Stack Black Ops
Black Hat USA 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, Nov. 2002
    - Scanrand – High Speed Network Auditor
    - Minewt – Userspace NAT Router
    - Linkcat – Simple Network Interface
    - Paratrace – Parasitic TCP Traceroute
    - Phentropy – Zalewskian Entropy Analysis
  - Goal is to bring new tools to the table, keeping with the primary advantage of the defender
    - The defender need not be stealthy.

How: Regions of Analysis

- Regions of Analysis
  - Intersections between layers
    - Layers are never entirely independent -- what happens when redundant data disagrees?
  - Manipulation of assumptions
    - Systems necessarily assume certain things to be always true about their environment, because they usually are. What happens when they're not?
  - The Human Factor
    - Somebody has to use all this stuff; someone needs to process an increasingly large amount of information.   How can this information be compiled into a maximally useful form?

LAYER 2: ARP vs. IP

- *Is it possible to acquire a usable IP address on a network that lacks a DHCP server?*
  - **Classic approach: Sniff for broadcasted ARPs, find “gaps” between claimed IP addresses, attempt static mapping**
    - **ARP: Translator between MAC and IP**
      - **If target in subnet, translate target IP, send to MAC.**
      - **If not in subnet, translate IP of router, send to MAC of router.**
  - **New Techniques**
    - **Router Detection: Router will route even if target was in subnet**
    - **Subnet Detection:**
      - **Router will ARP for us only if IP is in subnet range**
      - **Subnets aren’t randomly distributed**
      - **Binary search across ip_dst will thus quickly show subnet boundries**
      - **But what if all IP addresses are taken?**

NAT-in-the-Middle

- *Is is possible to acquire a usable IP address when all routable addresses are already in use?*
- **Yes: Splice into existing ones**
  - **NAT allows multiple hosts to share the same externally viewable IP address**
  - **"NAT-In-The-Middle"**
    - **ARP MITM vs. an existing IP**
    - **Create second gateway router on L2 Subnet (using Minewt)**
    - **Outgoing packets to second gateway MAC are NATted to IP**
      - **Added to state table**
    - **Incoming packets that match entries in state table are NATted appropriately, those that don't go on to original IP holder**
    - **Also supports MAT-in-the-Middle – All hosts can share external IP**
    - **Like NAT-DMZ w/o inline router**

More ARP Tricks

- *Is is possible to detect a single-port, multi-host portscan across a switched LAN?*
- **Yes: Watch for the router to spew ARP Requests**
  - **Blind scans don't know about empty IPs**
  - **Empty IPs don't show up in ARP caches**
  - **So whenever empty IPs are hit by a router, they elicit a *broadcast* ARP from the router**
    - **Even though nobody can see everyone else being scanned, everybody can see the router preparing to do the scan**
    - **Data point for the threat model**
  - **Some routers (DSL) may flood entire subnet with ARPs regularly**
    - **Small number of IPs = Why wait until a request, just send out ARPs every 30 seconds and actively maintain the ARP cache.**

Raw Network Access: Linkcat

- Linkcat: Standard-I/O Interface to Ethernet
  - Allows very simple command line access to ethernet
    - Plan 9: Everything is a file
    - Unix: Everything is a file…or a really small tool that does one thing well
  - Works over SSH
  - New for Paketto2: Automatic Checksums – write reasonably correct packets, and not only will they be sent, but the checksum will be automatically corrected

Packet Zen #1: Strings

```
# lc –l00 –tp | strings --bytes=8
FastEthernet0/6
Cisco Internetwork Operating System Software
IOS (tm) C2900XL Software (C2900XL-H-M), Version 11.2(8)SA2, RELEASE SOFTWARE (fc1)
Copyright (c) 1986-1998 by cisco Systems, Inc.
Compiled Fri 24-Apr-98 10:51 by rheaton
cisco WS-C2924C-XLv
GET / HTTP/1.0
Host: www.doxpara.com
Accept: text/html, text/plain, text/sgml, */*;q=0.01
Accept-Encoding: gzip, compress
Accept-Language: en
User-Agent: Lynx/2.8.4rel.1 libwww-FM/2.14 SSL-MM/1.4.1 OpenSSL/0.9.6
HTTP/1.1 200 OK
Date: Mon, 07 Apr 2003 13:53:30 GMT
Server: Apache/1.3.26 (Unix) DAV/1.0.3 PHP/4.3.1
X-Powered-By: PHP/4.3.1
Connection: close
Content-Type: text/html
```

Packet Zen #1.1: Strings (without Linkcat)

```
# tcpdump -w - -s2000 | strings --bytes=8
M-SEARCH * HTTP/1.1
Host:239.255.255.250:1900
ST:urn:schemas-upnp-org:device:InternetGatewayDevice:1
Man:"ssdp:discover"
SSH-1.99-OpenSSH_3.4p1
M!T7blnbXwG
SSH-2.0-OpenSSH_3.4p1 Debian 1:3.4p1-4
=diffie-hellman-group-exchange-sha1,diffie-hellman-group1-sha1
ssh-rsa,ssh-dss
faes128-cbc,3des-cbc,blowfish-cbc,cast128-cbc,arcfour,aes192-cbc,aes256-cbc,rijndael-cbc@lysator.liu.se
yourmom2
yourmom2
JlJmIhClBsr
JlJmIhClBsr
EJEDEFCACACACACACACACACACACACA
 FHEPFCELEHFCEPFFFACACACACACACABO
\MAILSLOT\BROWSE
JlJmIhClBsr
JlJmIhClBsr
g,QString,QString,QSL
```

Packet Zen #2: Ping over Copy and Paste

```
root@arachnadox:~# ping www.news.com
PING news.com (206.16.0.136): 56 data bytes
64 bytes from 206.16.0.136: icmp_seq=0 ttl=243 time=61.4 ms

root@arachnadox:~# lc -l00 -p "icmp and host www.news.com"
00 90 4c 49 00 2a 00 02 2d 4c 47 08 08 00 45 00 00 54 00 00 40 00 40 01 a0 4b \
c0 a8 0b 1d ce 10 00 88 08 00 bb 21 c5 4b 00 00 3e 59 40 ed 00 04 0d 45 08 09 \
0a 0b 0c 0d 0e 0f 10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f 20 21 22 23 \
24 25 26 27 28 29 2a 2b 2c 2d 2e 2f 30 31 32 33 34 35 36 37
00 02 2d 4c 47 08 00 90 4c 49 00 2a 08 00 45 00 00 54 a6 c9 a00 00 f3 01 86 81 \
ce 10 00 88 c0 a8 0b 1d 00 00 c3 21 c5 4b 00 00 3e 59 40 ed 00 04 0d 45 08 09 \
0a 0b 0c 0d 0e 0f 10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f 20 21 22 23 \
24 25 26 27 28 29 2a 2b 2c 2d 2e 2f 30 31 32 33 34 35 36 37
```

Packet Zen #2: Ping over Copy and Paste

```
root@arachnadox:~# lc -m00 -l00 -p "icmp and host www.news.com"
00 90 4c 49 00 2a 00 02 2d 4c 47 08 08 00 45 00 00 54 00 00 40 00 40 01 a0 4b \
c0 a8 0b 1d ce 10 00 88 08 00 bb 21 c5 4b 00 00 3e 59 40 ed 00 04 0d 45 08 09 \
0a 0b 0c 0d 0e 0f 10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f 20 21 22 23 \
24 25 26 27 28 29 2a 2b 2c 2d 2e 2f 30 31 32 33 34 35 36 37
00 90 4c 49 00 2a 00 02 2d 4c 47 08 08 00 45 00 00 54 00 00 40 00 40 01 a0 4b \
c0 a8 0b 1d ce 10 00 88 08 00 bb 21 c5 4b 00 00 3e 59 40 ed 00 04 0d 45 08 09 \
0a 0b 0c 0d 0e 0f 10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f 20 21 22 23 \
24 25 26 27 28 29 2a 2b 2c 2d 2e 2f 30 31 32 33 34 35 36 37
00 02 2d 4c 47 08 00 90 4c 49 00 2a 08 00 45 00 00 54 76 b1 00 00 f3 01 b6 99 \
ce 10 00 88 c0 a8 0b 1d 00 00 c3 21 c5 4b 00 00 3e 59 40 ed 00 04 0d 45 08 09 \
0a 0b 0c 0d 0e 0f 10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f 20 21 22 23 \
24 25 26 27 28 29 2a 2b 2c 2d 2e 2f 30 31 32 33 34 35 36 37
```

Layer 3: Scanrand Observations

- Scanrand
  - 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
  - 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
  - Able to use TTL to estimate how far a packet needed
    - Useful for network graph generation, DDoS tracing, etc
      - Very useful for peer-to-peer / grid computing designs
    - 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 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 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)**

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 Ping" is necessary**
    - **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**
- **NAT is less worrisome – no incoming access by default**
  - **Spoofing <u>internal</u> packet from <u>external</u> IP detects DMZ**

The State of State

- Scanrand is able to acquire a massive amount of data without keeping track of anything…or is it?
  - State is maintained through stdout – logs are dumped, the reader integrates all the information visually
    - "…it is left as an exercise to the reader"
  - Stdout works because all information dumped on each line can be extracted from each single packet as it arrives
    - "A response doesn't just contain 'yes this port is up', but 'yes, 1.2.3.4, the IP address 4.5.6.7 is listening on port 8910, please inform the socket you've bound to port 555 that this is the case.'"
  - But packets do not exist in isolation…

“Hidden Bits Between The Packets”

- 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
- Can we do this with Scanrand?
  - Scanrand uses the kernel to RST incoming replies, so they stop coming
    - Usually this is good – cut off the flood
    - Well now we want the flood…but we don’t want to interface with some firewall rules.
- Solution: Use a different IP. But have the kernel serve the MAC!

Temporal Fingerprinting with Scanrand 1.x

```
root@bsd:~# arp -s 10.0.1.190 00:e0:18:02:91:9f pub

root@bsd:~# arp -an | grep 10.0.1.190
? (10.0.1.190) at 0:e0:18:2:91:9f permanent published [ethernet]
root@bsd:~# scanrand -i 10.0.1.190 -t0 -b100k 10.0.1.1-254:139
(OUTPUT SORTED)
  UP:        10.0.1.12:139    [01]    0.235s
  UP:        10.0.1.12:139    [01]    3.191s
  UP:        10.0.1.12:139    [01]    9.109s
(+3+6)   # Windows
  UP:        10.0.1.36:139    [01]    0.715s
  UP:        10.0.1.36:139    [01]    3.624s
  UP:        10.0.1.36:139    [01]    9.639s
(+3+6)   # Windows
  UP:        10.0.1.38:139    [01]    0.755s
  UP:        10.0.1.38:139    [01]    4.560s
  UP:        10.0.1.38:139    [01]   10.560s
  UP:        10.0.1.38:139    [01]   22.758s
  UP:        10.0.1.38:139    [01]   46.756s
(+4+6+12+24)  # Linux
```

What’s significant to realize is that *no individual packet is special,* but the timing of each leaks the operating system of all. **No zero-latency scrubber can hide this fact (this impacts NAT detection, since each individual TCP session can be independently fingerprinted.)**

State Reconstruction Theory

- *Is it possible to extract deep information from hosts without losing the raw speed of a stateless network scan?*
- **Yes – by introducing a database to collate scan results**
  - **Slow model: Scan a host, compile results, print out…scan another host, compile results, print out…**
  - **Fast model: Scan all hosts, enter all results into a *dedicated state management engine* (better known as a database), compile results, do a secondary *and smaller* scan if necessary, recompile results**
    - **Split Mode Redux: Sender and receiver processes are built with wildly different philosophies**
    - **Sender optimized for speed and deployability, receiver optimized for comprehensive reports**
      - **Receiver technically still stateless – dumping SQL for a DB engine instead of fprintf-formatted rows of text**

State Reconstruction HOWTO

- Why DB? Because the world doesn't need another homegrown hash table
- 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
    - Stdout doesn't work too well (if at all) for report generation
    - Does work over SSH
      - Security issue: Passing raw SQL allows for injection attacks; may want to locally parse packets into SQL for untrusted servers

State Reconstruction Strategies

- Generic Model (Example: OS Fingerprinting)
  - Identify set of packets that elicits uniquely identifiable hosts
    - Nmap algorithm
    - New xprobe work from Ofir Arkin
    - Temporal fingerprinting
  - Insert not scan but scan configuration into a table
  - Compare scan results to scan config
    - No replies on host in range: Host unreachable (include ICMP Unreachable parsing)
    - Some replies on host in range: Host needs more packets – trigger retry in background
    - All replies on host in range: Compile results into format assumed by nmap/xprobe, pass struct to their evaluation routine. Add results to another table.

Layer 4: TCP Spoofing

- TCP Spoofing
  - Not normally possible – every packet sent (except the SYN) contains a token from a window of previously received packets
  - Some tools exist to blindly spoof – or measure ability for blind spoofing of – TCP ISN's (the token in question).
    - Phentropy + OpenQViz
  - Since another node cannot acquire the token, it cannot impersonate the server
    - But what if the server *sent* the impersonator the token?

Bandwidth Brokering

- *Is it possible for a single host to do load balancing across nearly arbitrary network boundries, 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**
  - **Global Load Distribution**
    - **Anyone who can spoof themselves as the redirector can serve clients**
  - **Central Traffic Monitoring**
    - **Each forwarded ACK contains, in its ACK#, the number of bytes sent by the anonymous server to the client.** ***As the redirector forwards those ACKs, it's able to monitor and measure the quality of the link.***
  - **Session migration:**
    - **Since the client always believes they're receiving packets from the redirector, the redirector can (with a little bit of magic to synchronize SEQ#'s) simply start forwarding ACKs to a less overloaded server.**
  - **Works for streams – MP3 Radio Stations, Web Servers, etc.**
  - **Much less bandwidth to move empty ACKs than payloads**

Bandwidth Brokering HOWTO

- Redirector
  - Upon receiving packet from client to redirected IP/Port, change IP Destination to redirector based on rule, recalc checksums, and send packet out appropriate interface
    - Stateless rules: TCP Source Port, IP Source (using geo-coding), etc.
    - Stateful rules: Who’s moving the least data, who’s dropped the fewest packets, etc.
- Anonymous Server
  - Before sending packet with the source port of an anonymous service, change the source IP to that of the redirector, recalc checksums, and send packet.
- Client
  - Do nothing, except notice the TTL bounce around as your stream comes from different sources.

Layer 5: SSL vs. IDS

- 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!

Layer 5: SSL vs. IDS

- 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.
- Ouch.

Layer 5: SSL vs. IDS

- Certificate Transfer
  - IDS gets a copy of the cert
  - Violates 1st Law of Private Keys: Thou Shalt Not Transport Thy Private Key
  - 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

Layer 5: SSL vs. IDS

- 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

Layer 5: SSL vs. IDS

- **Master Secret serves six keys:**
  - **Two handle encryption from client to server and vice versa**
  - **Two handle authentication from client to server and vice versa**
  - **Two handle initialization vectors from client to server and vice versa**
    - **3DES-CBC**
  - **These keys are** completely independent
- **Selective Centralized Monitoring of SSL-Encrypted Traffic**
  - **Since we have independent keys for independent traffic, we can transfer just the encryption key from the client to the server to the IDS -- it'll pick up the traffic from the insecure side of the network, without being able to intercept presumably secure (or at least mandatorily unshareable) content.**
  - **IDS doesn't need to do RSA -- just pick up keys on an encrypted channel**
  - **Key transfer occurs before data transfer -- can disauthorize traffic**
  - **IV** may **introduce interesting capability -- read traffic *after* highly sensitive exchange (authentication credentials)**
    - **UNVERIFIED**
  - **Works with Bandwidth Brokering!**

Layer 5: SSL vs. IDS

- **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: Not as cool**

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**
    - **Putty, OpenSSH, etc.**
- **Demo**