

# Black Ops of TCP/IP 2005

---

Dan Kaminsky

DoxPara Research

<http://www.doxpara.com>

# Introduction (Who am I?)

---

- Fifth year speaking at Black Hat
  - Subjects: SSH, TCP/IP, DNS
  - Code: Paketto Keiretsu, OzymanDNS
- Several books
  - Hack Proofing your Network
  - Stealing The Network: How To Own The Box
  - Aggressive Network Self-Defense
- Formerly of Cisco and Avaya

# What Are We Here To Do Today?

---

- MD5
- IP Fragmentation
- Firewall / IPS Fingerprinting
- DNS Poisoning (and other tricks)
- Scanning The Internet
- Visualizing That Scan
- Watch TV

# A Tale Of Two Pages: [www.doxpara.com/t1.html](http://www.doxpara.com/t1.html) and t2.html



Lockheed Martin - We never forget who we're working for. - Mozilla Firefox

File Edit View Go Bookmarks Tools Help

http://www.doxpara.com/t1.html

Home | Contact Us

LOCKHEED MARTIN  
We never forget who we're working for™

Search:  GO

Advanced Search

## PRESIDENTIAL HELICOPTER

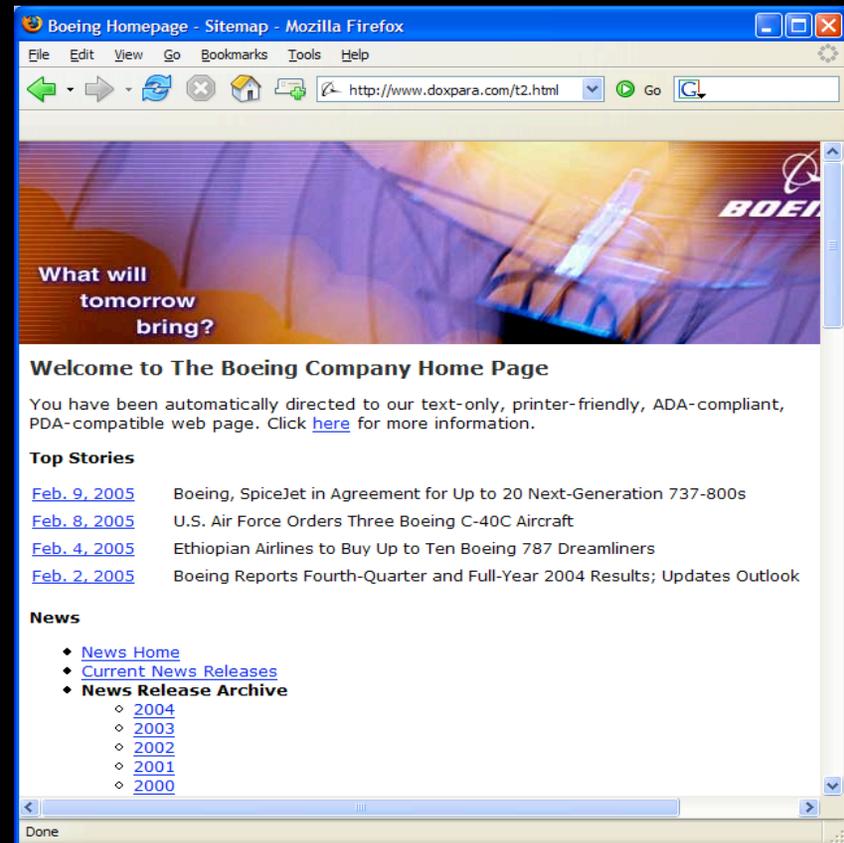


Defining Moments

© 2005 Lockheed Martin Corporation  
Disclaimer

Stock Price: [ 59.69 ]

Done



Boeing Homepage - Sitemap - Mozilla Firefox

File Edit View Go Bookmarks Tools Help

http://www.doxpara.com/t2.html

## What will tomorrow bring?

### Welcome to The Boeing Company Home Page

You have been automatically directed to our text-only, printer-friendly, ADA-compliant, PDA-compatible web page. Click [here](#) for more information.

#### Top Stories

- [Feb. 9, 2005](#) Boeing, SpiceJet in Agreement for Up to 20 Next-Generation 737-800s
- [Feb. 8, 2005](#) U.S. Air Force Orders Three Boeing C-40C Aircraft
- [Feb. 4, 2005](#) Ethiopian Airlines to Buy Up to Ten Boeing 787 Dreamliners
- [Feb. 2, 2005](#) Boeing Reports Fourth-Quarter and Full-Year 2004 Results; Updates Outlook

#### News

- ◆ [News Home](#)
- ◆ [Current News Releases](#)
- ◆ [News Release Archive](#)
  - ◇ [2004](#)
  - ◇ [2003](#)
  - ◇ [2002](#)
  - ◇ [2001](#)
  - ◇ [2000](#)

Done

# They Look Different...But Are They?

---

- `$ curl -s http://www.doxpara.com/t1.html | md5sum.exe`  
`c0f3adb824590b40944614268e627421 *-`
- `$ curl -s http://www.doxpara.com/t2.html | md5sum.exe`  
`c0f3adb824590b40944614268e627421 *-`
- **MD5 Sees the two web pages as possessing identical content!**
  - SHA-1 not fooled
  - `$ curl -s http://www.doxpara.com/t1.html | sha1sum.exe`  
`9a2b6e9de9c2343a26084ab64e6d902aab6e2b1d *-`
  - `$ curl -s http://www.doxpara.com/t2.html | sha1sum.exe`  
`d2da4f8bfef1d06ca1a821b99bd614fa45116790 *-`
- What is happening here?

# How We Got Here

---

- 1) We have an unsafe hash
  - Definition of a safe hash: “Computationally infeasible to find two files with the same hash”
  - Dr. Xiaoyun Wang made two files with the same hash.
- 2) Hashes degrade very poorly under collision conditions
  - If two things collide (like the Wang hashes), then anything can be added to both hashes and collision will be maintained
  - If  $\text{md5}(x) == \text{md5}(y)$ ,  $\text{md5}(x+q) == \text{md5}(y+q)$  for all values  $q$
  - This is because of the iterative design of cryptographic hashes – the information about past differences is lost.
- 3) The Web is very flexible
  - You can code to it (Javascript)
  - It accepts garbage (Javascript...and broken HTML)

# What It Looks Like

---

- *Start with the either vec1 or vec2, the two files from Wang...*

```
Ñ1†ÄæîÄi=_ü VÊµ†E~ «@X>,û %o U_4□...
```

- *Continue with javascript encoded arrays of both files...*

```
<script language=javascript type="text/javascript">
```

```
boeing_enc="\
```

```
%3C%21DOCTYPE%20html%20PUBLIC%20%22%2D%2F..."
```

- *Finish with code that decodes the arrays and chooses which to display based on the contents at the beginning of the file.*

```
alldata = document.getElementsByTagName("HTML")[0].innerHTML;
```

```
isVec1 = data.indexOf("%C2%B5%07%12F");
```

```
if(isVec1<0) isVec1=0; if(isVec1){
```

- ```
document.getElementsByTagName("BODY")[0].innerHTML="";
document.write(vec1message); } if(!isVec1){
document.getElementsByTagName("BODY")[0].innerHTML="";
document.write(vec2message); }
```

# How You Can Do It

---

- Tool Release: “Confoo”
  - `$ perl confoo.pl`  
confoo 1.0: Web Conflation Attack Using Colliding MD5 Vectors and Javascript  
Author: Dan Kaminsky(dan@doxpara.com)  
Example: `./confoo www.lockheedmartin.com`  
`active.boeing.com/sitemap.cfm`
  - Outputs t1.html and t2.html, as on the site
- For more information, see research paper, “MD5 to be considered harmful someday”
  - Stop using MD5 😊

# Introducing IP Fragmentation

---

- "Fragmentation...an interesting early architectural error that shows how much experimentation was going on while IP was being designed." -- Paul Vixie
- Fragmentation: If a packet is too large for the underlying link layer, it may be split by any router (unless behavior is explicitly disabled) into multiple fragments
- Why a problem? IP is supposed to be "stateless"
  - Fire a packet and forget about it
  - Receive a packet and be done with it
  - Fragmentation keeps the former but destroys reception
  - Systems need to keep fragments around, wait for future fragments, reassemble...what if fragments overlap?

# IP Fragmentation: Some History

---

- Major mechanism for evading IDS
  - “Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection.” – Newsham and Ptacek, 1998
  - Fragrouter, Dug Song, 1999

# Remaining Adventures in Reassembly: Adventures In Temporality

---

- IP has been mostly “picked clean”...is there anything left?
- Timing Attacks
  - Successful against cryptosystems all the time
  - Are there any timers in IP?
- The IP Fragment Reassembly Timer
  - Maximum amount of time a fragment will be held, unassembled, before it “expires” and is flushed
  - Differs from OS to OS – yes, it’s a fingerprint
    - Ofir Arkin noted IP fragment scanning, but not fingerprinting
  - Can we evade with this?

# It's Skew

---

- What if the IDS has a different concept of expiration time than the host?
  - If IDS expires first: Just send fragments too slow for the IDS but fast enough for the target
    - **This definitely happens**
  - But what if host expires first?
    - Linux/FreeBSD timer: 30s
    - Snort frag2 timer: 60s
  - Is it possible to still evade an IDS when its timer lasts longer than that of your target's?

# Protocol Inversion

---

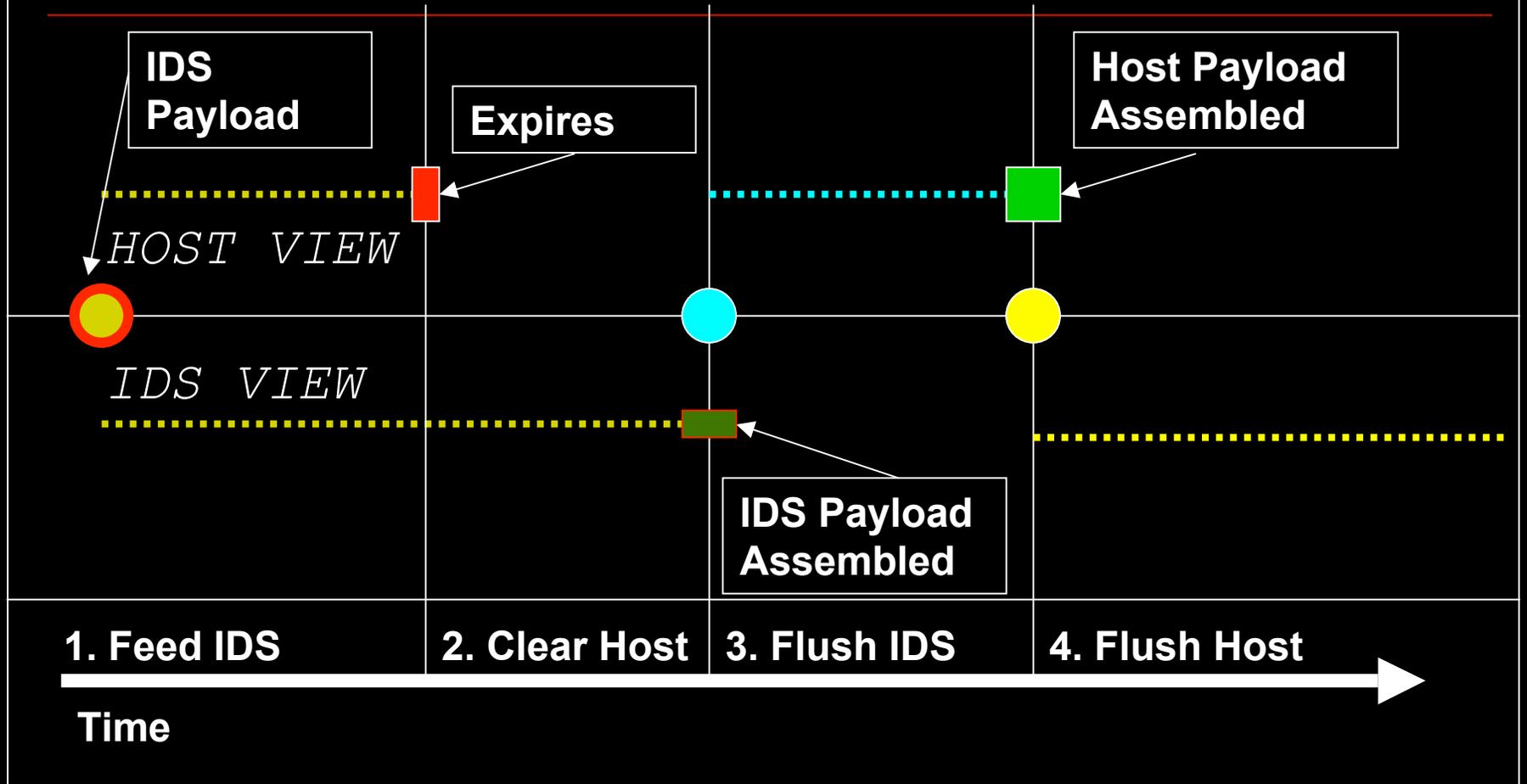
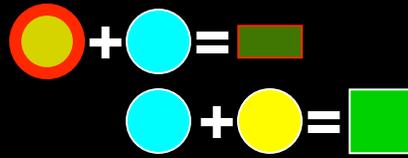
- Problem: IDS keeps fragments for too long
- Solution: Make IDS drop fragments
- Strategy: Fragments leave the reassembly queue when either they aren't reassembled...or when they are.
  - Is it possible to give the IDS something to reassemble against – without causing the target host to undergo a similar reassembly?
  - Of course – use a timing attack!

# The Temporal IP Attack

---

- Prepare:
  - Nice request, malicious request, and a shared header between the two
    - *Header:* HTTP 1/1 GET
      - *IDS Payload:* index.html
      - *Host Payload:*  
msadc/..%255c../..%255c../..%255c../winnt/system32/cmd.exe  
?/c+dir+c:%5c
- 1) Send IDS payload
- 2) Wait. Host will drop. IDS won't.
- 3) Send shared header. IDS sees the two fragments it needs to reassemble a packet – and gets a legitimate request. Host dropped the IDS payload, so it just stores the header.
- 4) Send host payload. Host sees the two fragments it needs to reassemble a packet – and gets attacked. IDS dropped the shared header, so it just stores the host payload (and never reassembles it).

# Art



# What about Checksums?

---

- A problem – we can certainly find a common header between two payloads, but won't the checksums be off?
- A solution – fix the checksums later
  - Strategy from Jeremy Bentham's TCP/IP Lean
    - AKA “How to use the Internet without enough RAM to store a single packet” and “How to debug Ethernet with an O-Scope”
  - Put a fixed checksum in your header
  - Add an offset in your payload to make the data agree with the header checksum
  - Works because there are ignored fields in payloads

# Polymorphic Exploits

---

- We can backport this polymorphic attack to all the original mechanisms used by Ptacek/Newsham/Song
  - Send a single series of packets that, based on the platform they arrive at, reassemble into the correct attack for that platform
    - Half credit for this goes to Jason Larsen, who thought of this with me last year

# Hitting the Brakes

---

- Right about now, several IDS vendors – *and especially IPS vendors* – are noticing flaws
  - In order to implement this attack, overlapping fragments must be transmitted
  - Some systems cache used IP ID's even after they've already reassembled data
  - IPS's can use this overlap to block entire sessions
    - An IPS is an IDS that can censor the incoming packet stream
- They're right. Against certain architectures, the temporal attack doesn't work as described

# Recovering the attack?

---

- All devices have a limited capacity for storing state data
  - Like, for example, which IPID's have already been used
  - We could flood the device with fragments, both with identical source/dest IPs and different, so as to exhaust this cache
    - Though this would alarm as well, in the IPS case it would overrun the censor
  - There is actually potential for combining this attack with the temporal attack, as some platforms will refuse to accept new fragments until  $n$  old fragments expire
    - And only we know when they'll expire ☺
- Overall, certain IPS architectures – even if they weren't aware of timing attacks in their design phase – are likely to still defend against these attacks
  - Especially once they notice hosts unexpectedly acknowledging

# Changing Course

---

- Some IPS's will block this. What now?
  - What are IPS's?
    - Firewalls w/ dynamic rulesets / censoring IDS
    - These dynamic rulesets can trigger on increasingly obscure faults across the entire communication stack
    - What they'll trigger against differs from product to product, version to version
  - Security products in general are under increased scrutiny
    - Combine complex state machines with a need for maximum efficiency
    - Over 20 advisories regarding vulnerabilities in security products
  - *Blocking sends information*
    - Is it possible to use this leaked information to **fingerprint security architectures?**

# Hopcount Desync (SLIDE FROM 2003 – FW fingerprinting is not new)

```
■ root@arachnadox:~# scanrand -blk -e
  local.doxpara.com:80,21,443,465,139,8000,31337
■   UP:      64.81.64.164:80      [11]    0.477s
■  DOWN:    64.81.64.164:21      [12]    0.478s
■   UP:      64.81.64.164:443    [11]    0.478s
■  DOWN:    64.81.64.164:465    [12]    0.478s
■  DOWN:    64.81.64.164:139    [22]    0.488s
```

What's going on:

The host is genuinely 11 or 12 hops away. All of the up ports reflect that, but only a few of the downed ports. The rest are showing double the remote distance. This is due to the a PIX firewall interspersed between myself and the target. It's (too) quickly reflecting the SYN I sent to it right back to me as a RST|ACK, without resetting values like the TTL. Thus, the same source value decrements twice across the network –  $22 = 11 * 2$  – and we can detect the filter.

# Firewall/IPS Fingerprinting: Other products

---

- Tipping Point: Does not allow out-of-order TCP segments – everything must arrive on the edge of a window
- Checkpoint: Does not allow (by default) DNS packets that declare EDNS0 (DNSSec!) support
- L3/L4 Mechanisms
  - Invalid Checksums (at IP, TCP, UDP, ICMP)
  - Invalid Options (at IP and TCP, and actually UDP too)
  - Out of order fragments/segments (at IP and TCP)
  - Invalid ICMP type, code
- Application Layer Mechanisms
  - Invalid HTTP request types, or TRACE/WebDAV
  - SQL Injection in TCP payloads (WITHOUT the necessary line terminator)
  - Invalid DNS
- Using Schiffman's "Firewalk" methodology, each query leaks the location of the blockage – and I can always walk to the host before the FW

# IPv6 Reassembly A Coming Fingerprint

---

- What encapsulations will a given IDS/IPS support?
  - There are so many variations
  - *They chain* – IPv6 in IPv4 in IPv6 in IPv4, etc.
  - Nowhere near all could possibly be parsed by every client
    - Thus many different possible signatures – blocks 4in6 exploits, blocks 6in4in6 exploits, blocks Torpedo exploits, etc.

# A Problem for IDS/IPS people

---

- There are an astonishing number of ways to bridge IPv4 and IPv6.
  - Here's another: Name servers hosted on both IPv4 and IPv6 can resolve names against either protocol, using addresses delivered via either protocol.
- These ways all chain – Teredo in IPv4 in IPv6 in IPv4 over DNS, etc.
- Not all chains can (or should) work for every client
  - How can an IDS/IPS have any hope of predicting what its clients will perceive?

# Three approaches to IPv6 Encapsulation Management

---

- 1) Enforce only a few encapsulations
  - So you drop traffic from a few hosts
    - This strategy makes the Internet fall apart
- 2) Scrub (unpack and repack) all encapsulations down to one mode you make decisions on
  - I very much like packet scrubbing, but there's not been a scalable scrubber deployed yet
- 3) Ask.
  - Upon seeing a new encapsulation style, synthesize a new, safe packet – an ICMP Ping, in particular – and submit it to a target host with the same encapsulation pattern
    - Will return both *whether* a packet can be encapsulated like that *and* the precise policy used to resolve fragmentation conflicts

# However, IPS's should not do this.

---

- “After sufficient amounts of invalid traffic, we just ban you from our network. Fingerprint THIS!”
  - I've heard this a lot lately. Some of you know why.
  - *Many* automatic shunning systems deployed
  - *Not a good idea.*
    - To understand why automatic shunning is bad – just dig.

# It Might Be Bad To Shun These Guys.

---

- ; <<>> DiG 9.3.0rc2 <<>>
- . 511355 IN NS F.ROOT-SERVERS.NET.
- . 511355 IN NS G.ROOT-SERVERS.NET.
- . 511355 IN NS H.ROOT-SERVERS.NET.
- . 511355 IN NS I.ROOT-SERVERS.NET.
  
- ;; ADDITIONAL SECTION:
- A.ROOT-SERVERS.NET. 172766 IN A 198.41.0.4
- B.ROOT-SERVERS.NET. 604777 IN A 192.228.79.201
- C.ROOT-SERVERS.NET. 604782 IN A 192.33.4.12
- D.ROOT-SERVERS.NET. 604786 IN A 128.8.10.90
- E.ROOT-SERVERS.NET. 604791 IN A 192.203.230.10
- F.ROOT-SERVERS.NET. 604797 IN A 192.5.5.241
- J.ROOT-SERVERS.NET. 172766 IN A 192.58.128.30

# Something More Elegant

---

- Spoofing malicious traffic from the root servers – ugly, yes, kills a net connection, sure, but:
  - Too large scale
  - Been whispered about for years
- But there are other name servers...
  - I've been investigating DNS poisoning
  - Is it possible, given networks that implement automatic network shunning, to poison name server caches and thus selectively hijack network traffic?

# The Name Game

---

- The general theme: Block communication between two name servers
  - Bad: Targeted Denial of Service – Customers from a particular network are unable to contact a particular bank/merchant/email provider
  - Worse: Targeted DNS Poisoning – Being unable to communicate, a window is left open for an extended period of time for a flood of fake replies to eventually hit on the correct answer
- Can either block server at client net, or client at server net

# Double Sided

---

- Spoof malicious traffic from the client network to the server network
  - Client will have outstanding requests to the server – if they're using a fixed DNS port\*, only 32K requests on average to find their TXID's
  - How do we make them look up a given network on demand?
    - Recursion – Just ask them to look up [www.merchant.com](http://www.merchant.com)
    - PTR NS Forwarding – Claim that, to look up your IP, it's necessary to ask the nameserver at [www.merchant.com](http://www.merchant.com). Then use your IP to go to their web server

# Double Density

---

- Spoof malicious traffic from the server network to the client network
  - Client can make requests, but server responses are blocked
  - But wait? Aren't *our own* forged responses blocked too?
    - Funny thing about DNS...about 15% of servers reply from a different IP address than you talked to in the first place!
    - With a lack of interface affinity in servers, comes an ignorance of incoming IP address on clients
      - This is BTW why UDP NAT2NAT works
    - So while the *legitimate* server responds in vain, *our attacks can come in from anywhere*
- **Moral of the story: Automated network shunning is a very bad idea. Do not give the world access to your firewall tables.**

# But I LIKE Autoshun

---

- Is it possible to mitigate the worst aspects of automatic network blocking?
  - Make sure you can still send mail to autoblocked networks (and actually do)
    - Implies – make sure you can still do DNS lookups against the network, and get the replies
  - If possible, make the block stateful – outbound connections from your network should override
    - Even “outbound sessions override and ‘hold down’ autoshuns” is a significant improvement
  - Be very careful about blocking access to any service which otherwise may be phished / impersonated.
    - Remember, your own name server is a dependency

# But...but...

---

- What about complaint emails?
  - Funny thing happens when you block nameservers...you lose the ability to retrieve MX records, so you stop being able to send complaint mail
    - I'm sure at least some autoshunners have taken this into account :)
- Now what would I know about complaints?

# Poppa's Got A New Pair Of Shoes

---

- Prolexic – who I worked with on the Opte internet mapping project – has given me a very high bandwidth connection to work with
  - They're a third-party spam filter for IP – your data is BGP'd to them, they forward you a filtered stream.
  - I actually can't generate packets faster than this network can route ☹
- Been actively probing the Internet DNS Infrastructure
  - Partnering with Mike Schiffman of Cisco Critical Infrastructure Assurance Group and Sebastian Kraemer at the University of Potsdam (and maybe you – send me a proposal?)
  - Extremely large scale scans – every IP, every name server, everywhere

# Always Bet On Black

---

- 100% legitimate packets – this isn't a global pen test, this is an investigation in to the largest cooperative caching architecture on the Internet – one that is getting poisoned again
- Asking: How is this architecture laid out? How prevalent is DNSSEC support? Where do we need to invest resources in protection? And what is going on with DNS poisoning?
  - We can't manage what we can't measure. This is an attempt to measure.
- Not the first to do a large scale network scan

# DON'T TRY THIS AT HOME

---

- “Where’d my colo go?” 😊
  - You **will** get complaints
  - You **will** get calls from scary sounding places
  - As well you should. This is behavior that normally precedes an attack.
    - So why am I doing it? **Because the attackers should not have better intel than we do.**

# Open And Honest

---

## ■ Reverse DNS

- deluvian root # nslookup 209.200.133.226  
Non-authoritative answer:  
226.133.200.209.in-addr.arpa name = infrastructure-audit-1.see-port-80.doxpara.com.

## ■ Web info

- Technical details
- Explanation of motivation
- Links to papers, news articles
- My phone #

# ARIN Updated

---

- NetRange: 209.200.133.224 - 209.200.133.255 CIDR: 209.200.133.224/27 NetName: DANKAMINSKY-SECURITY-RESEARCH NetHandle: NET-209-200-133-224-1 Parent: NET-209-200-128-0-1 NetType: Reassigned Comment: This is a security research project, please send all Comment: abuse and alert requests to dan@doxpara.com. RegDate: 2005-07-08 Updated: 2005-07-08

# And even with...

---

- Still, large scale analysis does not go unnoticed, uninvestigated, and uncomplained about
  - After further explanation, almost all administrators have been courteous
    - “Thank you for the information. See you in Vegas.”

# Some Early Results

---

- Priority 1: Google was taken out by an exploit that hit MSDNS systems forwarding to BIND4/8. Find all of these.
- To begin with – need to identify all name servers on the Internet
  - Requirement: Legitimate lookup that worked on every normal name server, but would not be of a type to require recursion
    - Disabling the recursion desired bit doesn't always work, apparently
    - Lookup: 1.0.0.127.in-addr.arpa PTR
    - Expected reply: localhost.
    - Actual replies: Rather more complicated.
  - Could also have sent traffic on TCP/53 but not all servers accept
- Now can set about finding which ones are related to which other ones

# Interrelationship Mapping[0]

---

- Slow: “Ask Bob to look up the stock price for an obscure stock. If you ask Sally, and she already knows, she talked to Bob”
  - Recursively request that a server acquire – and send you – a given name. Then, non-recursively ask everyone else if they’ve heard of that name. If they have – they share a cache with the first server.

# Interrelationship Mapping[1]

---

- Faster: “Ask everyone to look up the latest stock price. If someone comes back with the stock price as it was 13 minutes ago, they talked to the guy you asked 13 minutes ago.”
  - Recursively request the same information of everyone. You will either:
    - A) Get back the data – with a full TTL
    - B) Get back the data with the TTL decremented by some degree of seconds.
    - DNS records come with an expiration date
  - If the returned TTL = original minus 83 seconds, then this node is connected to whoever you were scanning 83 seconds ago.
  - If you were scanning more than one host at a time – repeat your scan in a different order, and the next time you’ll have a different value
  - A bit buggy – some hosts cache records, but do not decrement

# Interrelationship Mapping[2]

---

- Fastest: “Ask Bob to research something in your library. If John shows up to do the research – you know Bob asks John to do such things.”
  - 1. Create a wildcard domain
    - \*.maddns.net
  - 2. Insert a cookie into the name you would scan for, describing the address you are talking to
    - 1-2-3-4.maddns.net
  - When queries arrive, looking for a record that match 1-2-3-4.maddns.net, compare the name in the DNS query with the IP address the request is coming from. Interrelationship established!
  - `select cookieip, ipsrc from recursivequery group by cookieip, ipsrc;`
    - SQL emits a list of interrelated hosts

# What was found?

---

- 2.5M *verified* name servers
  - Up to nine million possible, but 2.5M have been / remain responsive
  - All 2.5M have been run through Roy Arend's FPDNS
    - NOTE: FPDNS gives more data than CH TXT (explicit version requesting), and...er...doesn't set off nearly as many alarms.
- At least 230K forwarding to Bind8, as specifically forbidden as per ISC BIND documentation – almost 10% of the sampled DNS!
- At least 13K Windows name servers still forwarding to Bind8!
  - At least 53K "OTHER"
  - BIND8->BIND8 forwardings must be further analyzed, to determine multihomed vs. a true forwarding relationship
    - This can be found by – can data enter one cache, without entering the other? If so, one is higher in a hierarchy than another
    - Is BIND9->BIND8 forwarding problematic? 18.7K instances.

# I Wonder...

---

- Normal exploit methodology: “What is this thing vulnerable to?”
- Reverse exploit methodology: “Is anyone vulnerable to this?”
  - Now, again, I can’t pen-test – so 100% legitimate packet requirement must be made
  - But...is anyone doing something wrong with the 100% legit data I’m sending them?

# Elegant Problem...

---

## ■ Potential Fault In Recursion

- In recursion, clients ask their local server a question, and their local server goes out and asks that same question elsewhere.
- If someone were to...say...just copy the incoming request, and forward it elsewhere, the DNS transaction ID would stay the same, and the client, having set this TXID, could spoof the response and thus pollute the cache for anyone else who tried to use that server.
- No *known* systems do this...but does anyone?

# ...Brute Force Solution

---

- 1. Send recursive queries out to servers w/ fixed (or calculatable transaction id)
  - The question name for the queries? Ourselves, basically
- 2. When servers come back to service those queries, check their transaction ID
  - Did they use ours? 1/65K chance of coincidence
- What happened?
  - ~110 hosts replied
  - ADSL modem from major vendor, and...uh...
    - An old version of the name server I was using at the time
  - TODO: Static TXID, *vaguely* predictable TXID/Source Port

# \*Speaking of Source Ports

---

- Something very interesting was discovered during this research
- UDP ports are not asymmetrical like TCP ports – there's simply open and shut, not “client and server”.
  - **This means you can scan for UDP client ports, such as used by name servers!**
    - “But name servers are *supposed* to deviate their source ports randomly!”
    - Lets check the data.

# Just The Facts

---

- `echo "select sport,count(sport) as num from forward_query group by sport order by num;" | mysql dns | tail -n 10`

```
32770      54617
1036       55059
50098      64200
5353       68854
50477      77099
1024       176922
32769      195008
1027       234082
53         462345
32768      823579
```

- It's good to have real data. Note that:
  - One can scan for default ports
  - The presence of 32769 means we can actually measure the usage level of many servers, as they assign their ports one by one

# Anything else?

---

- Probable evidence of DNS poisoning I cannot talk about yet.
- Many, many hosts out there do reverse lookups, not expecting the target they're investigating to be aware of this
  - 38K name servers doing lookups
    - Some who are invisible to direct querying
  - Exponential curve of requests – most only have 1, maximum has 14,221
    - Cable modem DNS
  - Warning: Possible to backwards map from scanned IP to elicited PTR request by shuffling scan orders and looking for correlation between a particular IP being contacted and the PTR request returning!

# As long as we're validating the infrastructure...

---

- DNS w/o DNSSec requires the infrastructure not to corrupt its data
  - This is a good reason to revive large scale high speed tracerouting
- Is it possible to collect enough information to map all Internet routes in a matter of hours?

# Rapid Infrastructure Mapping HOWTO [0]

---

- 1) Collect a list of subnets that have at least one host with one service. This will be the destination canary.
- 2) Setting a “max\_ttl” value to your average distance to a host, transmit canary connection attempts w/ Scanrand from 1 to max\_ttl.
  - Run the scan such that the *last* byte of the IP address is maintained
    - This minimizes bandwidth load per subnet
  - Scanrand places the original TTL in the ipid – can be recovered
  - `scanrand2 -b2m -f hostlist+:53 -l1-$MAX_TTL -t0 -H -M1 -T infra_map > results.sql; cat results.sql | mysql dns`
    - 2mbit, select port 53 for each IP, scan up to maximum TTL, disable timeouts, output SQL to table name “infra\_map”. Then cat the file into mysql.

# Rapid Infrastructure Mapping HOWTO[1]

---

- 3) After importing the data into MySQL, reorder it back into normal-seeming traceroutes as such:

```
select trace_hop,trace_mid,trace_dst from newscan  
group by trace_dst,trace_mid order by  
trace_dst,trace_hop
```

```
-----  
1      209.200.133.225 12.10.41.178  
2      67.17.168.1    12.10.41.178  
3      67.17.68.33     12.10.41.178  
4      208.50.13.254   12.10.41.178  
5      12.123.9.86      12.10.41.178  
6      12.122.10.53    12.10.41.178  
7      12.122.9.129   12.10.41.178  
8      12.122.10.2    12.10.41.178  
9      12.123.4.153   12.10.41.178  
10     12.125.165.250 12.10.41.178
```

# Rapid Infrastructure Mapping HOWTO[2]

---

- 4) For each line in the mass traceroute, if the destination of the previous line is the same as this one, and if the hop number for the last line is one less than the previous line, then there can be assumed a link between the last midpoint and the present midpoint.
  - 1 a bar
  - 2 b bar
  - 3 c bar
  - 5 d bar
  - 1 a car
- Links can be assumed between a and b, and b and c.

# Rapid Infrastructure Mapping HOWTO[3]

---

## ■ OPTIONAL:

- 1) For each IP where a hop was found at max\_ttl, scan that IP up to a new max\_ttl
- 2) Scanrand allows scans to come from different points in the network, but arrive at the same collector. Use this to collect routes invisible from your own position.
- 3) Schedule “gap filling” scans for packets dropped during an initial run
- 4) Attempt to source route packets, though so many networks block them
- 5) Graph the results!
  - **DEMOS**

# Rapid Infrastructure Mapping: IPv6?

---

- I need a high speed lab on the IPv6 backbone ☺
- Saturating the IP space gets replaced with discovering pockets of populated addresses
- Traceroute, DNS most obvious legitimate mechanisms for discovering populated space
- Some IP options – source routing, potentially spoofs from multicast may help

# It's Alive!!!

---

- Opte.Org dataset in realtime is neat – but how do we make it useful?
  - C++ now, Python will be workable *very soon*
- The plan is to import all data, streaming and otherwise, into a large scale graph manipulation framework.
  - Boost Graph Library allows very large scale operations w/ very generic data types
  - Dan Gregor, one of the authors of BGL, has *specifically* helped with this work

# Why use graphs?

---

- There's more than just pretty pictures
- Ultimately, services that do not adapt to broken networks are isolated onto *very broken networks*
- Traditional adaptation mechanisms completely fail, since we're only sending a few packets to every host
  - What we need are canaries – they are sent, a few a second, to each hop we're scanning through. When the canaries die, we know we've overloaded that network.
  - Graphs work **perfectly for this**
    - For every destination, we know which routers will get a traffic spike from us communicating with it
    - For every router we are canary-monitoring, we know which destinations we are now closer to
    - **We would thus be able to model outbound transmissions as a high pressure water system, against which taps may be made**
    - Demo of present progress level (visualizations only)

# Why Pictures

---

- A third of our brain is visual, and more of our decision making is visually modulated than we'd like to think.
- As proof – last year, I showed off audio over DNS. This year, video over DNS 😊
  - Large window, rate based codec. Much faster than TCP at same loss rates, but ... written in Perl, all client side logic
    - Can we please start monitoring DNS on our networks?
  - Demo

# Done

---

- That's all folks 😊
- Any questions?