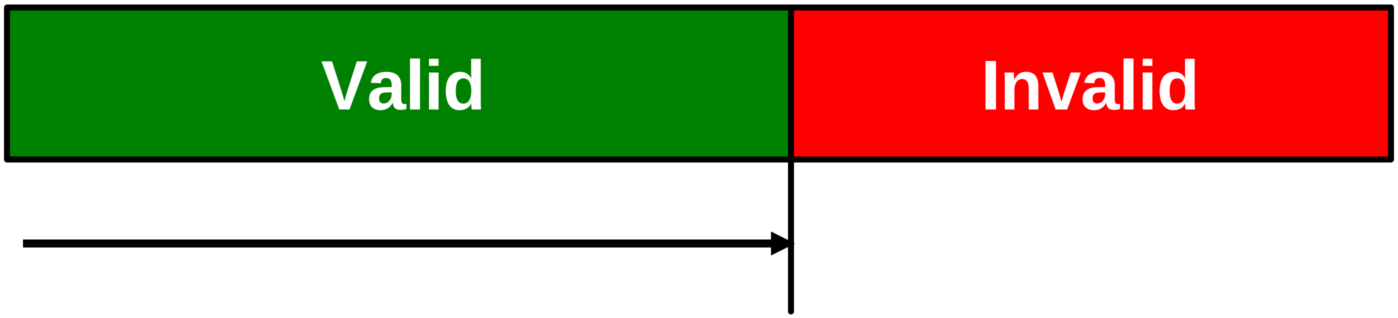
Protocol Fuzzing using PIF

- Provide a general purpose engine to generate malformed fields deeper into packet than existing tools such as ISIC
- Allow a large number of messages for many protocols to be quickly and easily generated without completely describing the protocol
- Focus on complex Type-Length-Value protocols such as BGP and IKE where implementation errors are likely

PIF: Basic Principle of Operation

- The deeper into the message we are able to inject invalid data, the greater confidence we have that the implementation will properly parse malformed input
- This will find improper handling of incorrect length values, truncated messages, and illegal type codes which can cause unstable operation

Message/Packet Depth



PIF Components

- Protocol Description Language (PDL)
  - Describes possible message syntax
  - Consists of a flat-file tree that is chained together
  - Each file is a "block" – discrete protocol unit that consists of multiple fields (line within file)
- User Input Module
  - Parses protocol descriptions and instantiates subset of protocol messages to be generated
  - Result is protocol "template" which is passed to generator
- Message Generation Module
  - Creates final binary output based on template
- Injection Scripts
  - Inject at TCP, UDP, IP, Ethernet layer

Sample Fuzzer run for BGP

```
ciag-530b:~/pif/pdl/bgp# pif bgp build fuzz

====>bgp.pdl<====
marker> fixed field, no input required

[value] [s]hort [l]ong [z]ero [r]andom [v]alid or e[x]it
bgp_len>v
        Using a valid length, calculating at fuzz time.

['0x04', 'keepalive', '0x01', 'open', '0x02', 'update', '0x03',
   'notification']
[c]ycle [value] [p]ermute [r]andom [s]weep [z]ero e[x]it

bgp_type>open

====>bgp-open.pdl<====

ver> fixed field, no input required

                [value] [p]ermute [r]andom [s]weep [z]ero e[x]it
my_as>100
```

From protocol description to identified flaw

Protocol Specs, Sniffer Traces
Source Code, etc.

PDL

PIF Engine

1) **User Input Module**
2) **Msg Gen Module**

```
# ike.pdl
V:64:init_cookie
V:64:resp_cookie
T:16:ike_next_pl:0x01<sa
F:8:version:0x10
```

```
07:34:47.987991 10.89.168.102.179
> 10.89.168.100.33867: S
4067176600:4067176600(0) ack
546274098 win 17376 <mss
1460,nop,nop,sackOK,nop,wscale
0,nop,nop,timestamp 573794391
716420709> (DF)

07:34:47.988123
10.89.168.100.33867 >
10.89.168.102.179: . ack 1 win 5840
<nop,nop,timestamp 716420709
573794391> (DF)

07:34:47.988192
10.89.168.100.33867 >
10.89.168.102.179: P 1:20(19) ack 1
win 5840 <nop,nop,timestamp
716420709 573794391>: BGP
(OPEN[|BGP]) (DF)
```

Testcases

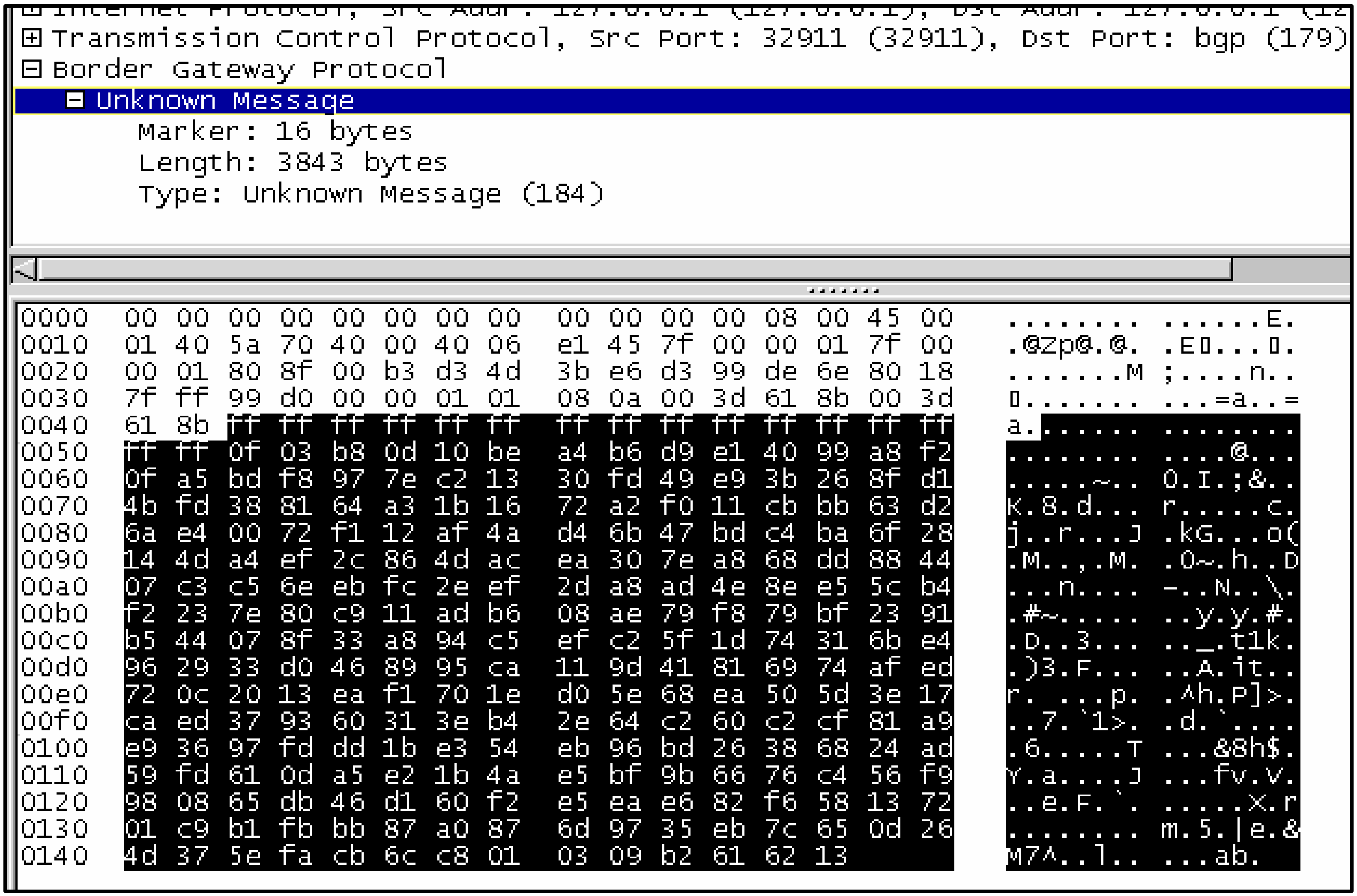
```
eng-
ciag:~/pif/pdl/ike/tc/ike-
test$ ls

0   1   2   3   4
```

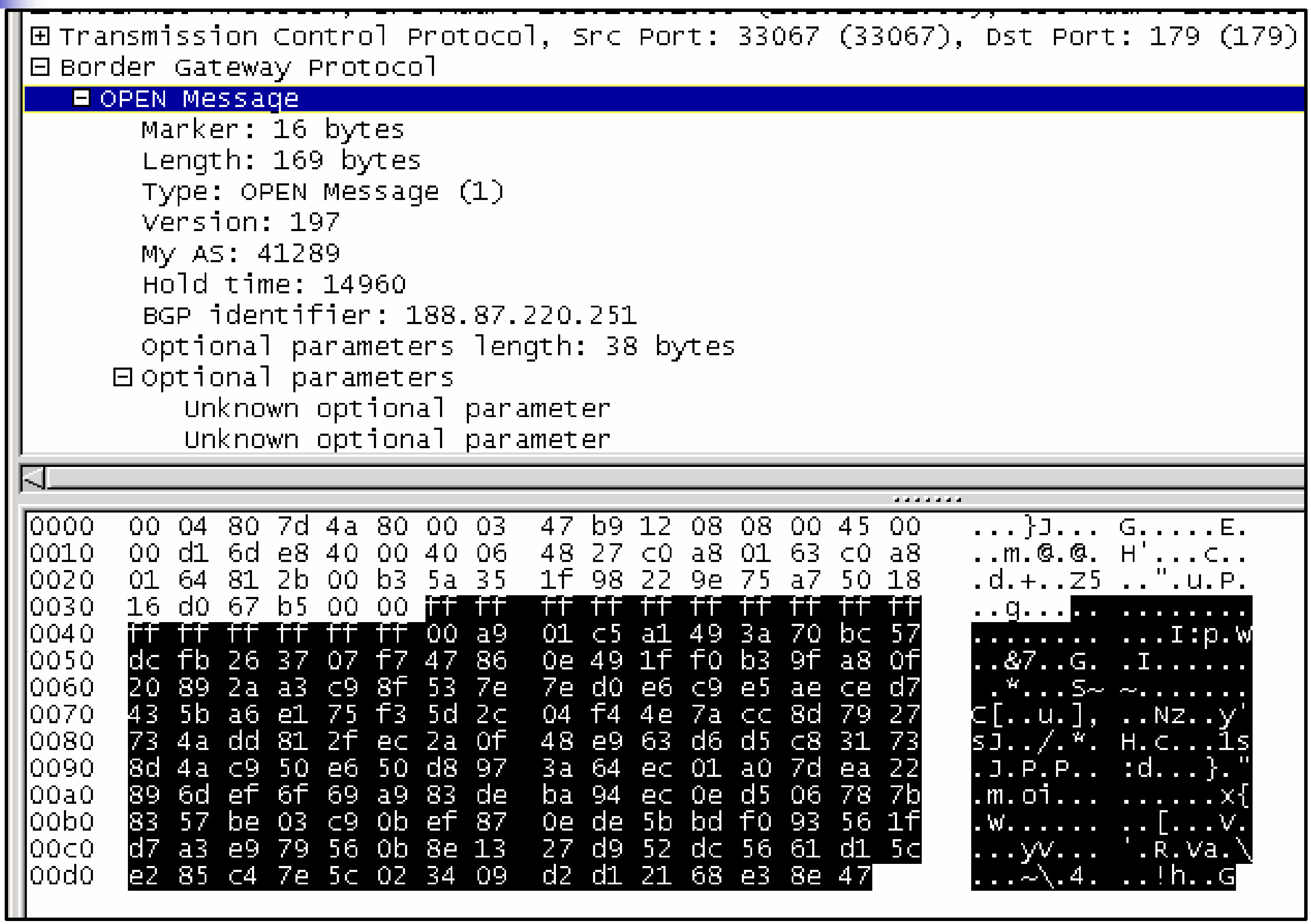
Malformed OPEN Testing

- Generated 100 test cases for each “layer” using pif “backtrace” function
- Messages were from completely invalid to mostly valid:
  - Completely Random
  - Valid Marker + fuzzload
  - Valid Length + fuzzload
  - Valid Version (4) + fuzzload
  - Valid AS + fuzzload
  - *Hold Time + fuzzload*
  - *Identification + fuzzload*
  - Random Option Parameters

Sample Malformed OPEN



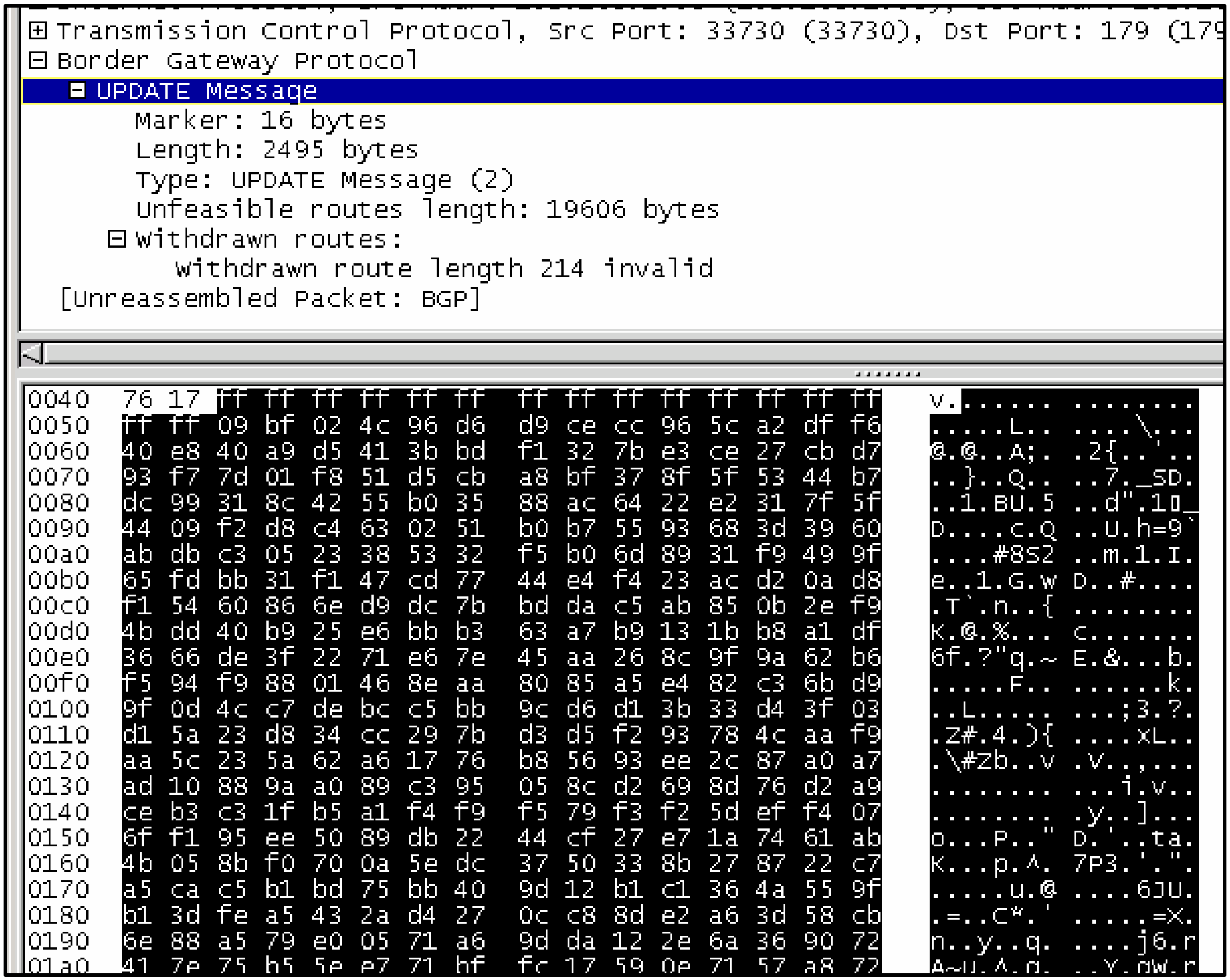
Another Malformed OPEN



Malformed BGP Update Testing

- Generated 100 test cases for each set:
  - Valid BGP type (UPDATE) + fuzzload
  - Valid BGP type (UPDATE with invalid BGP length) + fuzzload
  - Unfeasible length (set to 0) + fuzzload
  - Valid Path Attribute Length + fuzzload
- These test cases provide less comprehensive coverage than OPENs and more testing may be necessary

Sample Malformed BGP Update



BGP Malformed Message Results

- Based on 1200 test cases:
  - Only 4 different flaws were found – impacting 4 of the 7 implementations tested (flaws were unique to each implementation)
  - 3 of the flaws required the attacker to be a valid configured peer and/or valid AS

Areas For Further Testing

- Need more comprehensive set of test cases for UPDATE
- iBGP testing vs. eBGP testing
- Malformed update propagation issues
- Reproduce our tests to confirm results

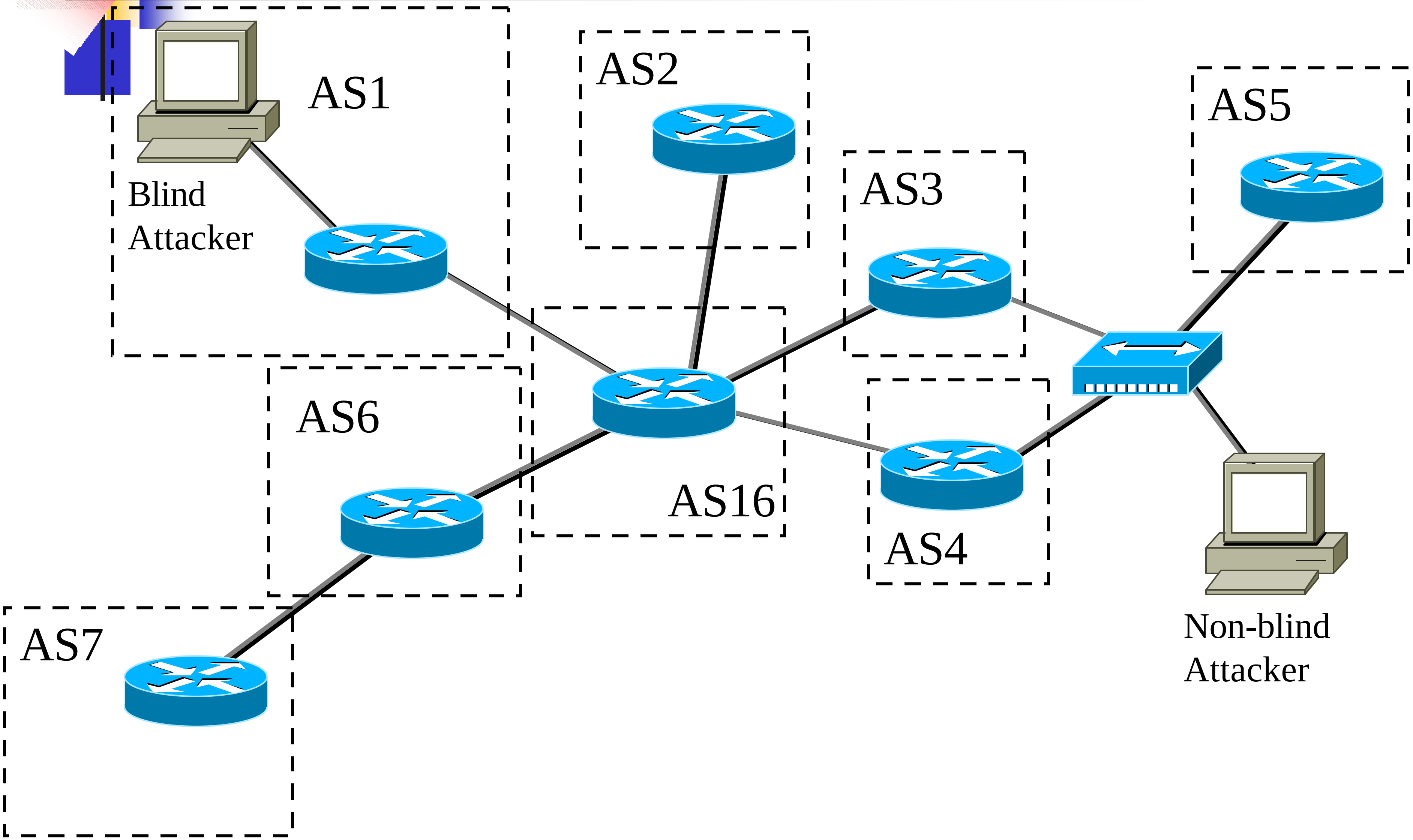
BGP/TCP Implementation Recommendations

- Extensive, configurable logging of connection failures (TCP, BGP, MD5)
- Aggressive rejection of TCP connections from non-configured peers and aggressive timeouts can minimize TCP resource exhaustion attacks
- Aggressive rejection of unauthorized (invalid peer and AS) can minimize the impact of most remote non-blind attacks
- Consider source port randomization
- Lengthy BGP session timeouts (i.e. 60 seconds) can minimize message flooding attacks
- Implement the BGP TTL Hack[18]

Agenda

- Introduction
- BGP Vulnerability Testing
- Analysis of BGP Best Practices
- "Active" ISP Survey Results
- Conclusions

Attack Test Network



Test summary w/No BGP BCPs

- Blind Attacker
  - Systems with TCP reset improvements [24] will be highly resistant to TCP resets, those without can be reset by a determined attacker with adequate bandwidth or by using a distributed attack
  - Most attacker goals depend on getting access to a link with BGP speakers or compromising a router
- Non-Blind Attacker
  - Sessions reset at will
  - Routes inserted (but ACK storm resets the session shortly)
  - Peer hijacking is possible with ARP spoofing
- Compromised Router
  - Tear down sessions, insert invalid routes, modify attributes (could require a rogue implementation), reconfigure to allow malicious peering.

BGP BCPs For Tests

- Based on basic router best practices and Rob Thomas' BGP Hardening Template[19] and ISP Essentials[23] (additions in red)
  - Unicast RPF (RFC 2827 Filtering)
  - Ingress and Egress Prefix Filters (with max prefix length limit and bogon filtering)
  - Route Flap Dampening
  - Bogon route filtering
  - BGP Network ACLs
  - TCP MD5 (with strong passwords)
  - Static ARP for Ethernet peering
  - Static CAM entries and port security [20] for IXP Ethernet switches
  - AS Path Filtering not tested (needs more research)

Key BGP BCPs

- Blind Attacker
  - RFC2827 - even without broad adoption, you can prevent people from spoofing your ranges, and thus all TCP attacks
  - BGP ACLs - Don't let invalid BGP packets on the wire
- Non-Blind Attacker
  - L2 best practices - stops sniffing, hijacking, etc.
  - MD5 - adds additional pain to the attacker
  - Ingress / Egress prefix filtering - limits damage in case of compromise (update flooding, etc.)
- Compromised Router
  - Ingress / Egress prefix filtering - limits extent of damage a compromised router can cause (update flooding, etc.)

BGP BCP Analysis Summary

- As expected, a compromised router is the most beneficial asset to an attacker in a network with BGP BCPs
- TCP MD5 is helpful everywhere, but is particularly useful in shared media environments (deployment issues are being worked on)
- L2 Best practices matter in shared media environments
- Packet filtering to stop spoofed BGP messages at your edge and on each peer will prevent almost all TCP based attacks—and as a result almost all BGP based attacks from non-compromised routers

Agenda

- Introduction
- BGP Vulnerability Testing
- BGP BCP Analysis
- “Active” ISP Survey Results
- Conclusions

Test Methodology

- Goal was to non-intrusively assess basic BCP adoption through probes from an arbitrary IP address
  - Limit scanning to prevent production impact—a single SYN with no retries
- Build table of potential BGP speakers by running traceroutes to approx. 120,000 hosts (one for each CIDR block in the Internet's route table)
- Probes:
  - Send 1 x TCP SYNs to ports 22, 23, 80, 179
  - Embed message in payload identifying probes as non-malicious
  - Measure response (SYN ACK, RST, No Response)
- Send BGP OPEN to those that SYN-ACK on port 179
  - Sessions used an unused AS #
  - Record BGP message that is returned

“Active” ISP Survey Results (Summary)

- Total non-1918 routers probed: 115,466
- BGP Speakers
  - SYN-ACK - 4,602
  - RST - 3,088
  - No Response - 107,777
- BGP Open Test Results
  - OPEN / NOTIFICATION - 1,666
    - AUTH FAIL - 1635
    - CEASE - 11
    - BAD AS - 20
  - NOTIFICATION ONLY - 84
    - AUTH FAIL - 1
    - CEASE - 83
  - RST - 264
  - Connect (No Data) - 2,147

- SSH daemons: 6,349
- Telnet daemons: 10,907
- HTTP Servers: 5,565
- 16,815 routers were reachable* on at least one admin interface (14.5% of probed routers)

*Based only on receipt of SYN-ACK, so daemons that you can actually connect() to could be lower!

Admin Port Reachability (by Country)

Several countries had either 100% of their routers accessible or 0% but were not counted since there were less than 10 routers probed in each of these countries.

Honorable Mentions:

Spain - 878 (5.13%)
France - 1820 (6.48%)
Great Britain - 4005 (7.72%)

Country	Total Probed Routers	Percentage Admin Reachable
Maldives	10	0%
Gibraltar	16	0%
Iceland	34	2.94%
Kazakstan	80	3.75%
Fiji	23	4.35%
USA	56481	14.22%
Average	--	14.5%
Canada	4555	15.32%
Kyrgyzstan	19	52.63%
French Polynesia	12	58.33%
Tanzania	10	60%
Uzbekistan	25	68%
Bahamas	15	73%

Conclusions

- The most damaging attacks are caused by the deliberate misconfiguration of a trusted router
  - Compromising the router is not BGP specific and is not covered here. Best practices should be well understood for router hardening[5]
- Resistance to TCP attacks largely depends on vendor implementations and operator best practices
  - Blind hijacking is impossible with RFC 2827 filtering
  - TCP Enhancements [24] make even a system without BCPs highly resistant to attacks
  - Even “easy attacks” (TCP Resource Exhaustion) against port 179 are non-trivial against tight implementations and have minimal impact compared to other DoS attacks
- Why bother with lower layer attacks (ARP, TCP) against BGP when you can own the box?

More Conclusions

- Encourage your vendors to to test their BGP implementations and do your own security testing
  - These tests should be repeatable using this document and the BGP Attack Tree
- Implement BGP BCPs, especially admin ports!
- Liberally use clue-stick next time someone says “BGP is totally insecure!”
  - Security isn’t an all or nothing proposition
  - soBGP[21] and S-BGP improve security, but…
    - New implementations, new bugs
    - Needs to go through the IETF process

What next?

- Generate more test-cases (more on BGP update and other message types)
- Test more platforms!
  - Need vendors, users, and independent researchers to repeat and extend tests we've outlined here
  - Based on "Active ISP Survey" there are more BGP implementations that need to be tested

References

[1] S-BGP - http://www.net-tech.bbn.com/sbgp/sbgp-index.html

[2] BGP Security Vulnerabilities Analysis, Murphy, 2003- http://www.ietf.org/internet-drafts/draft-murphy-bgp-vuln-02.txt

[3] BGP Security Protections, Murphy, 2002 - http://www.ietf.org/internet-drafts/draft-murphy-bgp-protect-01.txt

[4] Expert: Router holes threaten Net - http://zdnet.com.com/2100-1105-990608.html

[5] Cisco Router Security Recommendation Guides - http://www.nsa.gov/snac/cisco/guides/cis-2.pdf

[6] An Attack Tree for the Border Gateway Protocol, Convery et. al., 2003- http://trinux.sourceforge.net/draft-convery-bgpattack-00.txt

References (cont.)

[7] Hping - http://www.hping.org

[8] Nemesis - http://www.packetfactory.net/projects/nemesis/

[9] Netcat - http://www.atstake.com/research/tools/network_utilities/

[10] Naptha - http://packetstormsecurity.nl/0101-exploits/

[11] Dsniff - http://www.monkey.org/~dugsong/dsniff/

[12] The Problem with Random Increments, Newsham, 2001 - http://comsec.theclerk.com/CISSP/The%20Problem%20with%20Random%20Increments_4_30_01_final.PDF

[13] Strange Attractors and TCP/IP Sequence Number Analysis, Zalewski, 2001 - http://razor.bindview.com/publish/papers/tcpseq.html

References (cont.)

[14] Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing, RFC 2827, BCP 38, Ferguson and Senie, 2000 - http://www.ietf.org/rfc/rfc2827.txt

[15] Protection of BGP Sessions via the TCP MD5 Signature Option, RFC2385, Heffernan, 1998 - http://www.ietf.org/rfc/rfc2385.txt

[16] John the Ripper - http://www.openwall.com/john/

[17] Security Requirements for Keys used with the TCP MD5 Signature Option, Leech, 2002 - http://www.ietf.org/internet-drafts/draft-ietf-idr-md5-keys-00.txt

[18] The BGP TTL Security Hack (BTSH), Gill et. al., 2002 - http://www.ietf.org/internet-drafts/draft-gill-btsh-01.txt

[19] Secure BGP Template, Thomas, v.2.9 - http://www.cymru.com/Documents/secure-bgp-template.html

References (cont.)

[20] Hacking Layer 2: Fun with Ethernet Switches, Convery, 2002 - http://www.blackhat.com/presentations/bh-usa-02/bh-us-02-convery-switches.pdf

[21] Extensions to BGP to Support Secure Origin BGP (soBGP), Ng, 2002 - http://www.ietf.org/internet-drafts/draft-ng-sobgp-bgp-extensions-00.txt

[22] The Naptha DoS Vulnerabilities, Keyes. http://razor.bindview.com/publish/advisories/adv_NAPTHA.html

[23] Cisco ISP Essentials, Greene - ftp://ftp-eng/cons/isp/essentials

[24] Slipping in the Window: TCP Reset Attacks, Watson, 2004 – http://www.terrorist.net

[25] Transmission Control Protocol Security Considerations, Stewart, 2004, http://www.ietf.org/internet-drafts/draft-ietf-tcpm-tcpsecure-00.txt

Questions?