



Stack Black Ops Black Hat Federal 2003

**New Concepts for Network Manipulation
Dan Kaminsky, CISSP**

IP Telephony

Contact Centers

Unified Communication

Services

History:

Peace through Superior Firepower

- **History**
 - **“Black Ops of TCP/IP” @ Black Hat 2002**
 - **“We’re not getting new networks – so if we want new capabilities, we need to find ways of teasing desired (if unexpected) functionality from established systems.”**
 - **Paketto Keiretsu 1.0, Nov. 2002**
 - **Prove concepts**
 - **Paketto Keiretsu 2.0: Imminent**
 - **Prove concepts...useful.**

Present: New Code

- <http://www.doxpara.com/paketto-1.999-2.tar.gz>
- **These Slides:**
 - http://www.doxpara.com/SBO_Fed.ppt

Starting Simple at Layer 7: Executable Deployment

- *Is it possible to greatly simplify the deployment of critical updates to Windows desktops, without installing any code?*
- **Yes. Do this:**
 - `net view | findstr "^\\\\" > hostlist`

`for /f %i in (hostlist) do start psexec \\%i -c -i -u DOMAIN\administrator -p hotfix.exe -u -q >> patch.log`
 - **This patches every Windows desktop logged into DOMAIN.**
 - Uses PSEXEC From www.sysinternals.com to transfer patch, execute it, and log the results.
 - Scales to a few hundred nodes. More, and you overload the process table.
 - There are solutions, but they're not two liners.
 - ***Warning: Someone breaking into your PDC can do this to you, without knowing any special password or exploiting anything but the PDC. This is not theoretical.***
- **But what if we're trying to deploy code to people not logged into a domain?**

Layer 7: Generic ActiveX Encapsulation

- *Is it possible to use ActiveX to deploy something besides spyware, without writing custom applications / wrappers?*
- **Yes – Any win32 application – any .EXE file! -- can be cryptographically signed and used instead of a genuine ActiveX object**
 - **Object GUID is not checked; code only needs to be self-signed**
 - **Applications that require multiple files simply require a CAB to be generated containing all that is needed, and a simple .INF file that describes which executable to launch**
 - **Examples: <http://www.doxpara.com/apps>**
 - **Stinger: Network Associates' cleaner for Nachi, Blaster, and SoBig**
 - **Putty, OpenSSH, etc.**

Layer 7: Generic ActiveX Encapsulation

- *Certificate Generation:* `makecert -n "CN=Foobar" -sv key.pvk key.cer`
- *Certificate Conversion:* `cert2spc key.cer key.spc`
- *Code Signing:* `signcode -v key.pvk -spc key.spc -n "Random File" -t http://timestamp.verisign.com/scripts/timestamp.dll foo.exe`
- *HTML to Embed:*
 - `<OBJECT> CODEBASE="foo.exe" CLASSID="CLSID:DE70D9E3-C55A-11CF-8E43-780C02C10128">`
`</OBJECT>`
- *That's ALL! But that's not too interesting...*

Layer 7: Generic ActiveX Encapsulation

- *INF Generation: Create a file named whatever.inf, fill it with...*
 - [Setup Hooks]
hook1=hook1
[hook1]
run=%EXTRACT_DIR%\startup.exe
[Version]
; This section is required for compatibility on
both Windows 95 and Windows NT.
Signature="\$CHICAGO\$"
AdvancedInf=2.0
- *CAB Generation:*
 - cabarc -p -r n foo.cab *
- Then sign the cab. That's it!

LAYER 2: ARP vs. IP

The Problem

- *Is it possible to acquire a usable IP address on a network that lacks a DHCP server?*
 - **DHCP server provides:**
 - Free IP on LAN
 - Address of Upstream Gateway
 - DNS
 - **DNS can usually be some externally available default, but an IP and Gateway are needed**
 - **Classic approach**
 - Sniff for broadcasted ARPs, find “gaps” between claimed IP addresses, attempt static mapping
 - This often gets you an IP – but how do you find the gateway? Usually at borders of subnet...

LAYER 2: ARP vs. IP

The Protocol

- **ARP: Translator between MAC and IP**
 - **If target in subnet, translate target IP, send to MAC.**
 - **Your friend is in the city; you tell the cab driver to take you to his house**
 - **If not in subnet, translate IP of router, send to MAC of router.**
 - **Your friend is across the country; you tell the cab driver to take you to the airport**

LAYER 2: ARP vs. IP

What We Can Do

- **Router Detection**

- Router will route even if target was in subnet

- Your friend is in the city. You go to the airport...and hail a cab back to your friend's house. Now you know it's an airport – if it was some random person's house, you wouldn't be able to hail a cab.

- **Subnet Detection:**

- Routers need to ARP too!

- Router will ARP for us only if IP is in subnet range

- » Cab driver at airport won't take you anywhere if it's not in the city

- Subnets aren't randomly distributed

- Binary search across `ip_dst` will thus quickly show subnet boundaries

- But what if all IP addresses are taken?

LAYER 2: NAT 101

- **How do we normally get new IP addresses?**
 - **NAT**
 - **Specified when more hosts need to be networked than there are IP addresses to grant them**
 - **Packets sent from each host need to be returned to each host**
 - **Requires control of at least one IP that the network respects**
 - **Packets sent from this IP will route out**
 - **Packets sent to this IP will route in**
 - **NAT presents a single IP externally, and a private range internally.**
 - **Source of packets is changed to what the outside world respects**
 - **When packets return, source of packets is changed back**
 - » **“State Table” keeps track of who talked to Yahoo on what port**
 - **The network is out of IP addresses – they are all in use – we fail NAT’s requirement for an externally respected IP address**

Stuck In The Middle With You

- **Just because there aren't any respected IP addresses *free* doesn't mean there aren't any *available*...**
- **ARP Man In The Middle**
 - **Tell the client you are the router**
 - **Tell the router you are the client**
 - **All traffic between both hosts must now travel through you – you select what actually passes**
 - **You now control traffic for an *externally respected IP address***
 - ***You now meet the requirements of NAT***

NAT-in-the-Middle

- *Is it possible to acquire a usable IP address on a network that lacks a DHCP server?*
 - **Yes: Use an ARP Man-In-The-Middle attack to insinuate yourself between an existing node and the gateway. Use traditional NAT mechanisms to multiplex your own sessions into the IP address of the hijacked node.**
 - **Normal Incoming and Outgoing Streams are routed *as normal***
 - Individual incoming ports can be hijacked
 - **Inserted Outgoing Streams: Responses routed back to client requesting stream insertion**
 - **Uses**
 - **Breaks web-based WiFi security models**
 - MAC Address Filtering limits damage – users can still be hit on signin
 - **Allows for very nice planned migrations – some % of new incoming sessions go to new machine, old ones gracefully stay.**

Layer 3: Scanrand Observations

Scanrand 1.x

- **High speed port scanner / route tracer**
 - **Stateless design, embeds cookie in SYN reflected in SYN|ACK or RST|ACK**
 - **Sender and receiver don't need to be the same host ("Split Mode")**
 - **Able to analyze ICMP replies to determine original IP/L4 source**
 - **ICMP errors clone entire IP packet (including options), first eight bytes of TCP/UDP/ICMP/etc**

Layer 3: Scanrand Observations

Scanrand 1.x [TTL's]

- **Able to use TTL to estimate how far a packet travelled**
 - **Useful for network graph generation, DDoS tracing, etc**
 - **Very useful for peer-to-peer / grid computing designs: All traffic already contains traces of how many hops it took to get there!**
 - **Often shows results of network level trickery**
 - **Third parties can't easily know appropriate initial TTL to use, so their packets stand out vs. legitimate traffic**

Scanrand Returns #1: Email Hijacking

```
root@arachnadox:~/new_talk# scanrand local.doxpara.com
```

```
UP: 64.81.64.164:80 [19] 0.092s
```

```
UP: 64.81.64.164:25 [04] 0.095s
```

```
UP: 64.81.64.164:443 [19] 0.099s
```

```
UP: 64.81.64.164:22 [19] 0.106s
```

```
UP: 64.81.64.164:993 [19] 0.121s
```

```
root@arachnadox:~# telnet www.microsoft.com 25
```

```
Trying 207.46.134.155...
```

```
Connected to microsoft.com. Escape character is '^]'.
```

```
220 ArGoSoft Mail Server Pro for WinNT/2000/XP, Version 1.8 (1.8.2.9)
```


Scanrand Returns #2: Hopcount Desync

```
root@arachnadox:~# scanrand -b1k -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
DOWN:    64.81.64.164:8000   [22]    0.570s
DOWN:    64.81.64.164:31337  [22]    0.636s
```

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.

Scanrand Returns #3: Serverless NAT Identification

```
root@arachnadox:~# scanrand -l1-3 www.doxpara.com
```

```
001 =      172.16.0.1|80      [01]    0.024s(      172.16.1.97 -> 209.81.42.254      )
002 =      216.137.24.1|80    [01]    0.030s(  216.137.24.246 -> 209.81.42.254      )
003 =      216.137.10.45|80   [03]    0.100s(  216.137.24.246 -> 209.81.42.254      )
```

```
root@arachnadox:~/new_talk# scanrand -l2 -vv www.doxpara.com
```

```
Stat|====IP_Address==|Port=|Hops|==Time==|=====Details=====|
```

```
SENT: 209.81.42.254:80 [00] 0.000s Sent 40 on eth0:
```

```
IP: i=172.16.1.97->209.81.42.254 v=4 hl=5 s=0 id=2 o=64 ttl=2 pay=20
```

```
TCP: p=193->80, s/a=3012956787 -> 0 o=5 f=2 w=4096 u=0 optl=0
```

```
Got 70 on eth0:
```

```
IP: i=216.137.24.1->172.16.1.97 v=4 hl=5 s=0 id=35273 o=0 ttl=127 pay=36
```

```
ICMP: IP: i=216.137.24.246->209.81.42.254 v=4 hl=5 s=0 id=2 o=64 ttl=1 pay=20 ICMP: TCP: p=193->80,
s/a=3012956787
```

```
002 = 216.137.24.1|80 [01] 0.049s( 216.137.24.246 -> 209.81.42.254 )
```


Multihomed Node Detection

- *Is it possible, from the LAN, to detect clients that are directly connected both to the internal, firewalled LAN and the outside world?*
- **Yes – use scanrand in Split Mode:**
Fake a scan on the internal network from the outside world, then pick up replies that don't get stopped by the firewall
 - **Internal network is flooded with requests spoofed from external network**
 - **Nodes receive request, check routing tables to see where to send replies**
 - Replies routed through firewall are dropped (we assume)
 - Replies routed through unprotected link will leak out (w/ IP)
 - You control node on external IP, watch all packets come to it

Multihomed Node Detection #2: The NAT Case

- *Is it possible to detect clients that are indirectly connected, through a NAT, both to the internal, firewalled LAN and the outside world?*
- **Yes – but different requests may need to be used**
 - **Standard TCP SYNs will elicit SYN|ACKs or RST|ACKs that don't match up with anything in the NAT State Table**
 - **ICMP Pings (which can reflect an almost arbitrary amount of data) may also have state table issues**
 - **UDP is symmetric in and out (request and response are indistinguishable on the wire)**
 - **UDP/137 (SMB) may work – though is firewalled by certain DSL Providers**
 - **UDP/161 (SNMP) would work, but doesn't exist on most clients**
 - **UDP/113 (RPC) should work best**
- **NAT is less worrisome – no incoming access by default**

Scanrand 2.x: Architectural Improvements

- **Much more efficient internal architecture: Libpaketto**
 - **Unified interface for packet reception, parsing, manipulation, and retransmission**
 - **Paketto tools are mostly front-ends – actual intelligence is being engineered into libraries on day one for reintegration into larger scale systems**
- **Much, much, *much* faster**
 - **Georgia Tech: “We had to cap it at 100Mbit because it was overloading our GigE backbone”**
 - **These are 64 byte packets.**
 - **I didn’t believe him either at first.**

Scanrand 2.x: Bandwidth Management

- **Transmission Quanta**
 - Scanrand 1.x slept a small amount between each packet.
 - Linux does not like being asked to sleep 23,000 times per second.
 - Scanrand 2.x determines how many packets it is allowed to send per 20ms
- **Transmission Delay Measurement**
 - Sending packets takes time – it's "extra sleep" that needs to be accounted for
 - Even with nonblocking writes, eventually the kernel queue has to fill and the send call must return EAGAIN ("Try Again, I'm Too Busy Now").
 - Now we actually take that time into account

Scanrand 2.x: Overload Protection

- **Overloads: Why?**
 - Scanrand is stateless
 - Operating systems are robust
 - Firewalls and IDS's should scale, since they're built to handle traffic from way more machines than an individual stack is
 - They sometimes don't
- **Scanrand 1.x: Just scan slower – even 10k/s is 150 hosts per sec**
- **Scanrand 2.x: Proactive Resets**
 - The connection state opened by a SYN can be closed by a RST
 - Instead of waiting for a remote host to send us something that might make us RST ... we just send one after a short delay
 - Implementation is simple: Just matching “cleanup” thread on a small time delay
 - This frees resources allocated by the SYN, but may prevent a valid response.

Scanrand 2.x: Stateless Latency Detection

- **Latency:** Amount of time elapsed between stimulus and response
 - Difficult for stateless systems to track – the receiver didn't keep track of when it sent something, so how should it know how long it took to return?
- **Scanrand 1.x:** Measure time from start of listener
 - Worked well for very fast scans, because all packets would be out within 10-20ms
 - Scanrand sender can send as slow as you like
 - Scanrand receiver would detect tremendous lag
- **Scanrand 2.x:** Place timestamp in TCP Source Port
 - Port = 16 bits = 65K possible latency measurements
 - 6 seconds worth of 0.1ms clockticks or 65 seconds worth of 1ms ticks
 - Testing TCP Timestamp option – if it consistently isn't blocked or cleared, might default to that

Scanrand 2.x: Stateless Latency Detection

Latency Sampling

- **Scanrand allows iteration across hosts, ports, and hopcounts**
- **Can scan across a large portion of the net with a restricted hopcount, purely to determine directions in which latency is higher than desired**
 - **Can then run a deeper search to quickly determine precise routers that are flapping**

Scanrand 2.x: Stateless Latency Detection

Latency Sampling

```

• root@rotation2:~/pk/src# ./scanrand2 -l10,15 64-74.1-254.1.1 -b100k
• 010 :      207.46.37.2:80      [08]  11.7ms (  64.60.246.202 -> 64.4.1.1      )
• 010 :      144.232.3.165:80    [12]  11.8ms (  64.60.246.202 -> 64.5.1.1      )
• 010 :      152.63.1.45:80     [10]  11.5ms (  64.60.246.202 -> 64.10.1.1     )
• 010 :      152.63.0.253:80    [10]  11.7ms (  64.60.246.202 -> 64.11.1.1     )
• 010 :      144.232.3.158:80    [12]  12.2ms (  64.60.246.202 -> 64.6.1.1      )
• 010 :      208.172.147.61:80   [10]  11.8ms (  64.60.246.202 -> 64.14.1.1     )
• 010 :      208.172.147.61:80   [10]  13.7ms (  64.60.246.202 -> 64.22.1.1     )
• 010 :       66.80.133.18:80    [10]  15.1ms (  64.60.246.202 -> 64.7.1.1      )
• 010 :       66.109.3.198:80    [13]  15.5ms (  64.60.246.202 -> 64.8.1.1      )
• 010 :       67.17.72.105:80   [11]  25.7ms (  64.60.246.202 -> 64.21.1.1     )
• 010 :      165.117.200.122:80  [10]  23.4ms (  64.60.246.202 -> 64.48.1.1     )
• 010 :       12.122.10.26:80   [10]  23.6ms (  64.60.246.202 -> 64.57.1.1     )

```


Scanrand 2.x: Stateless Latency Detection

Temporal Fingerprinting

- **TCP Repairs Broken Connections**
 - If a packet is dropped, it will retry
 - “Hello? ... Hellllo? Hello?” <CLICK>
 - How many Hellos? How long inbetween them?
 - It varies from person to person, and from TCP/IP stack to TCP/IP stack
 - Discovered by Franck Veysset et al, demo'd with RING
- **Because scanrand can statelessly determine latencies, even a slow scan across an entire network will correctly detect operating systems**
 - Because leakage is not within the packet but related to other packets, this is hard (not impossible) to scrub

Scanrand 2.x: Database Integration

- **Scalability**
 - Scanrand as an engine could scan the Internet
 - Scanrand as a UI couldn't
 - Reports are important.
 - Hundreds of thousands to millions of lines of scan returns shouldn't be parsed with less, grep, and wc -l!
 - Very ugly Python code was written by others to compensate
- **Why DB? Because the world doesn't need another homegrown hash table**
 - Which Database? How to integrate with it?

Scanrand 2.x: Database Integration

SDBC

- **Which DB? MySQL, PostgreSQL, Oracle, SQLite, SAP, Informix...**
 - **So many API's for scanrand to potentially support...or not?**
 - **We've been using stdout already...why not simply output raw SQL?**
 - **Stdout: The ultimate database abstraction layer**
 - **Allows us to insert data into any number of databases**
 - **API doesn't need to be linked with scanrand as a client**
 - **More portability**
 - **SQL easier to forward to remote databases**
 - **We'll come back to this**

Scanrand 2.x: Database Integration

SQL Output

- ```
./scanrand2 -r -l10,15 64-74.1-254.1.1 -b100k -T scan -H -M1
create table scan (abs_tv_sec integer unsigned, abs_tv_usec
integer unsigned, rel_tv_sec integer unsigned, rel_tv_usec integer
unsigned, stat char(5), src varchar(64), dst varchar(64), port
integer unsigned, hopcount integer unsigned, trace_hop integer
unsigned, trace_src varchar(64), trace_dst varchar(64), trace_mid
varchar(64));
insert into scan values(1063900109,545527,0,7000,'010 ',
'64.60.246.202','64.4.1.1',80,8,10,'64.60.246.202','64.4.1.1',
'207.46.37.2');
insert into values(1063900109,548329,0,10000,'010',
'64.60.246.202','64.5.1.1',80,12,10,'64.60.246.202','64.5.1.1',
'144.232.3.165');
```



## Scanrand 2.x: Split Mode Formalized

- **Traits of the senders:**
  - Ephemeral – Possibly Cron Jobs
  - Job determined at start -- “Fire and forget”
  - Located anywhere; sync their source IP and their cryptographic seed to that of the receiver
    - `scanrand -S -s key -b100k 10.0.1-20.1-254:quick`
- **Traits of the receiver:**
  - Permanent – Possibly Daemonized
    - `scanrand -L -s key -t0 -T newtable -H -M1 | mysql db`
  - Job is to react to responses triggered by senders
    - Still stateless – prints lines of SQL instead of user-friendly text
  - Located in a centralized, well known location.
- ***Scanrand does not handle its own reporting (should it?)***



## Scanrand 2.x: Sample SQL Queries

- **Show all nodes running Windows File Sharing (RPC)**
  - `select dst,port from db where stat = 'UP' and port = 139;`
- **Show all unique host/port combinations**
  - `select dst,port from db where stat = 'UP' group by dst,port;`
- **Show nodes with the greatest number of open ports**
  - `select dst,count(port) from db where stat = 'UP' group by dst order by count(port);`
- **Show all Linux/BSD machines using nothing but their temporal fingerprint**
  - `select dst,port from db where rel_tv_sec > 23;`
- **Much, much more is possible now that this data is in a *real* environment**
  - **Excel fans who don't like ODBC can be happy too**
    - Scanrand can output to CSV
    - `echo "select * from scanrand" | mysql db > read_me_in_excel.txt`



## Scanrand 2.x: TODO

- **Support for OS Fingerprinting**
  - One of many types of multi-packet queries
  - **Sender: Send set of packets that elicits OS-specific responses**
    - Nmap/xprobe/libsf/etc
  - **Receiver: Parse results**
    - No replies: Host unreachable (include ICMP Unreachable parsing)
    - Some replies: Host needs more packets – trigger sender
    - All replies: Compile results into format assumed by nmap/xprobe, pass struct to their evaluation routine. Incorporate results.
    - This is a “state reconstruction” approach
- **Transmission Logging – sender transmits to DB over SSH**
  - Not appropriate for flood scans, since most traffic is wasted. Critical for monitoring scans, which can be extremely targeted. Syntax will shift to better support these very different uses.
    - “Flood lightly, target deeply”



# The Return of SSH Scanrand

- SSH and scanrand complement each other well
  - Order remote host to initiate scan that will return results to local listener:
    - `ssh user@remote_host scanrand -i local_host 10.0.1.1-254:quick`
  - Initiate remote scan, but this time have the results parsed remotely and forwarded into the local database
    - `ssh user@remote_host "scanrand -M1 -b1m 10.0.1.1-254:quick" | mysql db`
  - Forward results of local scan into remote database over a secure link
    - `scanrand -M1 -b1m 10.0.1.1-254:quick | ssh user@db_server "cat | mysql db"`



# The Return of SSH

## tcpdump

- **For the paranoid: Run sniffing component and parsing component of tcpdump as separate users**
  - `sudo tcpdump -w - | tcpdump -r -`
    - Of course, an attacker can break the tcpdump account, run “sudo tcpdump” proper, and trigger the attack in the root account. This can be mitigated by using linkcat instead.
      - Linkcat: Bidirectional tcpdump that dumps to hex or libpcap
- **Sniff remotely, over an encrypted channel:**
  - `ssh user@host sudo tcpdump -w - not port 22 | tcpdump -r -`
  - Can parse `$SSH_CLIENT` to filter out only the sniffing backchannel



# The Return of SSH

## Wormsmashing [1]

- **Targeted worms have a problem**
  - **If they hit an IP, it can be quickly blackholed**
    - **Whitehouse.Gov DDOS**
  - **If they hit a DNS server, they cannot hide their identity on the LAN**
    - **Their DNS request cannot come from a spoofed IP, because then they won't receive the response**
    - **MS.Blaster spoofed source IPs, but couldn't spoof DNS**



# The Return of SSH

## Wormsmashing [2]

- Connect to remote DNS server and report each lookup of [www.windowsupdate.com](http://www.windowsupdate.com)
  - `ssh user@dns sudo tcpdump -w - -s 1500 port 53 | tcpdump -l -n -s 1500 -r - | perl -nle 'print $1 if /\s(\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3})\b.*A\?.*windowsupdate\.com/' >> infected.log`
  - `cat infected.log | sort | uniq`
- Connect to remote DNS server and report each lookup of [www.windowsupdate.com](http://www.windowsupdate.com) (SQL Output)
  - `ssh user@dns sudo tcpdump -l -n -s 1500 port 53 | perl -nle 'print "insert into table foo {$1}" if /\s(\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3})\b.*A\?.*windowsupdate\.com/' | mysql db`
- SSH is used to prevent execution of “untested Perl” on production servers



## DNS Counterattack: Dark Ranges

- **What to return for a lookup assumed to be executed by a worm?**
  - **127.0.0.1: Localhost. Old standby. May cause host to respond in strange and negative ways (since traffic is allowed to flow).**
  - **“Dark Ranges” – IP’s that certain stacks utterly refuse to talk to**
    - **0.1.2.3: Silences Linux**
    - **127.0.0.0: Silences Windows**
    - **250.1.1.1: Silences Linux and Windows**
- **Not all who send traffic are wormed...**



## **Layer 4: Bandwidth**

### **The Ultimate Almost**

- **Bandwidth – end to end, from the core to the last mile – is the ultimate cheap-but-non-zero-cost commodity**
  - **Even power usage for business is strongly correlated to increased revenue**
    - **Production, extra hours of availability, etc.**
  - **Popularity is punished, mostly only socially rewarded**
- **Available bandwidth can be transient but widely distributed**
  - **P2P programmers noted this**
- **It ought to be possible to harness available bandwidth, quickly, fluidly, and affordably**



# Bandwidth:

## Mirror Mirror, Fall Off The Wall

- **Mirror Selection is a guessing game**
  - What's actually up?
  - What will continue to be up?
  - What's fast?
  - What's synced?
  - What's in a nice position (the California Ballot effect)
- **Desires:**
  - Users want valid content.
  - Content providers want to validate users get what they want
    - Who says a mirror actually served anything?
  - Someone Else has bandwidth to spare.
    - The content provider is bleeding from all those users!



## Bandwidth Brokering

- *Is it possible for a single host to do load balancing across nearly arbitrary network boundaries, without any special code on the client?*
- **Yes – by transforming the server into a mere redirector of client-provided packets, and having the actual (and anonymous) servers spoof the source IP of the redirector when providing the payload**



# Bandwidth Brokering: Load Balancing Without Limits

- **“Server” replaced with redirector**
  - Doesn’t actually serve data
  - Forwards incoming traffic to an **“Anonymous Server”** according to a session-consistent rule
    - Rules:
      - Stateless: Source IP, Source Port
      - Stateful: Who has the least number of active streams
    - Forwarding:
      - Change Destination IP, *not source IP*
      - Recalculate Checksum
      - Send to Anonymous Server, which can be on any network
- **Anonymous Server receives request, apparently from client. Serves request. Lump in IP stack changes Source IP to that of redirector.**
- **Client sends data to IP. Client receives data from IP. Client doesn’t know or care how this works.**



# Bandwidth Brokering: The President's Mail [1]

- **The President's Mail**
  - President couldn't possibly respond to every letter
  - Could hire a herd of interns to do so
  - Each response would be ghostwritten, stamped "by the president", and sent off from the White House.
  - *This is basic load balancing.*



## **Bandwidth Brokering: The President's Mail [2]**

- **The President's Mail...Brokered**
  - **Could also outsource Intern Herding to China**
  - **Letter arrives in Washington, logged as received**
  - **Drop-shipped to one of several Chinese suppliers**
  - **Doesn't go back to DC – response is mailed directly to original writer**
  - **If writer not satisfied, White House receives another letter...**



## **Bandwidth Brokering: Specifics [1]**

- **Client doesn't need to know a thing**
  - May notice shifting TTL's, as Little Timmy might notice colorful postmarks
- **Redirector only needs to handle incoming traffic, which can be much lower than outgoing**
  - If protocol bandwidth profile leans more towards uploading than downloading, *and* if this isn't a play to distribute CPU instead of Bandwidth (MOSIX), then brokering is inappropriate



## Bandwidth Brokering: Specifics [2]

- **Anonymous Server can serve *any* IP-based protocol**
  - You can load balance pings if you like
  - **Protocols where IP address is in payload require special handlers (IPSec, FTP)**
    - Special handlers allow interesting capabilities, like session migration from a slow/lost mirror to a fast one
  - **TCP-Based Protocols work best**
    - HTTP: SQUID “Just Works”
    - Shoutcast MP3 Distribution
    - Oracle/MySQL/etc.



# Bandwidth Brokering: TCP Session Tracking with Ackmon

- **TCP: Byte Oriented Protocol**
  - It doesn't matter if you send 1000 bytes in ten segments or in one; the stack ensures 1000 bytes, not n segments, got through successfully
- All TCP packets have a SEQ# and an ACK#
- Every byte sent increments SEQ#
- Every byte received increments ACK#
- Redirector received TCP ACKs from client acknowledging data successfully transmitted
  - Acknowledgements contain rapidly increasing ACK#
  - Redirector can thus measure the flow of data from the anonymous server to the redirector *without being party to it*
- Limitations: You have to trust your anonymous servers
  - You also have to trust your upstream routers.



## Layer 5: SSL vs. IDS Conflict

- **SSL vs. IDS: The Eternal Conflict**
  - **SSL Annoys Me.**
  - **Certificate compromise is extraordinarily damaging – all past data lost, all future data lost, attacker only needs to passively monitor or sniff**
- **IDS Annoys Me.**
  - **“We’re under attack!” “That’s nice, dear.”**
  - **I respect those who have faith in both**
- **The conflict between the two annoys me most!**



# **SSL vs. IDS**

## **The problem at hand**

- **The Scientific Summary**
  - **IDS monitors the network traffic between the trusted and the untrusted, watching for attacks**
  - **SSL encrypts the network traffic between the trusted and the untrusted, blinding all watchers except for the presumably vulnerable endpoint**
  - **Choice: Suppress passive and suffer active, or suppress active and suffer passive.**
- **The Bottom Line:**
  - **SSL: The network is evil. I trust the hosts.**
  - **IDS: The hosts are stupid. I trust the network.**
  - **Sysadmins: You're both right. But I need you both.**



# **SSL vs. IDS**

## **Bad Solution #1**

- **Certificate Transfer**
  - **IDS gets a copy of the cert**
  - **Violates 1st Law of Private Keys: Thou Shalt Not Transport Thy Private Key**
    - **Impossible if you're using FIPS140 equipment**
  - **Adds RSA decryption load to IDS, which is already scrounging for cycles**
  - **ssldump can be pressed into service today to support this for SSL3**
    - **Attack: Switch to SSL2**



# **SSL vs. IDS**

## **Bad Solution #2**

- **Mix IDS w/ Inline SSL Accelerators**
  - IDS lives between accel and server farm
  - IDS's are famously DoSable – use hubbed net
  - Servers never see cryptography (can't make any decisions based on it)
  - Issues with HTTP rewriting
  - Puts plaintext on a wire



## SSL vs. IDS

### My Solution

- *Is it possible to allow an IDS to function reasonably in a SSL-protected environment, without transferring the key or losing the end-to-end nature of SSL?*
- **Yes – by recognizing that SSL negotiates keys on a per session basis and that these keys, not the entire cert, are sufficient for the IDS to complete its task.**



# SSL vs. IDS

## Session Key Transfer

- **SSL negotiates per-session keys**
  - Instead of transferring Certificate (aka all session keys, forever), transfer keys on a per-session basis
  - Unique keys are used in each direction – could just grant IDS the client->server key, preventing regulatory conflicts with plaintext readable outside the secure facility
    - Content from outside users is always suspect
  - Separate Read and Write keys – can prevent IDS from being able to spoof traffic, at cost of making it more difficult (not impossible) for IDS to validate traffic
    - Can look for sudden closing of stream by endpoint receiving bad data
- **IDS receives key**
  - Doesn't need to do RSA decryption
  - Bulk ciphers, even 3DES are only mildly CPU intensive (SSL accelerators usually only do modular exponentiation for RSA)



# SSL vs. IDS

## Key Transfer Mechanics

- This is not theoretical; code works *today*
  - Built using ssldump, which allows ssl to be remotely decrypted given possession of the certificate
- Today: Key delivered using SSH
  - Ssldump split
    - Local process has cert; decrypts session keys using it
    - Keys transmitted to remote ssldump via SSH tunnel
    - Remote ssldump, without possession of cert, is able to decrypt traffic in directions keys provided for
  - This does not scale
    - Time used to synchronize key delivery and session selection
      - Never do this
    - Only one SSL server can talk to the IDS at a time
    - SSL server needs to know precisely where IDS is
    - IDS needs to receive incoming SSH sessions



# SSL vs. IDS

## Advanced Key Transport

- **Session Key Transmission over DNS**
  - **Session Key Material requires ~120 bytes**
    - This fits in a DNS *request*, even encrypted and padded
  - **Idea is to use Asynchronous DNS library to “resolve” name that IDS can decrypt to access a given session**
  - **Supports both types of IDS**
    - Transparent IDS will see request on way to DNS server
    - Explicitly addressed IDS will be the authoritative provider for the domain that keys are requested from/to.
      - SSL nodes don't need to know precisely where that is, nor do they find out
  - **Enforcement**
    - IDS can block session from proceeding unless key is received and validated
  - **Implementation**
    - **OpenSSL can be patched (even at runtime) to “leak” keys**



# SSL vs. IDS

## A Completely Different Approach

- **Plaintext Forwarding over Encrypted Tunnel**
  - “I got this message from a user...”
    - Optionally: “Should I respond?”
    - Adds latency if each message needs to be authenticated
  - Relatively high bandwidth
  - Doesn’t require interfacing with crypto engine, or even web server
    - Can be built into web applications, which are necessarily passed the web request of the client
    - Totally immune to dissynchrony
    - Can be even more selective about what traffic to expose / verify
  - Disadvantage: Only really works for HTTP, not *nearly* as cool



# Volumetric Visualization

- **Volumetric Rendering**
  - **“Bitmaps with Voxels”**
  - **Often used for CT scans, MRI’s, other scanning mechanisms that acquire detailed data slices**
    - **Medical world filled with useful code that rarely escapes**
  - **2002: Phentropy with OpenQVIS, first known use of a volumetric renderer to display abstract characteristics of an arbitrary data source**
    - **Demo**



# Advantages of Volumetrics

- **Fixed Complexity**
  - Arbitrary data loads get quantized into the renderer
    - Much harder to render a hundred points
    - Much easier to render one million points
- **Psychologically Rich**
  - Human visual system is astonishingly capable of integrating large amounts of data about a point cloud in motion
  - Color also very useful



# The New Volumetric Renderer: Volsuite

- Packages for Windows, Unix, Mac
- Fast, flexible, free (Open Source)
- *Full Color Renderer With Alpha Channel*
- Potential Inputs
  - Phase Space Comparison: Overlay 3 different data sources w/ RGB, see if entropic characteristics match
  - Packet Dimensions: IPID vs. TCP Source Port vs. Last Byte of Destination IP
    - See Shoki Packet Hustler
  - Video Sources
    - $2D+Time = 3D$ : Can translate between temporal and spatial