

SCTPscan - Finding entry points to SS7 Networks & Telecommunication Backbones

Philippe Langlois
Telecom Security Task Force
Philippe.Langlois@tstf.net

Agenda

- History of telecommunications security
- Review of digital telephony concepts
- Discovering the backbone
- SIGTRAN: From SS7 to TCP/IP
- Attacking SIGTRAN
- Q&A

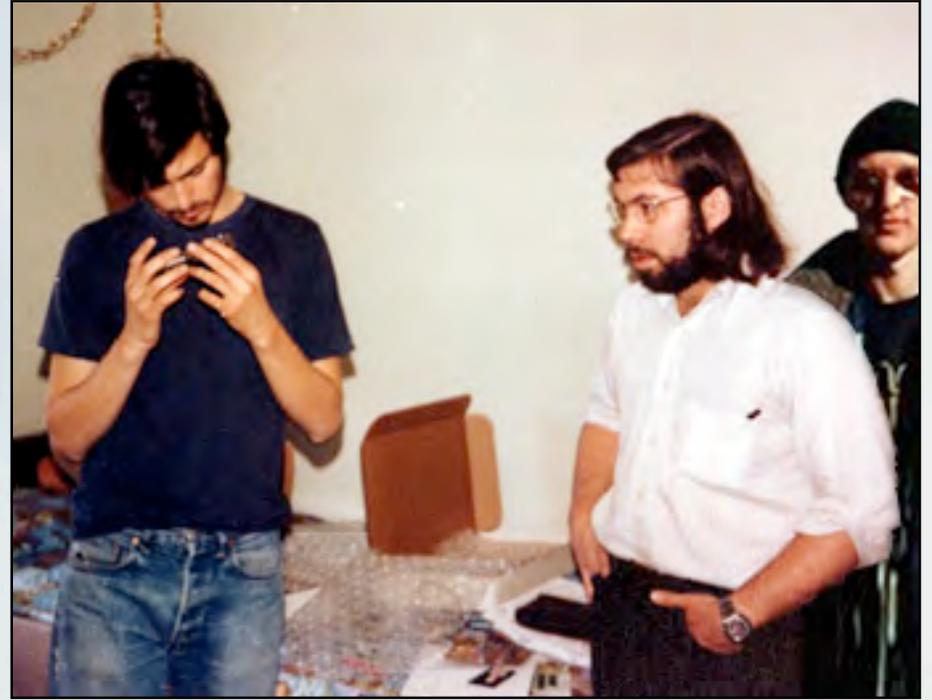
The origins

- *Phreaking* is a slang term for the action of making a telephone system do something that it normally should not allow.
- Telecommunications security problems started in the 1960's when the hackers of the time started to discover ways to abuse the telephone company.

But... what is it?

- Discovery and exploration of features of telecommunications systems
- Controlling Network Elements (NE) in a way that was not planned by its designers
- Abusing weaknesses of protocols, systems and applications in telephone networks

The Blue Box



Steve Jobs and Steve Wozniak in 1975 with a bluebox

- CCITT#5 in-band signalling sends control messages over the speech channel, allowing trunks to be controlled
- Seize trunk (2600) / KP1 or KP2 / destination / ST
- Started in mid-60's, became popular after Esquire 1971
- Sounds produced by whistles, electronics dialers, computer programs, recorded tones

The end of the blueboxing era



- Telcos installed filters, changed frequencies, analyzed patterns, sued fraudsters
- The new SS7 digital signalling protocol is **out-of-band** and defeats blueboxing
- In Europe, boxing was common until the early nineties and kept on until 1997-1998
- In Asia, boxing can still be done on some countries.

Past & current threats on the telecom backbone

- Fraud
 - Blue Box
 - Internal Fraud
- Reliability
 - US: 911, Europe: 112
 - How much lost revenue is one minute of downtime?



21st century telecom attacks

- SIP account hacking
 - Remember "Calling Cards" fraud?
- VoIP GW hacking
 - Remember "PBX hacking"?
- Signalling hacking directly on SS7 –
SIGTRAN level
 - Back at the good old BlueBox?
 - Not nearly but, the closest so far...

Example of SS7 Attacks

- Theft of service, interception of calling cards numbers, privacy concerns
- Introduce harmful packets into the national and global SS7 networks
- Get control of call processing, get control of accounting reports
- Obtain credit card numbers, non-listed numbers, etc.
- Messages can be read, altered, injected or deleted
- Denial of service, security triplet replay to compromise authentication
- Annoyance calls, free calls, disruption of emergency services
- Capture of gateways, rerouting of call traffic
- Disruption of service to large parts of the network
- Call processing exposed through Signaling Control Protocol
- Announcement service exposed to IP through RTP
- Disclosure of bearer channel traffic

Agenda

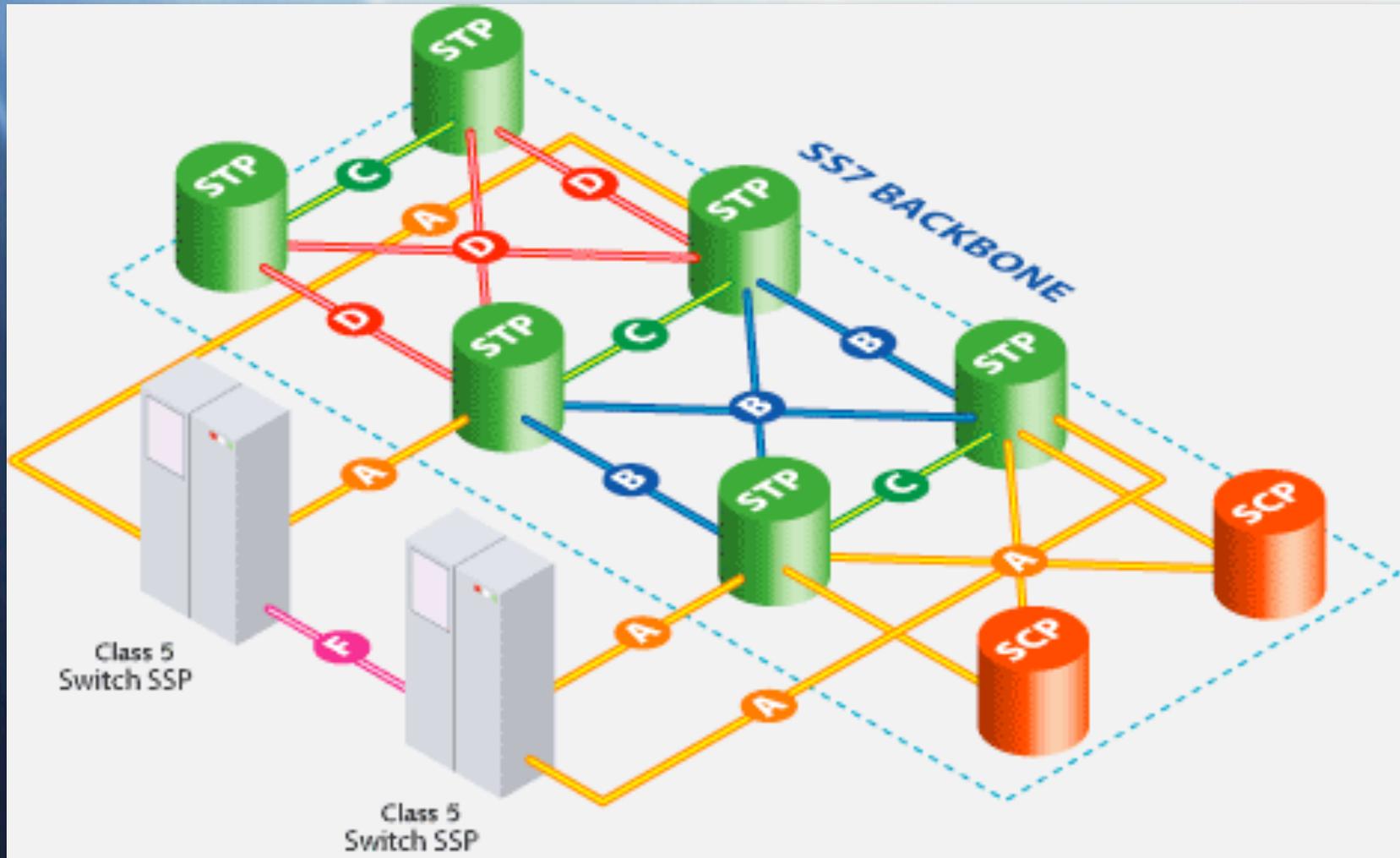
- History of telecommunications security
- **Review of digital telephony concepts**
- Discovering the backbone
- SIGTRAN: From SS7 to TCP/IP
- Attacking SIGTRAN
- Q&A
- Lab - BYOL

Telephony 101 (recap)



- Fixed line (PSTN): analog, digital (ISDN)
- Mobile: analog (AMPS, NMT), digital (GSM, CDMA, 3G), private (PMR, Military)
- Telephony switches speak out-of-band SS7 signalling
- Speech and data convergence is increasing
- Services are growing (SMS, MMS, packet data, WLAN integration, etc.)
- VoIP and related technologies (SIP, IMS, PacketCable)

Telecom Backbones Organization



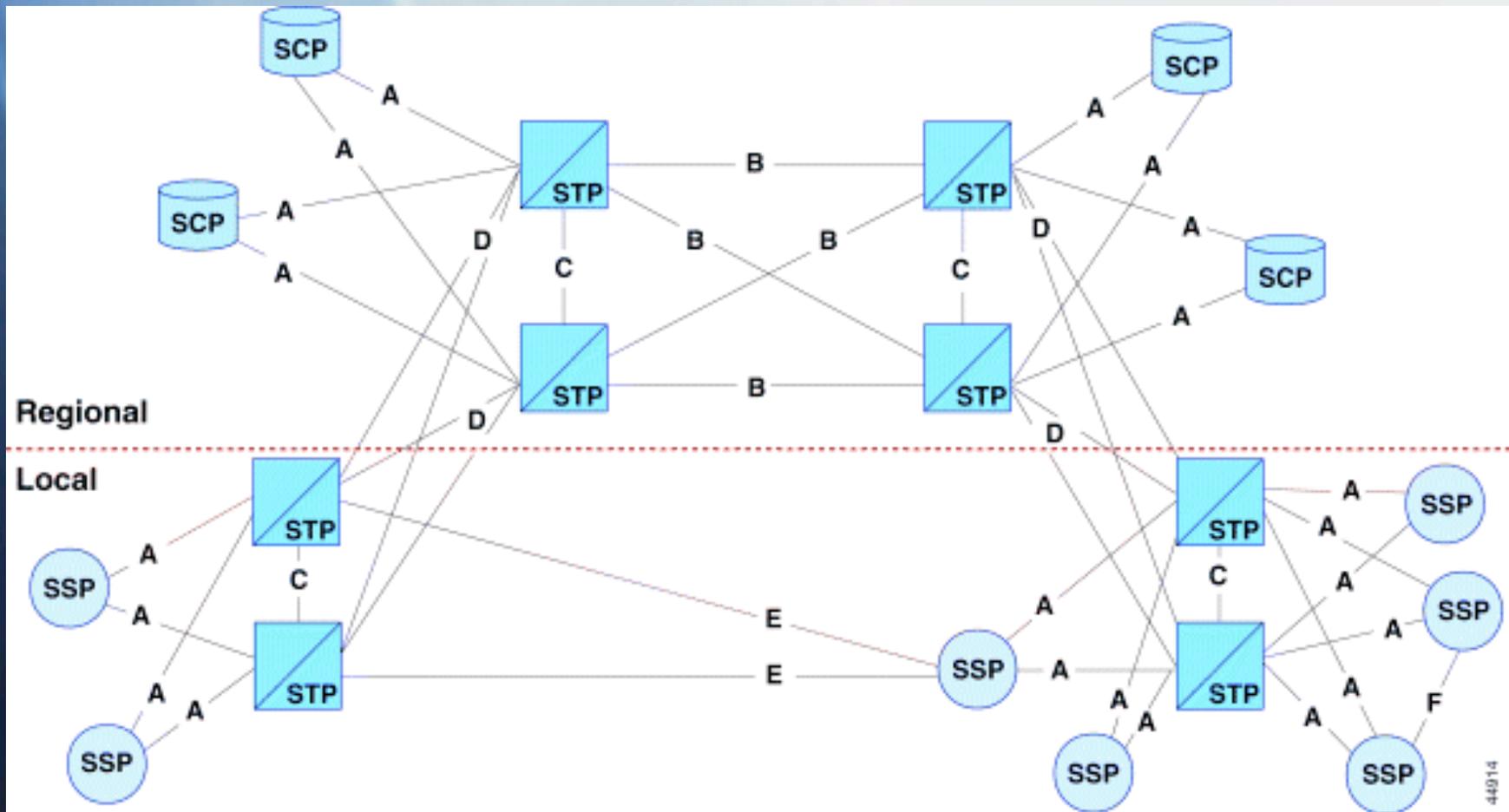
SS7: The walled garden

- From a customer perspective
 - Wikipedia: “Walled Garden – Mobile Network Operators (MNOs). At the start of 2007, probably the best example. MNOs manage closed networks - very hard to enter the garden, or leave the garden, especially as it pertains to Internet, web services, web applications. Fearful of losing customer and brand control, the MNOs opt to guard the garden as much as possible.”
- But also from a technology perspective
 - OSI : Open Protocol - Proprietary Stacks
 - Closed OSI network, IP management network

Agenda

- History of telecommunications security
- Review of digital telephony concepts
- **Discovering the backbone**
- SIGTRAN: From SS7 to TCP/IP
- Attacking SIGTRAN
- Q&A
- Lab - BYOL

SS7 Network: Regional & Local

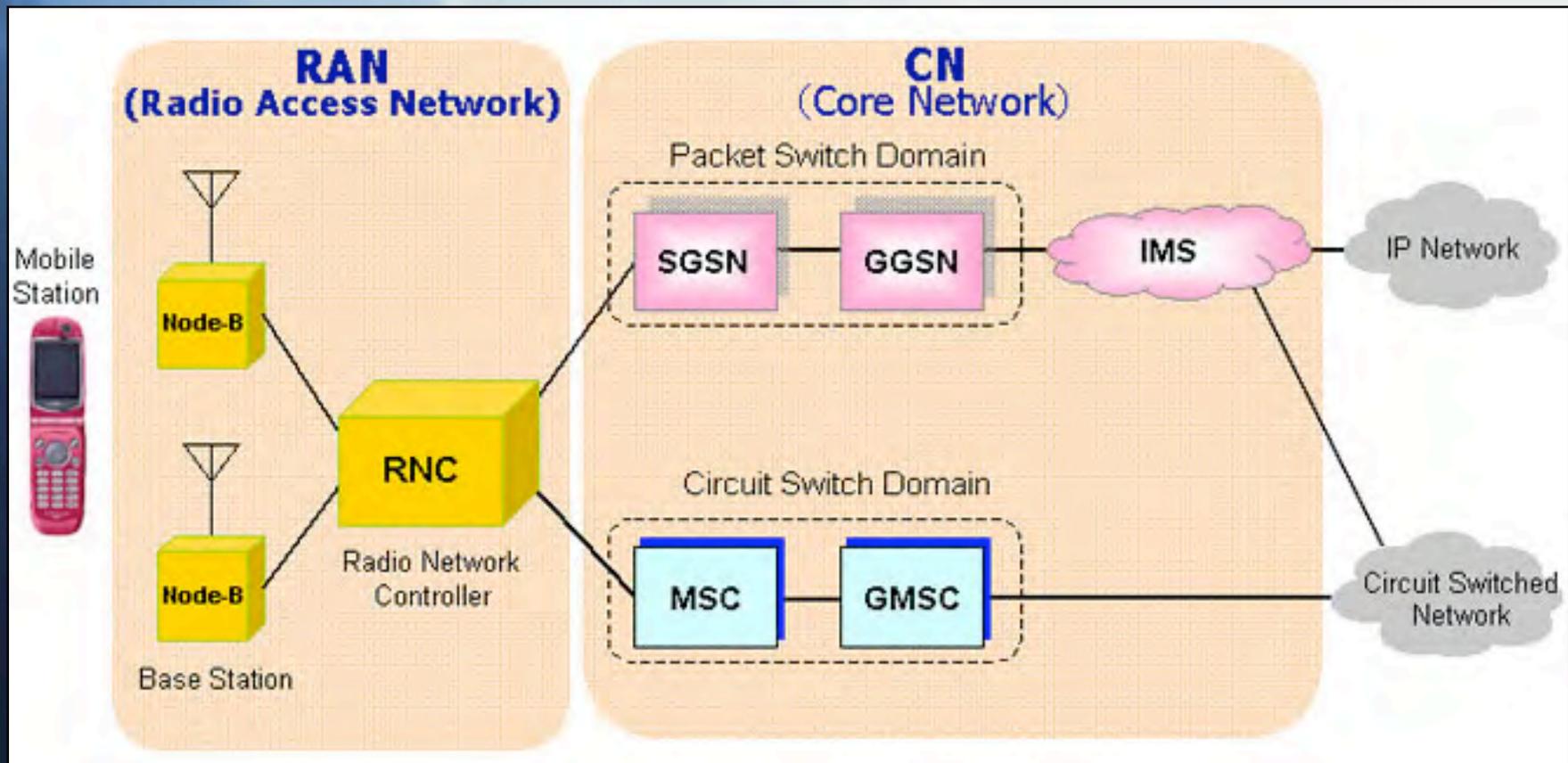




Opening up

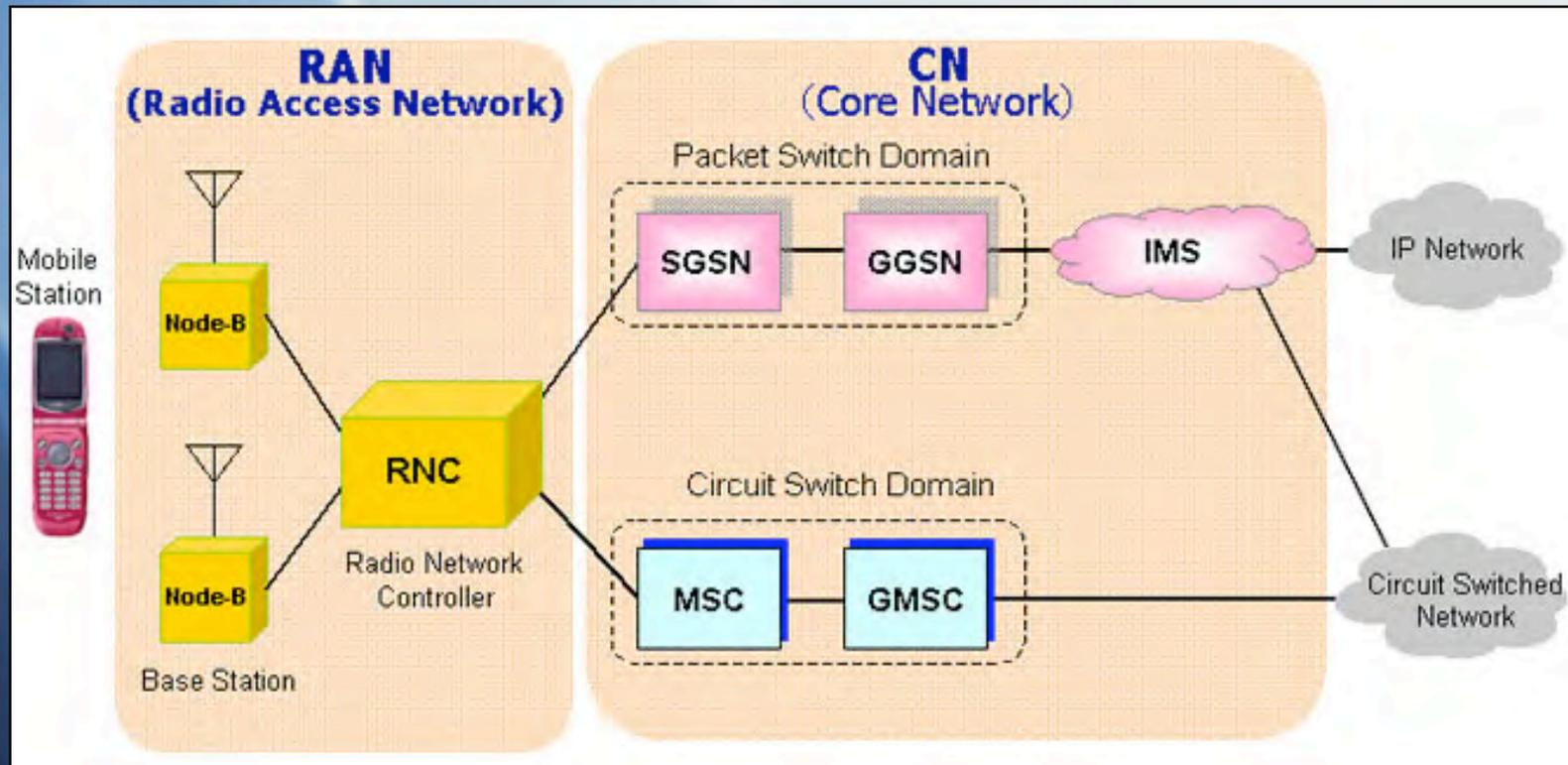
- Deregulation
 - Europe / US: CLEC vs ILEC
- New services and new business partners
 - Premium numbers, SMS providers, ...
- Push toward an “All IP” infrastructure
 - Management network first...
 - Cost
 - SIGTRAN (SS7 over IP)

Telco Backbone Global Picture



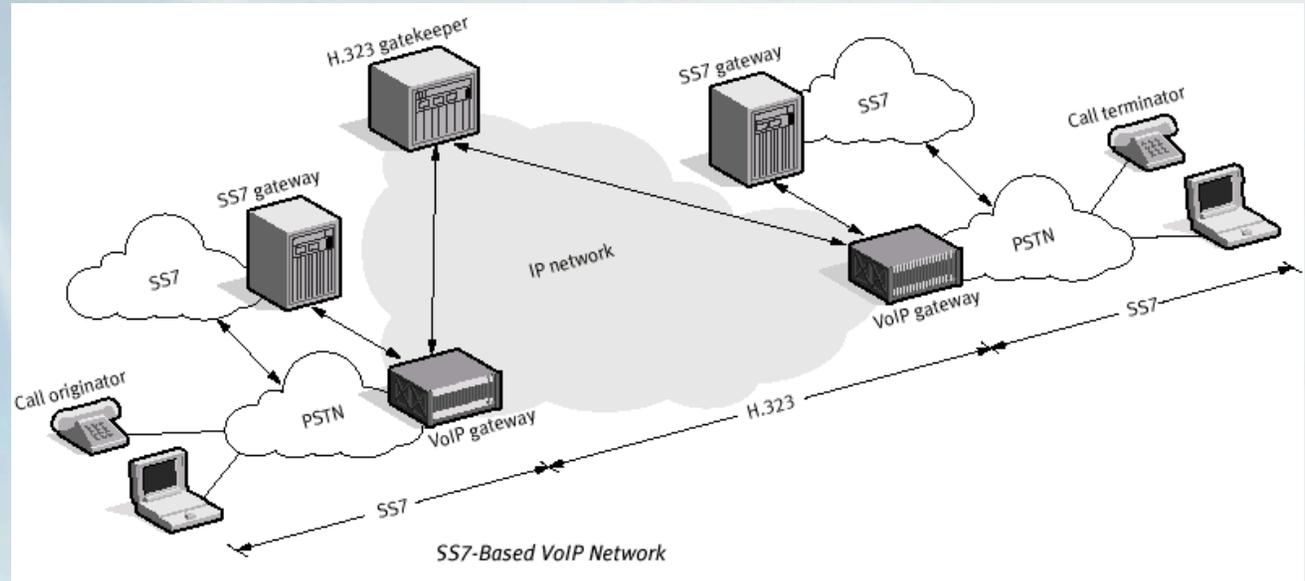
IMS = SS7 SIGTRAN + IP-based Advanced Services 17

VoIP and SIGTRAN



- SS7 & SIGTRAN
 - Core
 - Formerly, the walled garden
- VoIP
 - Edge
 - Hard to make it reliable (QoS, SBCs)

SS7 and IP



- There is also exponential growth in the use of interconnection between the telecommunication networks and the Internet, for example with VoIP protocols (e.g. SIP, SCTP, M3UA, etc.)
- The IT community now has many protocol converters for conversion of SS7 data to IP, primarily for the transportation of voice and data over the IP networks. In addition new services such as those based on IN will lead to a growing use of the SS7 network for general data transfers.
- There have been a number of incidents from accidental action on SS7, which have damaged a network. To date, there have been very few deliberate actions. Far from VoIP here.



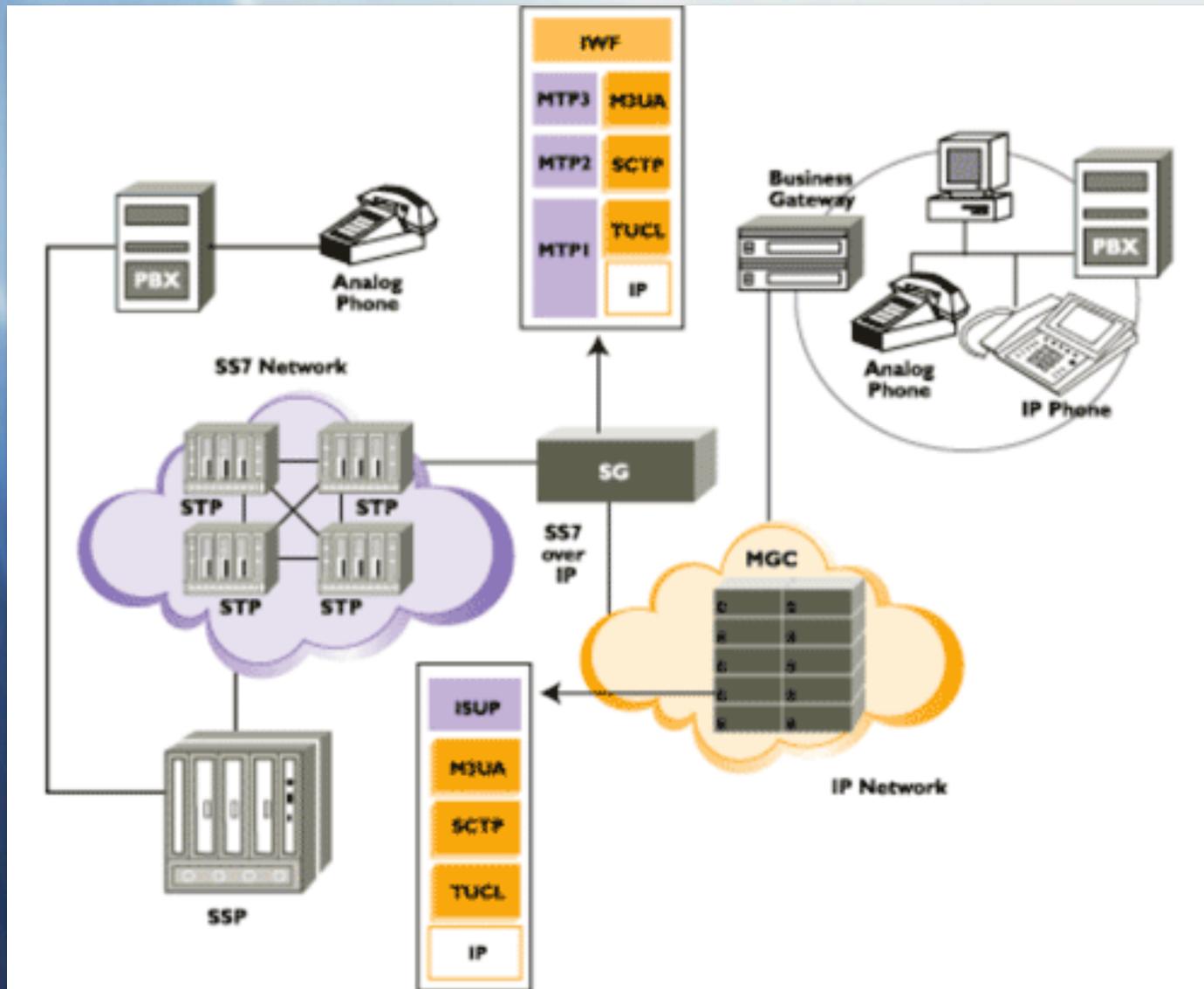
A shock of culture: SS7 vs. IP

- Different set of people
 - IT vs Telecom Operations
- New Open Technology
 - Open stack
 - Open software
 - Interconnected Networks
- Habits and induced security problems
 - Eiffel, QA, Acceptance tests, ...

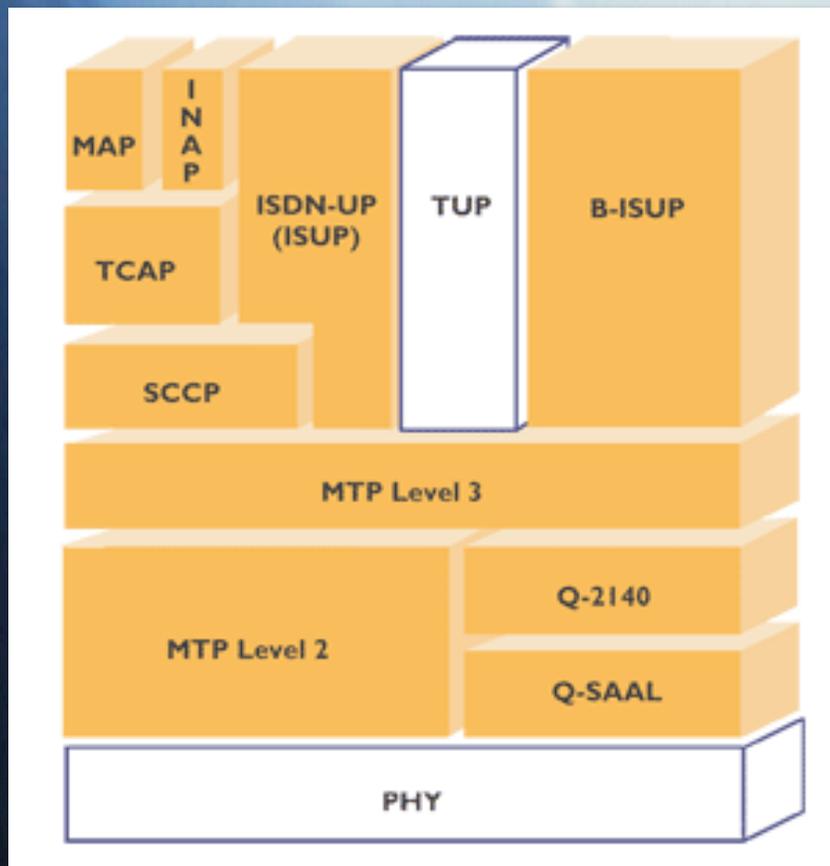
Agenda

- History of telecommunications security
- Review of digital telephony concepts
- Discovering the backbone
- **SIGTRAN: From SS7 to TCP/IP**
- Attacking SIGTRAN
- Q&A
- Lab - BYOL

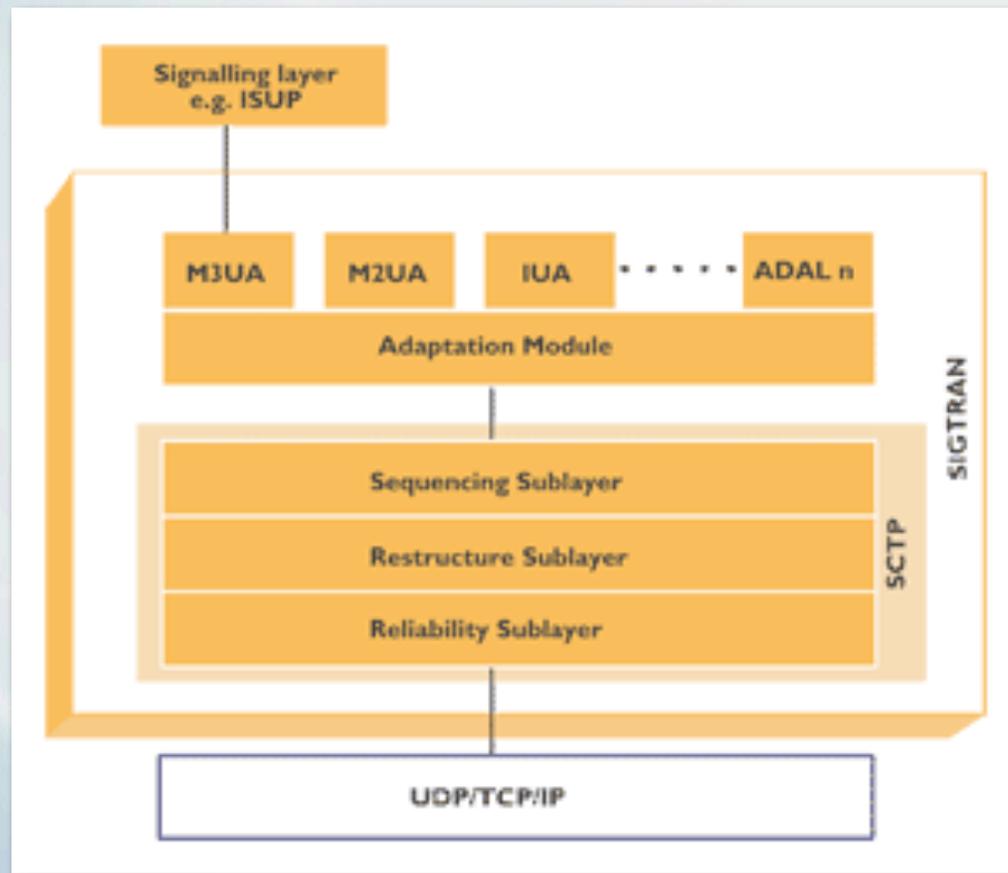
SIGTRAN in the VoIP big picture



SCTP as SIGTRAN Foundation

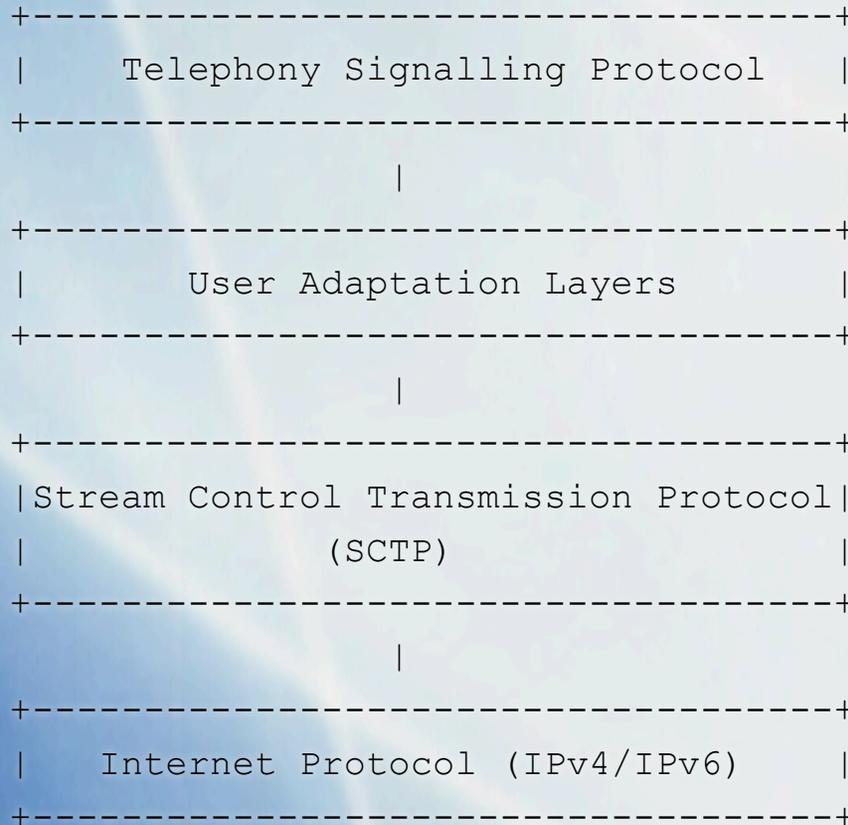


SS7



SIGTRAN

SCTP over IP



From RFC 4166

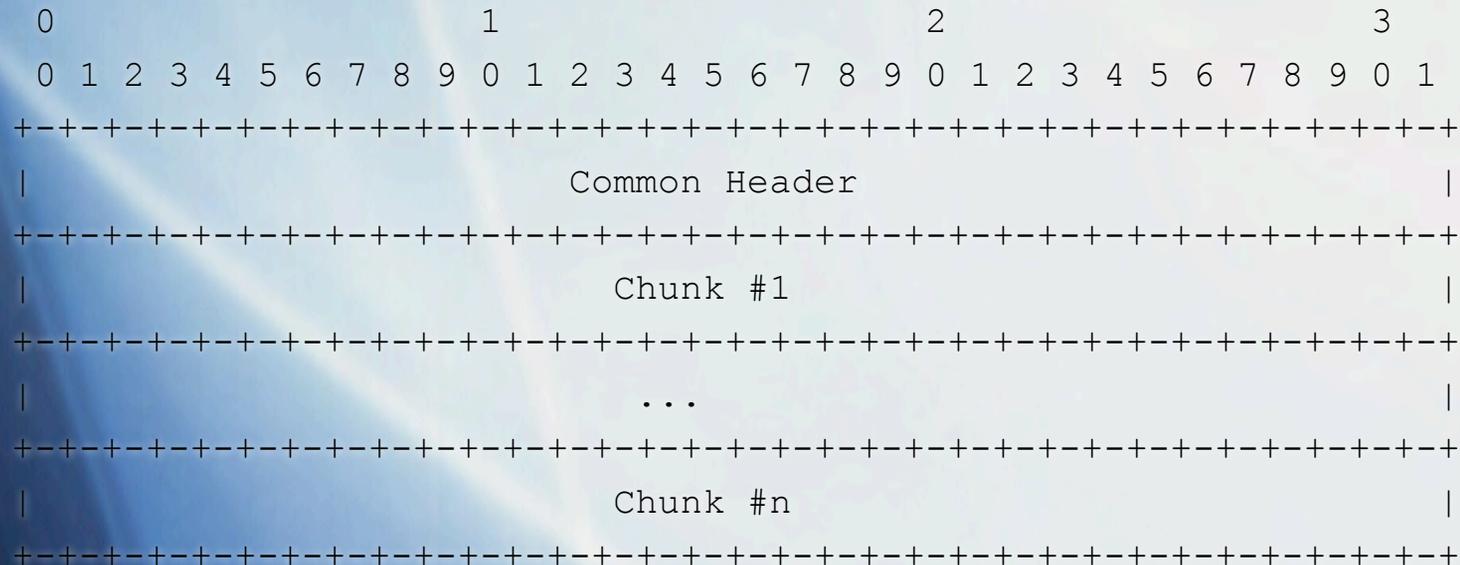


SCTP Specs & Advantages

- RFC2960
 - SCTP: Stream Control Transmission Protocol
- Advantages
 - Multi-homing
 - DoS resilient (4-way handshake, cookie)
 - Multi-stream
 - Reliable datagram mode
- Some of TCP & UDP, improved

SCTP Packets

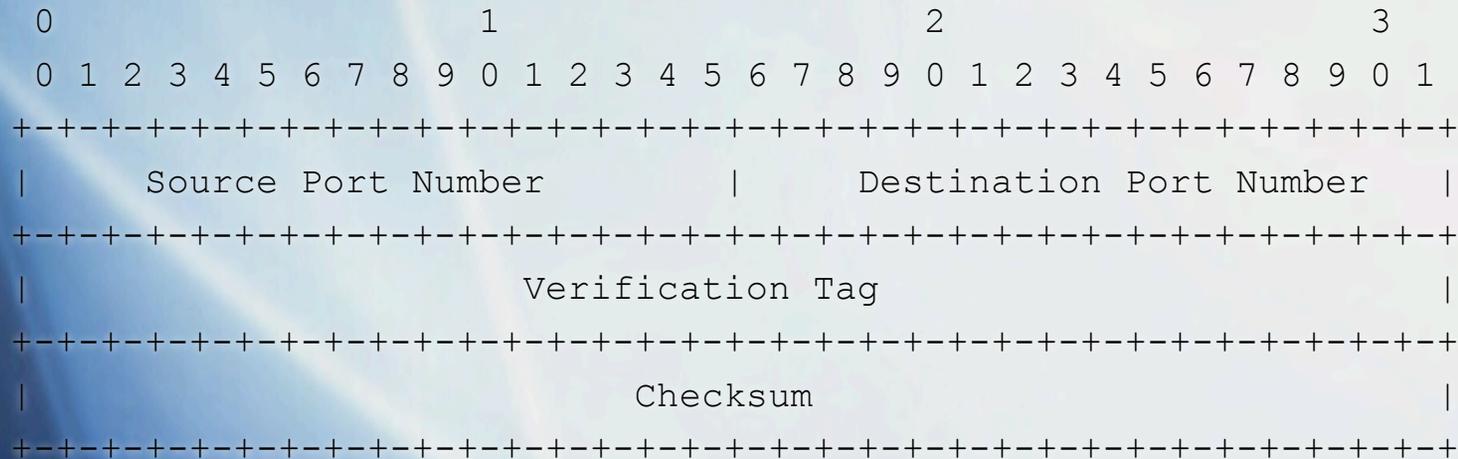
3. SCTP packet Format



- From RFC 2960

SCTP Common Header

SCTP Common Header Format



Straight from RFC2960

SCTP Chunk types

ID Value	Chunk Type
-----	-----
0	- Payload Data (DATA)
1	- Initiation (INIT)
2	- Initiation Acknowledgement (INIT ACK)
3	- Selective Acknowledgement (SACK)
4	- Heartbeat Request (HEARTBEAT)
5	- Heartbeat Acknowledgement (HEARTBEAT ACK)
6	- Abort (ABORT)
7	- Shutdown (SHUTDOWN)
8	- Shutdown Acknowledgement (SHUTDOWN ACK)
9	- Operation Error (ERROR)
10	- State Cookie (COOKIE ECHO)
11	- Cookie Acknowledgement (COOKIE ACK)
12	- Reserved for Explicit Congestion Notification Echo (ECNE)
13	- Reserved for Congestion Window Reduced (CWR)
14	- Shutdown Complete (SHUTDOWN COMPLETE)

SCTP in the wild

- Software
 - Tons of proprietary implementations
 - Open source implementations (Linux, BSD...)
- Network presence
 - Stack widespread with Linux 2.6 support
 - Scarcity on the open Internet
 - Rising in telco backbones / intranet
- Adoption by other worlds: MPI clusters, high speed transfers, ...

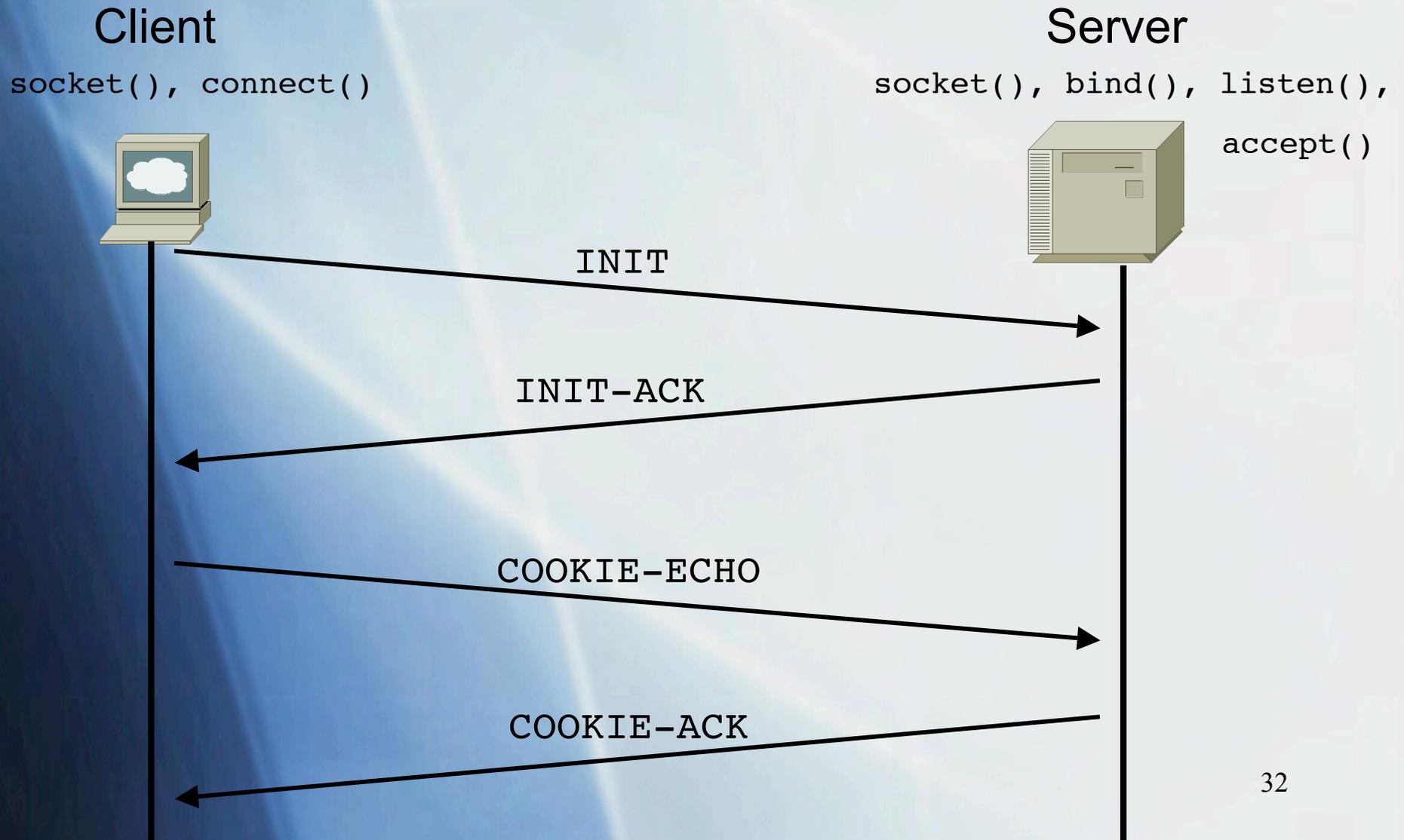
SCTP Ports & Applications

- <http://sctp.tstf.net/index.php/SCTPscan/SCTPports>
 - Common ports from IANA and RFCs
 - Augmented with open source package ports
 - Updated based on SCTPscan results
- Open to contribution
- Watch out for the application fingerprinting
 - Cf. collaborative scanning

Agenda

- History of telecommunications security
- Review of digital telephony concepts
- Discovering the backbone
- SIGTRAN: From SS7 to TCP/IP
- **Attacking SIGTRAN**
- Q&A
- Lab - BYOL

SCTP Association: 4-way handshake

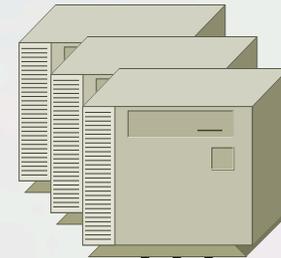


Scanning vs. Stealth Scanning

Attacker



Servers

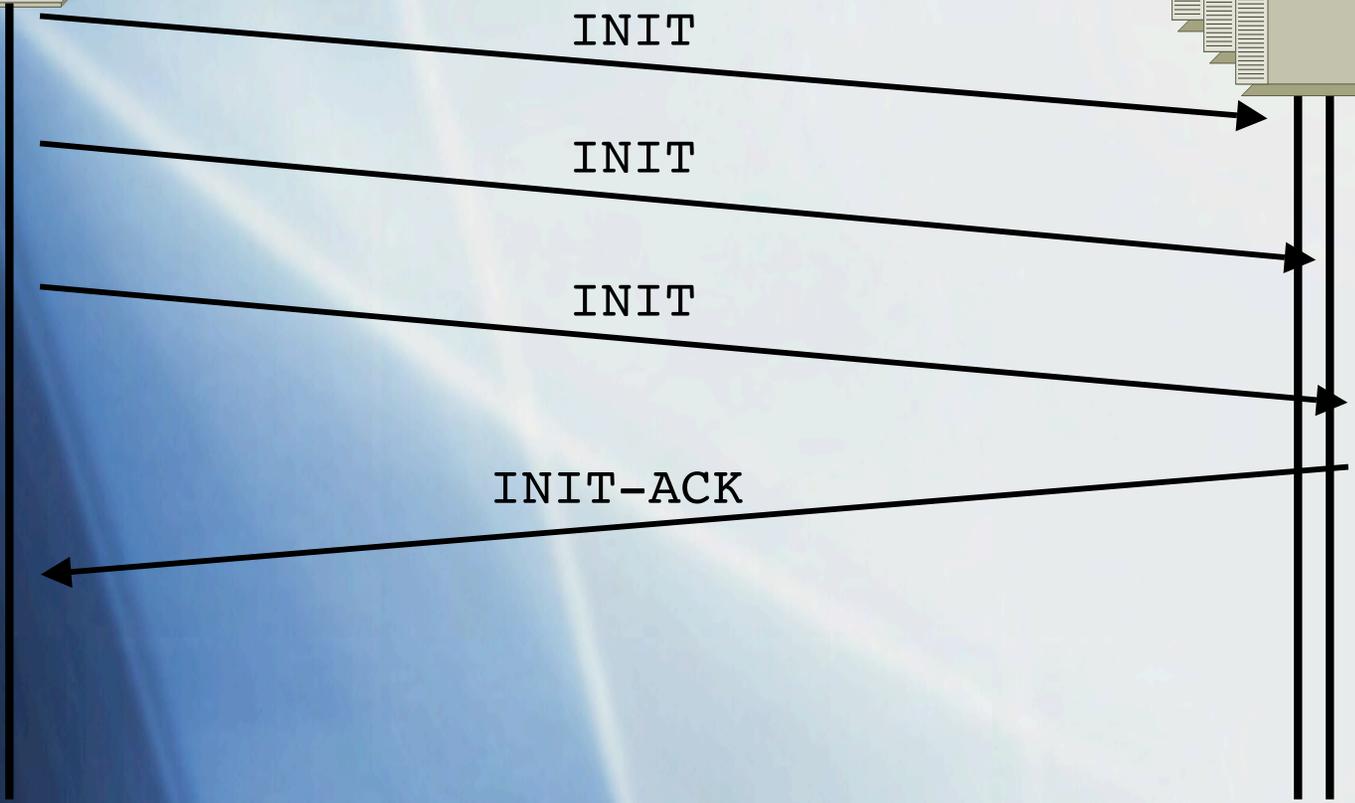


INIT

INIT

INIT

INIT-ACK



RFC & Implementation

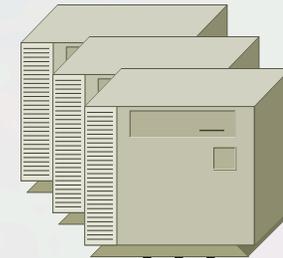
- Where implementation diverge from RFCs
- RFC says « hosts should never answer to INIT packets on non-existing ports. »
 - RFC: 0, hacker: 1.
- Syn scanning is slow when no RST
 - Same here, but thanks to over-helping implementation
 - on scanning, hacker wins

Scan against current implementation

Attacker



Servers



INIT (port=a)

ABORT

INIT (port=b)

INIT-ACK

Below the IDS radar

- How many firewall logs dropped SCTP packets?
- How many IDSes watch for SCTP socket evil content?
- Example
 - Dshield.org – Real life distributed IDS
 - Hundreds of thousands of IP scanned
 - Not detected / Not reported as scanner

INIT vs SHUTDOWN_ACK Packet Scanning

- From RFC 2960
 - “8.4 Handle "Out of the blue" Packets
 - An SCTP packet is called an "out of the blue" (OOTB) packet if it is correctly formed, i.e., passed the receiver's Adler-32 / CRC-32 check (see Section 6.8), but the receiver is not able to identify the association to which this packet belongs.
 - The receiver of an OOTB packet MUST do the following: [...]
 - 5) If the packet contains a SHUTDOWN ACK chunk, the receiver should respond to the sender of the OOTB packet with a SHUTDOWN COMPLETE.”
- New way to elicit answers even if not answering ABORTs to INITs targeted at not-opened port.

Tool Demo: SCTPscan

- Like nmap for SCTP ports (-sS)

```
root@gate:~/sctp# ./sctpscan-v11 --scan --autoportscan -r
203.151.1
Netscanning with Crc32 checksumed packet
203.151.1.4 SCTP present on port 2905
203.151.1.4 SCTP present on port 7102
203.151.1.4 SCTP present on port 7103
203.151.1.4 SCTP present on port 7105
203.151.1.4 SCTP present on port 7551
203.151.1.4 SCTP present on port 7701
203.151.1.4 SCTP present on port 7800
203.151.1.4 SCTP present on port 8001
203.151.1.4 SCTP present on port 2905
root@gate:~/sctp#
```

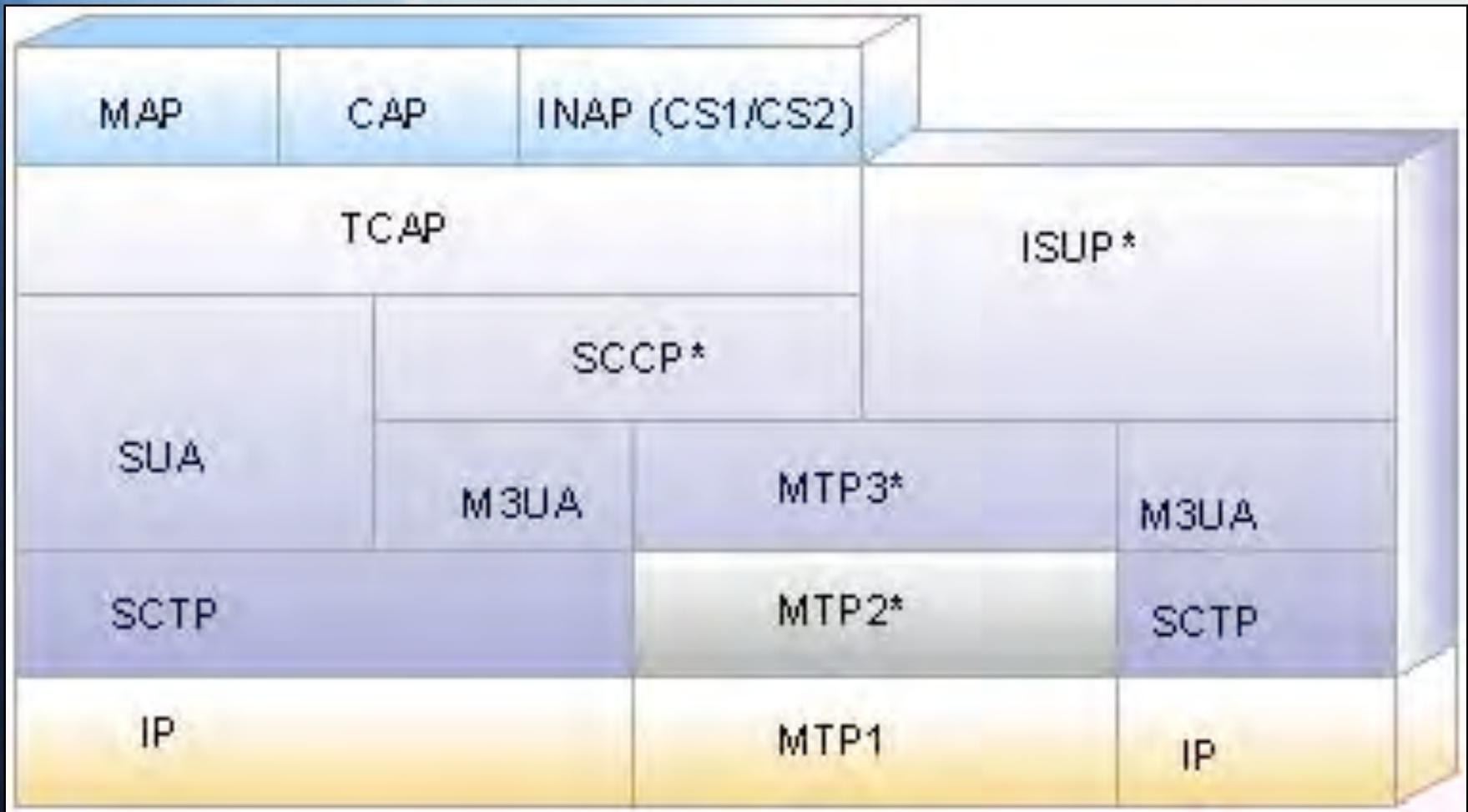
SCTP Stack Fingerprinting

- SCTP stack reliability
 - Robustness testing (stress testing)
 - QA of a few stacks
 - Fuzzing built-in SCTPscan
- SCTP stack fingerprinting
 - Discrepancies in SCTP answer packets
 - Different stack behaviours
 - Much more states than TCP=opportunities
- Cookie randomness

Scarce Presence - Distributed Collaborative Scanning

- SCTP application is rare on the internet
 - But common on modern telco backbones
- Research needs collaborative effort
 - Built-in collaborative reporting with SCTPscan.
- Going to be expanded for
 - Fuzzing results
 - Application Fingerprinting

Going up: SIGTRAN & SS7



Going up: upper layer protocols

- Key to the upper level
 - M2PA and M3UA
- Vulnerabilities
 - Telecom potential
 - Technical vulnerability
- The expert way & the automated way
 - Ethereal is our friend
 - In need of new packet captures: open call!

Demo: Ethereal Dissection of Upper Layer Protocols

- Fire up your Ethereal or Wireshark!
 - Collect your own examples
 - And contribute to the SCTPscan wiki!
- Lots of SS7 specifics in higher level protocols
 - DPC/OPC
 - BICC, ISUP, TCAP, GSM-MAP protocols
- Less and less IP-related
 - IP is only a bearer technology
 - Transport only

Fuzzing upper layer protocols

- Quick way to find vulnerabilities
 - Automated inspection
- State fuzzing vs. input fuzzing
 - Already some stack vulnerabilities in the wild
 - Only found DoS for now
- Input fuzzing for UA layers
 - SIGTRAN higher protocols
 - User Adaptation layers
 - Largest “opportunity” / work area



Shadow Chamber Roger Ballen

PHAIDON

Vulnerability evolution

- Same as with TCP
 - First, stack and “daemons” vulnerabilities
 - More and more application-level vulnerabilities
 - Custom & Application-related
- Requires more knowledge of Telecom
 - Same as with web app testing
 - “niche”: requires understanding of SS7 world
- Specifics
 - Defined Peers make attack difficult

References & Conclusion

- New realm
 - New “tunneling / VPN”
 - Same Rules
 - New fun!
- Lots of references, complex
 - RFC 2960, 4166, 4666
 - ITU (Now free)

Thanks

- Questions?
- Thank you very much!
- Special thanks to all TSTF OOB Research Team; Emmanuel Gadaix, Fyodor Yarochkin, Raoul Chiesa, Grugq, Inode, Stealth, Raptor, Job De Haas, Halvar, Michael M. Kemp, and all the community

<http://sctp.tstf.net>

Q&A

- Thanks a lot!

Agenda

- History of telecommunications security
- Review of digital telephony concepts
- Discovering the backbone
- SIGTRAN: From SS7 to TCP/IP
- Attacking SIGTRAN
- Q&A
- **Lab - BYOL**

Lab: Hands-on Agenda

- Setup
- Network Inventory
 - Scanner vs. Targets
- Scanning types
- Scanning conflicts & Kernel impact
- Analyze a SCTP exchange
 - Ethereal
- Discover a SIGTRAN architecture
- Exploring & Finding vulnerabilities

Required Skills

- Know how to compile a C program
- Know how TCP protocol works
- Know how to use tcpdump and ethereal

Hands on requirement

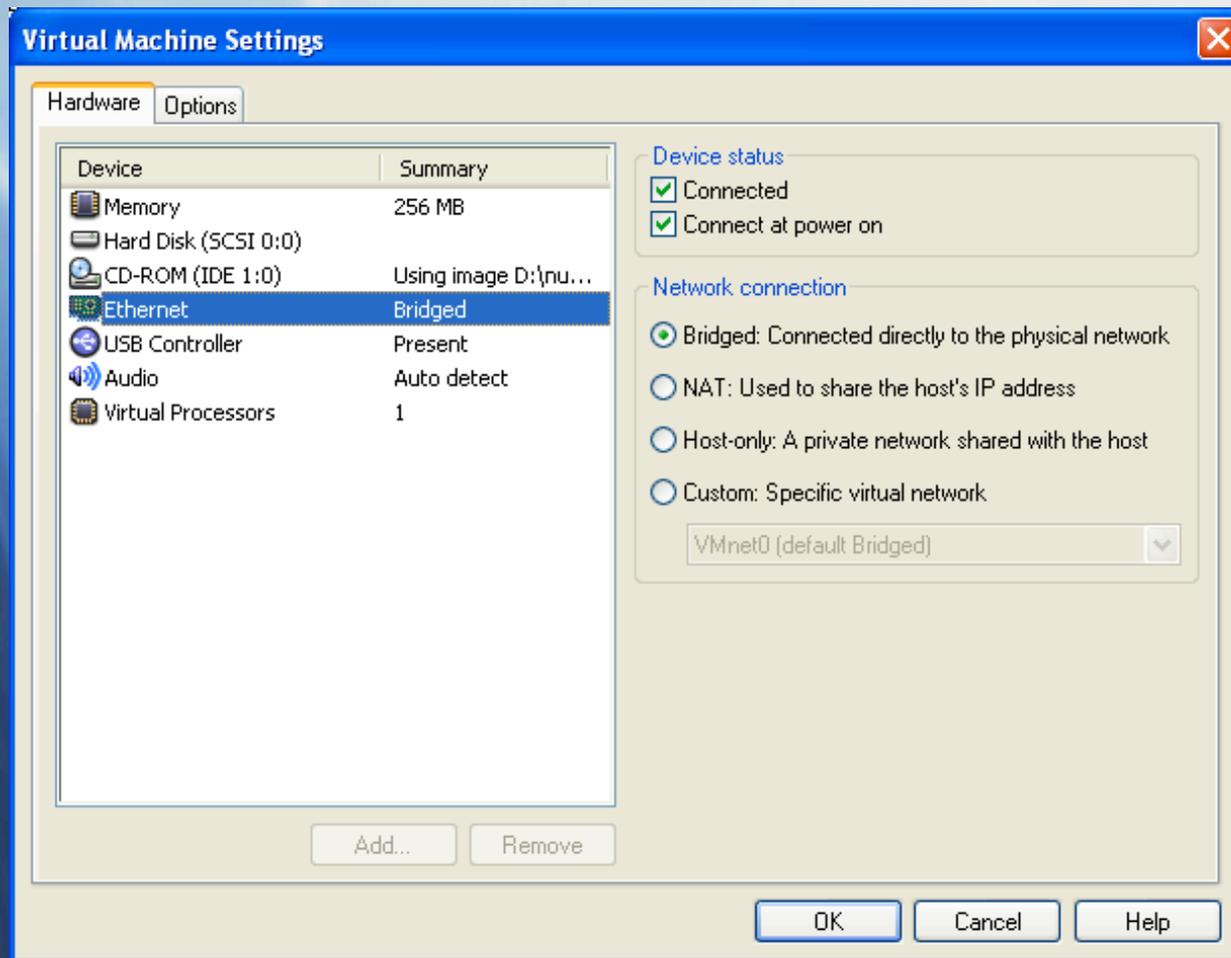
- Laptop with VMWare or bootable distribution with
 - Ubuntu with Linux 2.6 kernel (scanner and dummy server tested ok) - Download
 - nUbuntu Live CD with Linux 2.6 kernel (scanner and dummy server tested ok) - Download
 - Linux 2.4 distribution (only scanner will work, not the dummy server)
 - Solaris 10
 - Nexenta OS (GNU/Linux Solaris 10) (dummy server only) - Download instructions or distrib or VMWare image at Distrowatch
 - MacOSX (scanner and dummy server tested ok)
- Software
 - C Compiler (apt-get install gcc)
 - Glib 2.0 development library
 - Libpcap development librar
 - tcpdump (apt-get install tcpdump)
 - ethereal (apt-get install ethereal)
 - netstat

Important workshop notes!

- Your computers / VMware images must be installed before the workshop.
- OS installation or vmware image setup is not covered during the workshop.
- We have some ISOs of these Oses available for download in any case, but beware of the short time.
- <http://sctp.tstf.net/index.php/SCTPscan/Workshop>

Notes on VMware images

- Make sure to select "Bridged mode" for your ethernet connector.



Hands-on Tests

- Who scans who?
 - Scanners vs. Targets
- Scanning types
- Scanning conflicts & Kernel impact
- Analyze a SCTP exchange
 - Ethereal

Common problems

Q: I try to run the Dummy SCTP server for testing, and I get: "socket: Socket type not supported"

A: Your kernel does not support SCTP sockets.

SCTP sockets are supported by Linux Kernel 2.6 or Solaris 10.

For Linux, you may want to try as root something like: `modprobe sctp`

Then rerun: `sctpscan --dummyserver`

Note: you only need a SCTP-aware kernel to run `dummyserver`.

Scanning is ok with 2.4 linux kernels!

For Mac Os X, you may add support for SCTP in Tiger 10.4.8 by downloading:

<http://sctp.fh-muenster.de/sctp-nke.html>

Install the software package and run as root:

```
kextload /System/Library/Extensions/SCTP.kext
```

Then you can run `"sctpscan -d"` to run the dummy server.

Note that `"netstat"` won't report the use of the SCTP socket, use instead:

```
lsof -n | grep -i '132?'
```

Kernel conflicts: Linux 2.6

```
[root@nubuntu] ./sctpscan -s -r 192.168.0 -p 10000
Netscanning with Crc32 checksumed packet
192.168.0.3 SCTP present on port 10000
SCTP packet received from 192.168.0.4 port 10000 type 1 (Initiation (INIT))
End of scan: duration=5 seconds packet_sent=254 packet_rcvd=205 (SCTP=2,
    ICMP=203)
[root@nubuntu] uname -a
Linux nubuntu 2.6.17-10-386 #2 Fri Oct 13 18:41:40 UTC 2006 i686 GNU/Linux
[root@nubuntu]
```

- If after this scan, we test the dummy server SCTP daemon built in SCTPscan, we'll notice that further scans from this host will have different behavior:

```
[root@nubuntu] ./sctpscan -d
Trying to bind SCTP port
Listening on SCTP port 10000
^C
[root@nubuntu]
[root@nubuntu]
[root@nubuntu] ./sctpscan -s -r 192.168.0 -p 10000
Netscanning with Crc32 checksumed packet
192.168.0.3 SCTP present on port 10000
SCTP packet received from 192.168.0.4 port 10000 type 1 (Initiation (INIT))
SCTP packet received from 192.168.0.4 port 10000 type 6 (Abort (ABORT))
End of scan: duration=5 seconds packet_sent=254 packet_rcvd=206 (SCTP=3,
    ICMP=203)
[root@nubuntu]
```

Kernel conflicts: MacOS X

```
localhost:~/Documents/sctpscan/ root# kextload
/System/Library/Extensions/SCTP.kext
kextload: /System/Library/Extensions/SCTP.kext loaded successfully
localhost:~/Documents/sctpscan/ root# ./sctpscan -s -r 192.168.0 -p 10000
Netscanning with Crc32 checksumed packet
End of scan: duration=9 seconds packet_sent=254 packet_rcvd=3 (SCTP=0, ICMP=3)
localhost:~/Documents/sctpscan/ root# kextunload
/System/Library/Extensions/SCTP.kext
kextunload: unload kext /System/Library/Extensions/SCTP.kext succeeded
localhost:~/Documents/sctpscan/ root# ./sctpscan -s -r 192.168.0 -p 10000
Netscanning with Crc32 checksumed packet
SCTP packet received from 127.0.0.1 port 10000 type 1 (Initiation (INIT))
192.168.0.4 SCTP present on port 10000
End of scan: duration=9 seconds packet_sent=254 packet_rcvd=5 (SCTP=2, ICMP=3)
localhost:~/Documents/sctpscan/ root#
```

- You saw in this example that loading the SCTP kernel module prevents SCTPscan to receive the response packets, and thus is not capable to detect presence of a remote open port.

Thanks

- Thank you very much!
- Special thanks to Emmanuel Gadaix, Fyodor Yarochkin, Raoul Chiesa, Inode, Stealth, Raptor, Job De Haas, Michael M. Kemp, all TSTF OOB Research Team and all the community
- Contact / Questions:
 - Philippe Langlois - pl@tstf.net

Backup slides

Comparison SCTP, TCP, UDP

SCTP vs TCP vs UDP			
Services/Features	SCTP	TCP	UDP
Connection-oriented	yes	yes	no
Full duplex	yes	yes	yes
Reliable data transfer	yes	yes	no
Partial-reliable data transfer	optional	no	no
Ordered data delivery	yes	yes	no
Unordered data delivery	yes	no	yes
Flow control	yes	yes	no
Congestion control	yes	yes	no
ECN capable	yes	yes	no
Selective ACKs	yes	optional	no
Preservation of message boundaries	yes	no	yes
Path MTU discovery	yes	yes	no
Application PDU fragmentation	yes	yes	no
Application PDU bundling	yes	yes	no
Multistreaming	yes	no	no
Multihoming	yes	no	no
Protection against SYN flooding attacks	yes	no	n/a
Allows half-closed connections	no	yes	n/a
Reachability check	yes	yes	no
Pseudo-header for checksum	no (uses vtags)	yes	yes
Time wait state	for vtags	for 4-tuple	n/a

Details of an SSP / STP

