SCTPscan & SIGTRAN Research paper

By Philippe Langlois, Founder & Researcher
Telecom Security Task Force

Philippe.Langlois@tstf.net

Abstract

SCTP is an emerging protocol that is used to carry SS7 over TCP/IP, among other things. It is part of the SIGTRAN protocol family, for SIGnalling TRANsport. We will study what vulnerabilities and entry point may exist within the context of SCTP and the migration from SS7 to an TCP/IP environment using SIGTRAN protocol family. We will also inspect the emergence of SIGTRAN use in internal networks as well as on the Internet. We will then analyze what kind of security dynamics we are facing, in term of vulnerability kinds that you find on SS7 networks, attacks seen in the wild and the emergency response that you get in telecom backbone environment.

1 PRESENTATION ... 3
1.1 HISTORY OF TELECOMMUNICATIONS SECURITY ... 3
1.2 REVIEW OF DIGITAL TELEPHONY CONCEPTS ... 5
1.3 SS7 NETWORKS ... 7
1.3.1 SS7 Terminology ... 7
1.3.2 SS7 Network Description ... 10
1.3.3 SS7 Addressing ... 10
1.3.4 Business & People ... 11
1.3.5 Security consideration ... 12
1.4 SIGTRAN & SCTP ... 13
1.4.1 SIGTRAN Protocol Family ... 13
1.4.2 VoIP and SIGTRAN ... 14
1.4.3 SCTP as SIGTRAN Foundation ... 15
1.4.4 SCTP Software ... 15
1.4.5 SCTP Ports & Applications ... 16
1.5 DIFFERENT DEVELOPING DYNAMICS ... 18
2 SCTPSCAN IN REAL LIFE ... 18
2.1 TEST & PENTEST ENVIRONMENT ... 18
2.1.1 Supported environments ... 18
2.1.2 Virtual Machine test setup ... 19
2.1.3 Requirements when using a NATed address ... 20
2.2 USAGE ... 20
2.2.1 Scanning ... 22
2.2.2 Automatic portscanning ... 23
2.2.3 Dummy server mode ... 24
2.3 COMPILING AND TROUBLESHOOTING ... 25
2.3.1 Compiling ... 25
2.3.2 Common problems ... 26
2.3.3 Kernel conflicts with Linux 2.6 ... 27
2.3.4 Kernel conflicts on MacOS X ... 27
3 TECHNOLOGY DISCUSSION ... 29
3.1 PACKETS & CHUNKS ... 29
3.1.1 SCTP Packets ... 29
3.1.2 Header ... 29
3.1.3 SCTP Chunks ... 30
3.2 INIT VS SHUTDOWN_ACK PACKET SCANNING ... 30
3.3 CHECKSUMS ... 31
3.4 COLLABORATIVE SCANNING ... 31
3.5 SCTPSCAN IMPROVEMENTS ... 31
4 RESULTS ... 32
5 CONCLUSIONS ... 32
6 REFERENCES ... 33

1 Presentation

1.1 History of telecommunications security

Telecommunication security was first brought to general public attention due to "Phreaking". 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. Mostly, this freaking was due to "blue boxing".

The Blue Box

The Blue Box was a tool to simulate the behavior of a telephone switch talking to another telephone switch.
At that time, CCITT#5 [1] in-band signaling system was used. The C5 system used DTMF-like tones to indicate call-status and send call orders from one switch to another.

These control messages were sent right over the speech channel, allowing trunks to be controlled by other switches, or by normal user sending these very tones, and that was the source of the Blue Boxing problems.

A blue boxing session or call would typically look like that:

1. Call a toll free number (1 800 xxx xxxx in the US, 05 xx xx xx xx in France),
2. Send Seize trunk (2600) so that the remote switch thought the call ended and some switch is talking to it,
3. Send KP1 or KP2 to say "I'm the switch and I'm starting a new call, here is the destination number".
4. Send destination number in C5 tones (MF tones),
5. Send ST tone to actually start the call.

The first switch would still believe that you'd be connected to a toll free number whereas you'd be actually calling now another number.

Blue Box impact

But Blue box enabled to do much more:

- Control telephone system as operators would do
- Start emergency calls

- Do manual routing with routing codes (think “Source Routing” in the TCP/IP world.
- And even interrupt some calls currently being connected or tap into them!

Blue boxing started in mid-60’s, became popular after an article in Esquire magazine in 1971.

Actually, some famous hacker called “Captain crunch” found this accidentally by whistling a toy whistle found within “captain crunch” cereal boxes! He blew the whistle while calling a friend, and the remote switch responded to this by hanging up the phone line! Call disconnected! That was the source of telecommunication security problems.

The blue box sounds could be easily produced by whistles, electronics dialers, computer programs, recorded tones.

What these tones did was to control network elements of the telecommunication network posing as other telecommunication equipment.

Actually, this problem has happened on other network equipments in other realms like X25: some PAD would not filter control-characters that were used internally to separate packets. You could send binary data through the serial link to the PAD, and the PAD would convert that to X25 individual packets that would be inserted as RAW packets. As such, that was a not-well-known vulnerability of PADs, and this was due to the same class of vulnerabilities.

Telecommunication attack has been traditionally done through different steps:

1. Discovery and exploration of features of telecommunications systems (think: telephone numbering plan, war dialing, etc…)
2. Taking control Network Elements (NE) in a way that was not planned by its designers,
3. Abusing weaknesses of protocols, systems and applications in telephone networks to defraud telcos of manipulate actual calls

End of the blue boxing era

The end of the blueboxing era was a period that lasted for roughly 15 years. At first, Telcos installed filters, changed frequencies, analyzed patterns, sued fraudsters, but it proved not very efficient.

The new SS7 [2] digital signaling protocol used out-of-band signaling and defeats completely blueboxing by design. It was named Signaling System 7 because it took two iteration of design and specification after CCITT signaling system 5.

In Europe, blue boxing was common until the early nineties and kept on until 1997-1998. In Asia, boxing can still be done on some countries. You can actually still hear sometime the short “beeps” when calling developing countries with your phone. This means that the termination of the call is done through C5 system and can sometime be blue boxed.

Internal problems and reliability

But past and current threats on the telecom backbone come from other source than fraud and Blue Box: Internal fraud and reliability are two big issues in telecom environment.

Reliability deals with making sure that emergency service such as 911 in the US or 112 in Europe are available. Also, it deals with the amount of lost revenue that one minute of downtime can produce.

Current status

Now, in the 21st century, the face of telecom attacks has changed:

- VoIP account hacking (Remember ”Calling Cards” fraud?) where attacker can bruteforce accounts through Web billing account access and then abuse the SIP account.
- VoIP GW hacking by organized groups who massively defraud the victim by using their gateway to sell traffic (billable minutes) to a list of call-shops who don’t even know their provider is using illegally obtained service. (Remember ”PBX hacking”?)
- Signaling hacking directly on SS7 – SIGTRAN level, basically what we’re analyzing here. It’s already done with internal fraud where some employees create ghosts numbering plans, provision calling cards with massive credits with SS7 messages. Could we be back at the good old BlueBox? Not nearly but, the closest so far…

Before diving deep into the attack of telecommunication networks with SIGTRAN, let’s mention various telecommunication concepts and technologies.

1.2 Review of digital telephony concepts

Core vs Edge

Voice access networks that use SS7 as their signaling network are of various forms:

- Fixed line
  - Fixed line aka POTS – Plain Old Telephone Service, that is analog copper-based lines using standard telephone with either rotary or DTMF-dials
  - Digital telephone lines (ISDN)
- Mobile
  - Analog (AMPS, NMT),
  - digital (GSM, DCS, CDMA, 3G),
  - private (PMR, Military)
- IP-based telephony
  - SIP-based VoIP
  - H323-based VoIP
- Broadband
  - Voice over DSL technologies VoDSL
  - PacketCable and other proprietary systems

Also, some non-voice related technologies do use SS7 network:

- SMS and MMS
- GPRS, EDGE, EVDO packet data,
- UMA (Unified Mobile Access) also called "WLAN integration"
- NGN (Next Generation Network or Next Generation Nightmare [6]).

Open vs. Closed philosophy

All these technologies are mostly located at the edge of the SS7 cloud. This SS7 cloud is a nicely guarded walled garden.

Quote from 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."
```

This philosophy is true also from a technology perspective where access to SS7 network is closely watched, but inside, depending on the telco, everything can be very insecure.

SS7 networks use OSI protocol stack. It’s an “Open Protocol” but actually it is only proprietary OSI stacks that are used: Sun, HP, Tekelec, …

IT & Management network
That doesn’t prevent telco from also using IP networks in their SS7 network. It’s mostly used for management of the network elements, not at all for the signaling transport.

Telecom people actually often disregarded TCP/IP as a proper signaling network.

1.3 SS7 Networks

First, one must say that SS7 world is full of a terminology which is quite different from TCP/IP world. We’re going to describe SS7 networks in an understandable way for people familiar with TCP/IP technology.

1.3.1 SS7 Terminology

Home Location Register (HLR)
The main database of permanent subscriber information for a mobile network. The HLR is an integral component of code division multiple access (CDMA), time division multiple access (TDMA), and Global System for Mobile communications (GMS) networks.

Integrated Services Digital Network User Part (ISUP)
The functional part of the Signaling System No. 7 (SS7) protocol, (i.e., the part that specifies the interexchange signaling procedures for the setup and tear down of trunk calls between networks).

Intelligent Network (IN)
A telephone network architecture that separates service logic from switching equipment, allowing new services to be added without redesigning switches to support those new services.

Link Sets

Signaling data links are grouped into link sets. All links in a link set must connect to a single point code. Up to 16 links can be assigned to a single link set.

M2PA
Allows communication between SS7 systems over IP rather than T-1 or E-1 TDM links. An M2PA link may be used in place of an MTP2 link, removing the need for dedicated and expensive SS7 hardware.

M3UA
SIGTRAN M3UA signaling gateways run the lower levels of the SS7 stack (MTP3 and MTP2), and application servers the higher levels (ISUP, SCCP). M3UA is used for carrying MTP3 traffic between the signaling gateway and application post. It allows reliable, resilient and flexible system architectures to be developed while offering interoperability between vendors.

Message Transfer Part (MTP)
Provides physical, data link, and network layer functions. MTP transports information from the upper layers (including the user parts and SS7 applications) across the SS7 network and includes the network management procedures to reconfigure message routing in response to network failures. Refers to level 1 through 3 in the SS7 protocol stack (MTP1-MTP3).

Mobile Application Part (MAP)
SS7 standards that address the registration of roamers and the intersystem hand-off procedure in wireless mobile telephony.

Mobile Switching Center (MSC)
A mobile switching center is responsible for connecting calls together by switching the digital voice data packets from one network path to another — a process usually called 'call routing'. MSCs also provide additional information to support mobile service subscribers, including user registration, authentication, and location updating.

Point Code
A number that uniquely identifies a node in an SS7 network. Used to direct messages to the appropriate destination. Each switch, Service Control Point (SCP), Signal Transfer Point (STP), and IP (Intelligent Peripheral) has a unique point code in the SS7 network.

Service Control Points (SCP)

Service Control Points contain centralized network databases for providing enhanced services. The SCP accepts queries from the Service Switching Point (SSP) and returns the requested information to the requestor.

Service Switching Points (SSP)
Service Switching Points are telephone switches interconnected by SS7 links. SSPs perform call processing on calls that originate, tandem, or terminate at that site. SSPs generate SS7 messages to transfer call-related information to other SSPs or to query an SCP for routing instructions.

Signal Transfer Points (STP)
Signal Transfer Points are switches that relay messages between network switches and databases. Their main function is to route SS7 messages to the correct outgoing signaling link, based on SS7 message address fields.

Signaling Connection Control Part (SCCP)
Provides address resolution services, such as global title, for locating services within the network. SCCP supports both connectionless and connection-oriented operation.

Signaling Data Links
Used to connect SS7 signaling points. In most countries, these links are 56 Kbps or 64 Kbps data facilities.

Signaling Gateway
A telephone company switch transmits SS7 signals to a signaling gateway. The gateway, in turn, converts the signals into SIGTRAN packets for transmission over IP to either the next signaling gateway or, if the packet destination is not another PSTN, to a softswitch.

Signaling System 7 (SS7)
Signaling System 7 (SS7) is an architecture for performing out-of-band signaling in support of the call-establishment, billing, routing, and information-exchange functions of the public switched telephone network (PSTN).

Telephone User Part (TUP)
The predecessor to ISUP. Now used primarily in China. In France, a national variant called SSUTR2 is still in common use.

Transaction Capabilities Application Part (TCAP)

Used for transporting transaction-oriented data across the SS7 network. TCAP implements standard Remote Operation Service Element (ROSE) services for applications such as GSM-MAP and IS-41. These "applications" provide IN services such as Home Location Register (HLR) or Short Message Service (SMS).

Visitors' Location Register (VLR)
A local database maintained by the cellular provider to track users who are roaming in the provider's home territory.

1.3.2 SS7 Network Description

SS7 networks are composed of Network Elements (SSP, STP, SCP), most of these are either dedicated hardware (Tekelec, Alcatel, …) or specialized software running on top of big-irons OS such as Solaris, HP-UX, etc…

1.3.3 SS7 Addressing

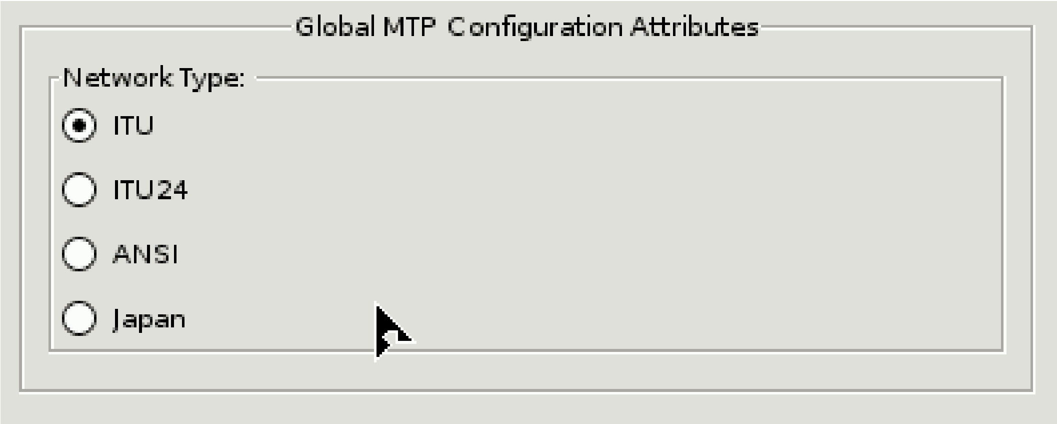
All the addressing in an SS7 network is performed through Point Codes. These are like the IP addresses of the SS7 network. Each Network Element has one.

“Every signalling point (SP) and signalling transfer point (STP), when integrated in an SP, will be allocated its own unique point code.”, [2] (ITU / CCITT Recommendation Q.700 (03/93), Introduction to CCITT Signalling System No. 7)

Depending on the source and destination of any SS7 message, each SS7 message has two Point Code in its description:

- Originating Point Code (OPC)
- Destination Point Code (DPC)

These Point Codes can have several format and define the network type:



Global MTP Configuration Attributes

Network Type:

● ITU

○ ITU24

○ ANSI

○ Japan

Each of these network type has a different address type, changing the number of bits on which Point Codes are coded.

All the Network Elements are organized in SS7 networks to manage the traffic at the local, regional and national level. Then, each operator's SS7 network is connected to other operator's network. Of course, each SS7 network of each operator is independent from the other SS7 networks. To enable correct routing of SS7 messages from one SS7 network to another, a translation between dialed digits and Destination Point Codes is necessary. That is called the Global Title Translation (GTT).

This GTT is done in the Signalling Connection Control Part (SCCP). You can picture that as a sort of NAT between international addresses (phone numbers for example, but other kind of addresses can be used) and local addresses (Point Codes) which define a precise Network Element within the telecom operator network.

1.3.4 Business & People

With the opening up due to deregulation the number of CLECs (competitive operator) that have been interconnecting with ILECs (Incumbent or legacy operators) has skyrocketed. Interconnection is necessary to roll out new services between new business partners. They are also mandatory because the telecommunication authority of each country actually dictates what's to be done.

It enables new classes of premium numbers, helps the setup of competitive SMS providers, etc…

Of course, for these new player, an “All IP” infrastructure would make more sense from a financial and management perspective.

In SS7 networks, the management network used to monitor and configure the network element was the first part of the telecommunication backbone to be using TCP/IP.

TCP/IP was formerly seen in telcos as “Only good for windows-only networks and remote printing” ;-) It lacked the reliability and thoroughness of SS7 in the mind of the telecommunication crowd.

But now that the technology is being adapted to TCP/IP with SIGTRAN, SS7 over IP becomes a reality that is changing the face of telecommunication backbone.

Very few people in the network operators (be it Mobile Network Operators, national telecommunication companies or global tier-1 telecommunication provider) actually do know what’s going on with the technical part of the backbone. These people who do know are talented telecommunication engineers, and there are few of them, usually on contract job with network operators, and very often coming from equipment manufacturers.

This has quite an impact in the telecommunication world. The telecommunication experts are already struggling with the huge technical challenge to make a whole telecom network actually work.

1.3.5 Security consideration

This is not an easy task and requires month of setup, roll out, testing. In doing so, security is important but not as important as launching the new network or adding the new services. And very often, the security test won’t be performed on the actual backbone by fear of bringing it down. This management of security is really dangerous: people don’t really know if their network is secure, and pentests or audits should be conducted much more often than they are.

Another aspects that does not help security of the telecom network is that they still function in a “Walled garden” kind of attitude. The engineers think that they are in an internal network, and thus can leave some machines unsecured. The problem is that now with GPRS flows crossing the backbone, constant interconnection, new services rollout and the associated network changes they require, internal IP addresses are often exposed to external entities, be it partner or unwillingly to end-users or the whole Internet.

Network security management in the telecommunication relies on monitoring. That’s good, if it’s done well. Monitoring of telecommunication backbone is done 24 x 7, and from highly centralized Network Operation Centers. But most of the monitoring is usually focused on operational parameters. Very few attention is really paid to technical security system as another part of the system will take care of some security events after or when they occur. Monitoring people will not really watch for logged on sessions coming from weird IP addresses, they will not read the system logs but will focus on the telecom logs. That’s definitely not helping security, while management people believe their system is well monitored for security.

These events looked for by another department are: fraud cases.
Fraud cases are managed by the Fraud Management Department, very often using an FMS (Fraud Management System) which is mainly focused on financial aspects. It will detect some illegal spikes in some usage, it may or may not detect that a pre-paid GSM card has been credited with $10,000 worth of communication credit, but it won’t detect that someone is actually enumerating all the network elements on the SS7 backbone.

Thus, you see that from a security standpoint, some area are really well managed, and in most of the firms, the technological management of security is really under-funded and under-estimated compared to the “fraud” side of the problem.

1.4 SIGTRAN & SCTP

1.4.1 SIGTRAN Protocol Family

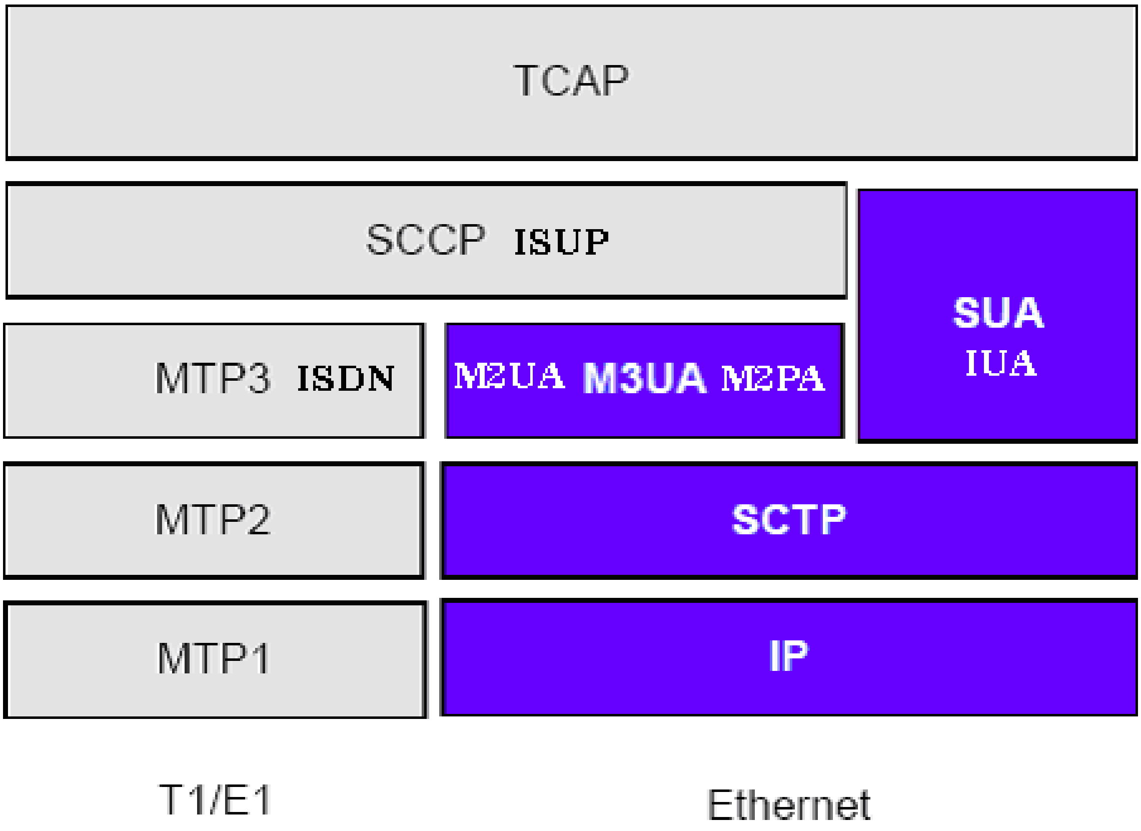
SIGTRAN protocol family is using SCTP [4] as foundation protocol. SCTP sits directly on top of IP, at the same level as ICMP, TCP and UDP.

SIGTRAN protocol family comprises these protocol:

- SCTP – transport data on top of IP with strong requirement
- M3UA – adapt SS7 protocols (MTP3) [5] to be transported over SCTP
- M2UA – same as M3PA but on lower level (MTP2)
- M2PA – same as M2UA but on a peer to peer basis

- SUA – Signaling Connection Control Part User Adaptation Layer
- IUA – ISDN Q.921-User Adaptation Layer

SIGTRAN protocol family exists as a replacement of lower-level OSI-type network:



1.4.2 VoIP and SIGTRAN

VoIP and SIGTRAN come from the same direction: pressure for an all-IP homogenous network.

VoIP deals with things that are common with SIGTRAN: numbering plans, connecting calls, other kind of signaling. Why not replace SS7 completely with VoIP? In fact, it’s quite hard to have the same quality in term of reliability with VoIP than with traditional phone systems. That’s why SIGTRAN is going to happen: it tries to address reliability issue in order to provide the same quality to phone systems using TCP/IP. SIGTRAN attempts even to migrate technologies like ISDN [7] that have received a very mild public response. This shows the depth and span of the normalization attempt.

VoIP will coexist with SIGTRAN, VoIP being used more and more for access network and SIGTRAN still being used for core backbones.

1.4.3 SCTP as SIGTRAN Foundation

SCTP is specified in RFC2960 [3] as “SCTP: Stream Control Transmission Protocol”. It’s like a version of TCP with multiple advantages:

- Advantages
- Multi-homing
- DoS resilient (4-way handshake, cookie)
- Multi-stream
- Reliable datagram mode

Somehow, it can also be compared to UDP when used in datagram mode (i.e. packet mode).

1.4.4 SCTP Software

SCTP is getting adopted quite fast, at least for the SCTP stack. Software that use SCTP are much more rare.

There are tons of proprietary implementations of Operating System’s SCTP stack:

- IBM AIX
- Sun Solaris
- QNX

There are also now plenty of Open source implementations of SCTP stack:

- Linux,
- BSD with KAME project

Actually, the SCTP stack is quite widespread because Linux 2.6 includes it by default. The first time you use an SCTP socket, your kernel will actually load a SCTP kernel module to support SCTP sockets.

There are also Netfilter modules to take care of the filtering and NAT.

Yet, SCTP software is quite scarce on the open Internet, it’s much more present in telco backbones because of it’s growing adoption. Big players like Tekelec are using SCTP.

Also, SCTP is being adopted by other worlds such as clusters (with MPI), high speed transfers, etc…

1.4.5 SCTP Ports & Applications

SCTP is organized with ports as are TCP or UDP. To get into SS7 / SIGTRAN, you need to find entry points which are basically open ports on alive hosts.

That’s the whole goal of SCTPscan, to find SCTP live machines and open ports.

As with TCP or UDP, the address space of SCTP ports is 16 bits, 65536 ports (from 0 to 65535).

SCTPscan embeds a list of common SCTP ports in order to find them by not scanning the 65535 ports but only a smaller subset.

Here is the list of common SCTP ports that you will find in SCTPscan:

```
 100
 128
 260
 250
 1167  cisco-ipsla - Cisco IP SLAs Control Protocol
 1812  radius
 2097
 2225        rcip-itu -- Resource Connection Initiation Protocol
 2427  mgcp-gateway –
      http://en.wikipedia.org/wiki/Media_Gateway_Control_Protocol

 2477
 2577
 2904  m2ua
http://www.pt.com/tutorials/iptelephony/tutorial_voip_mtp.html

 2905  m3ua –
      http://www.ietf.org/rfc/rfc3332.txt
      http://www.hssworld.com/voip/stacks/sigtran/Sigtran_M3UA/overview.htm

 2944  megaco-h248 - Megaco-H.248 text
 2945  h248-binary - Megaco/H.248 binary
```

3097	ITU-T Q.1902.1/Q.2150.3
3565	m2pa
3863	RSerPool's ASAP protocol – http://tdrwww.iem.uni-due.de/dreibholz/rserpool/
3864	RSerPool's ENRP protocol (asap-sctp/tls) -- http://tdrwww.iem.uni-due.de/dreibholz/rserpool/
3868	Diameter
4739	IPFIX (IP Flow Info Export) default port
5000	
5001	
5060	SIP - Session Initiation Protocol
5061	sip-tls
5090	car - Candidate AR
5091	cxtp - Context Transfer Protocol
5675	v5ua
6000	
6790	
6789	
7000	
7102	
7103	
7105	
7551	
7626	simco - SImple Middlebox COnfiguration (SIMCO)
7701	
7800	
8001	
8787	
9006	
9899	sctp-tunneling, actually is usually tcp/udp based
9911	
9900	iua
9901	enrp-sctp - enrp server channel
9902	enrp-sctp-tls - enrp/tls server channel
10000	
10001	
11997	wmereceiving - WorldMailExpress
11998	wmedistribution - WorldMailExpress
11999	wmereporting - WorldMailExpress
14001	sua
30000	
32931	
32768	

This list can be found at:
http://sctp.tstf.net/index.php/SCTPscan/SCTPports

It’s updated from scans in the real world (be it Internet or Intranets) and help better understanding and knowledge of SCTP application.

1.5 Different developing dynamics

Development in SS7 environment was really focused on reliability. Developers used programming languages that helped deliver strong and reliable software such as Eiffel or even proprietary languages.

Actually, that controlled development practice enabled signaling software to be quite secure too, but only as a side effect, not as a target goal from ground up.

The difference now with SCTP and TCP/IP is that telecommunication backbone software can be developed by people who are not accustomed to this kind of reliability requirement or strong languages.

This will generate a different kind of software with different security properties. One can expect that using more flexible languages will enable the creation of feature rich telephone systems, but at the same time, will also generate more bugs and more vulnerability opportunities.

2 SCTPscan in real life

SCTPscan enables you to do many things:

- Scan a single port on a machine
- Scan whole A, B or C networks for machines with SCTP stacks
- Scan for frequent ports on individual machine or whole networks
- Fuzz SCTP stacks on one host
- Portscan a host for the whole 65535 ports
- Start a dummy server which bind socket 10000 which can be later scanned.

2.1 Test & Pentest environment

2.1.1 Supported environments

Supported and tested environment for SCTPscan as host and port scanner are:

- Linux 2.4
- Linux 2.6
- Mac OS X

Supported and tested environment for SCTPscan dummy server are:

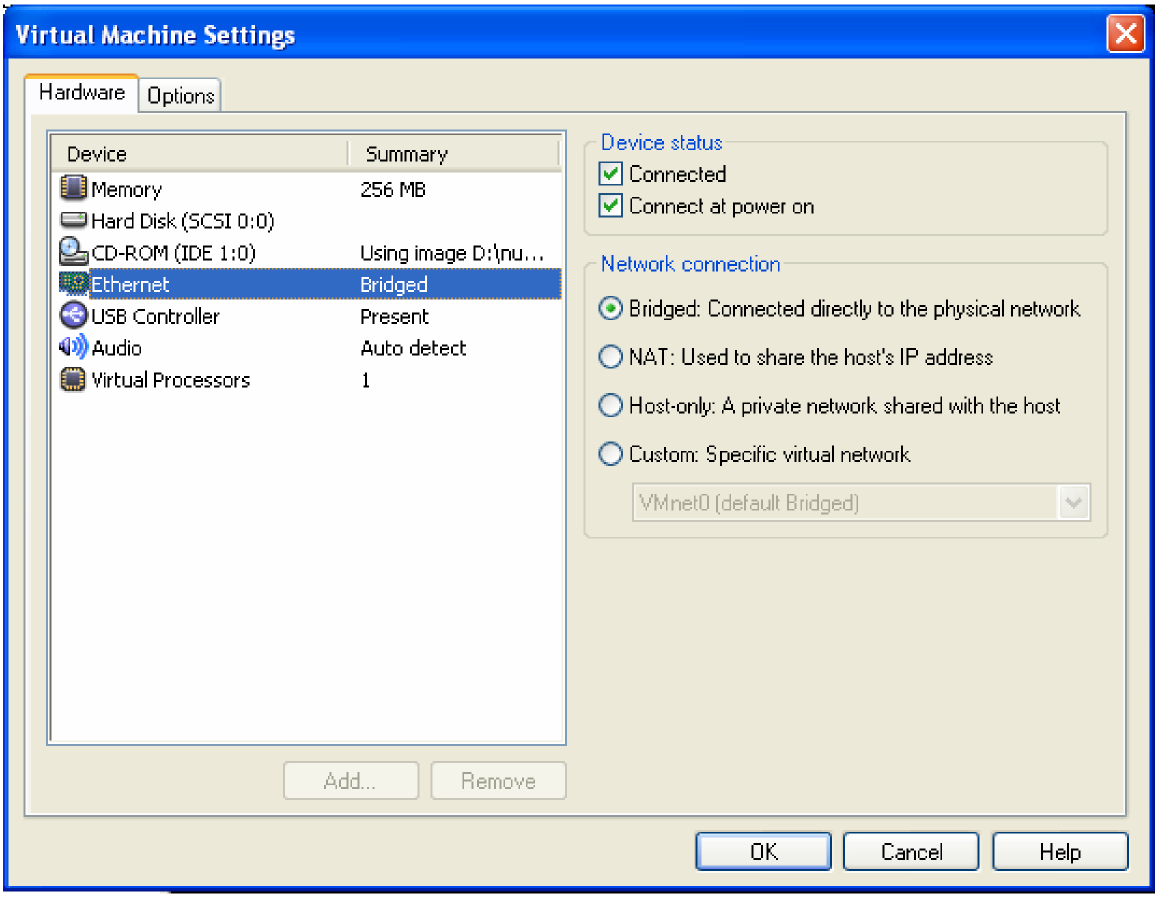
- Linux 2.6
- Mac OS X with SCTP Extension
- Solaris 10 (Open Solaris, NexentaOS, …)

2.1.2 Virtual Machine test setup

Virtual Machines are really handy to setup your test environment. A few requirement make it easier to scan multiple hosts in your environment.

When using VMware images, make sure to select "Bridged mode" for your ethernet connector. You must obtain a separate IP address from the hosted machine.

To perform this, right click on the VMware image then select "Settings...", then click on "Ethernet" and select "Bridged: Connected directly to the physical network".



Virtual Machine Settings

Device	Summary
Memory	256 MB
Hard Disk (SCSI 0:0)	
CD-ROM (IDE 1:0)	Using image D:\nu...
Ethernet	Bridged
USB Controller	Present
Audio	Auto detect
Virtual Processors	1

Device status

☑ Connected

☑ Connect at power on

Network connection

● Bridged: Connected directly to the physical network

○ NAT: Used to share the host's IP address

○ Host-only: A private network shared with the host

○ Custom: Specific virtual network

VMnet0 (default Bridged)

2.1.3 Requirements when using a NATed address

SCTPscan usually doesn’t work behind NATs when scanning outside hosts because most of NATs don’t work with SCTP packets.

If you're on a NATed network, you'll have to do the scanning locally, on your LAN.

To perform this you'll need two different hosts up and running:

  * The scanner (attacker) system (can be Linux 2.4, Linux 2.6, Mac OS X)
  * The scanned (target) system (can be Linux 2.6, Mac OS X, Solaris 10, NexentaOS)

You can achieve that by either pre-installing two computers with these OS, or using VMware images of these OS.

As a side note, your two systems (attacker and target) must of course have two different IP addresses, make sure you know which is which. It's pretty easy to get confused.

Please note that workshop at physical places (Conference, onsite) do not need to care about NAT. We'll scan each other's computers.

2.2 Usage

SCTPscan comes with an handy usage page:

```
root# ./sctpscan
Usage:  sctpscan [options]
Options:
  -p, --port <port>           (default: 10000)
      port specifies the remote port number
  -P, --loc_port <port>           (default: 10000)
      port specifies the local port number
  -l, --loc_host <loc_host>   (default: 127.0.0.1)
      loc_host specifies the local (bind) host for
the SCTP
```

```
      stream with optional local port number
  -r, --rem_host <rem_host>   (default: 127.0.0.2)
      rem_host specifies the remote (sendto) address
for the SCTP
      stream with optional remote port number
  -s  --scan -r aaa[.bbb[.ccc]]
      scan all machines within network
  -m  --map
      map all SCTP ports from 0 to 65535 (portscan)
  -F  --Frequent
      Portscans the frequently used SCTP ports
    Frequent SCTP ports: 1, 100, 128, 260, 250, 1167,
    1812, 2097, 2225, 2427, 2477, 2577, 2904, 2905,
    2944, 2945, 3097, 3565, 3740, 3863, 3864, 3868,
    4739, 5000, 5001, 5060, 5061, 5090, 5091, 5675,
    6000, 6790, 6789, 7000, 7102, 7103, 7105, 7551,
    7626, 7701, 7800, 8001, 8787, 9006, 9899, 9911,
    9900, 9901, 9902, 10000, 10001, 11997, 11998,
    11999, 14001, 30000, 32931, 32768
  -a  --autoportscan
      Portscans automatically any host with SCTP
aware TCP/IP stack
  -i  --linein
      Receive IP to scan from stdin
  -f  --fuzz
      Fuzz test all the remote protocol stack
  -B  --bothpackets
      Send packets with INIT chunk for one, and
SHUTDOWN_ACK for the other
  -b  --both_checksum
      Send both checksum: new crc32 and old legacy-
driven adler32
  -C  --crc32
      Calculate checksums with the new crc32
  -A  --adler32
      Calculate checksums with the old adler32
  -Z  --zombie
      Does not collaborate to the SCTP Collaboration
platform. No reporting.
  -d  --dummyserver
```

```
      Starts a dummy SCTP server on port 10000. You
can then try to scan it from another machine.
  -E  --exec <script_name>
      Executes <script_name> each time an open SCTP
port is found.
      Execution arguments: <script_name> host_ip
sctp_port

Scan port 9999 on 192.168.1.24
./sctpscan -l 192.168.1.2 -r 192.168.1.24 -p 9999

Scans for availability of SCTP on 172.17.8.* and
portscan any host with SCTP stack
./sctpscan -s -l 172.22.1.96 -r 172.17.8

Scans frequently used ports on 172.17.8.*
./sctpscan -s -F -l 172.22.1.96 -r 172.17.8

Scans all class-B network for frequent port
./sctpscan -s -F -r 172.22 -l `ifconfig eth0 | grep
'inet addr:' |  cut -d: -f2 | cut -d ' ' -f 1 `

Simple verification end to end on the local machine:
./sctpscan -d &
./sctpscan -s -l 192.168.1.24 -r 192.168.1 -p 10000

This tool does NOT work behind most NAT.
That means that most of the routers / firewall don't
know how to NAT SCTP packets.
You _need_ to use this tool from a computer having a
public IP address (i.e. non-RFC1918)

root#
```

Refer to this usage page in case of doubt on the tools' usage.

2.2.1 Scanning

SCTPscan doesn't do full SCTP Association as the 4-way handshake: it would be slow, costly in term of computing. Even worse, doing a full 4-way handshake would establish a real connection to the target host in case of success which would be as obvious as a connect scan with TCP.

This scan is made possible by sending SCTP packets with an INIT chunk and listening for SCTP packets with an INIT_ACK chunk.

So let’s call SCTPscan “stealth SCTP scanning” as TCP syn scanning was called.

Here is an example of usage of SCTPscan to synscan a whole C-class network with INIT stealth scan:

```
root# ./sctpscan -s -r 192.168.0
Netscanning with Crc32 checksumed packet
192.168.0.3 SCTP present on port 10000
End of scan: duration=5 seconds packet_sent=254
packet_rcvd=206 (SCTP=1, ICMP=203)
root#
```

2.2.2 Automatic portscanning

There are some handy option which enable SCTPscan to automatically portscan a machine as soon as it finds that this hosts actually supports SCTP.

This is made possible because SCTPscan will listen for SCTP ABORT packets or ICMP protocol unsupported packet. If we receive a SCTP ABORT from one machine we scanned, then it means for sure that SCTP is supported on this machine. Maybe some ports are open, so SCTPscan will autoportscan the machine. The problem is that now with Linux 2.6 by default replying with SCTP ABORT, you get a lot of machines to autoportscan even if there’s absolutely no open ports in the whole network.

On the other hand, if you send INIT probes during a network scan and don’t receive any packet, it may either means that the packet was dropped by a firewall or that the machine respects the RFCs and doesn’t reply to INIT packets to not-open ports.

This last behavior leads to a problem: we autoportscan machines which are protected by firewall that drops unauthorized packets.

Both these reasons may lead to vert slow scan when using this option.

```
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#
```

2.2.3 Dummy server mode

Dummy server mode binds a socket on port 10000 which can be later scanned by another machine.

This is very useful when you want to try the scanning platform.

If your host doesn’t have support for SCTP, you’ll get a message like this:

```
root# ./sctpscan -d
socket: Protocol not supported
Your kernel does not seem to support SCTP sockets.
It's supported by Linux Kernel 2.6 or Solaris 10.
For Linux, you may want to run as root: modprobe sctp
root#
```

Note:
But you only need a SCTP-aware kernel to run dummyserver. Scanning is ok with systems without SCTP support such as 2.4 linux kernels!

To get support for your OS for SCTP sockets, try these:

Linux

Try as root:

```
 modprobe sctp
```

Then rerun:

```
 sctpscan --dummyserver
```

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.

In case of success, you’ll get this:

```
root# ./sctpscan -d
Trying to bind SCTP port
Listening on SCTP port 10000
```

And it will hang there, waiting for connections.

It may be a good idea to put it in the background and use some tools to monitor packets (tcpdump, ethereal and wireshark will decode SCTP packets).

2.3 Compiling and troubleshooting

2.3.1 Compiling

The platform I advise to use is Linux (kernel 2.4 or 2.6).

You need to have GLIB 2.0, especially the development version so that you can compile SCTPscan with it.

On linux:

```
cc -g sctpscan.c -o sctpscan -I /usr/include/glib-
2.0/ -I /usr/lib/glib-2.0/include/ -lglib-2.0
```

On MacOSX:

```
cc -g sctpscan.c -o sctpscan -L/sw/lib/ -I
/sw/include/glib-2.0/ -I /sw/lib/glib-2.0/include/ -
lglib-2.0
```

2.3.2 Common problems

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

Answer:
Your kernel does not support SCTP sockets. For example, SCTP sockets are supported by default 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
```

MacOS X:
Load the SCTP kernel extension as said previously:

```
       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?'
```

2.3.3 Kernel conflicts with 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]
```

2.3.4 Kernel conflicts on MacOS X

MacOS X’s Darwin kernel support for SCTP is affecting much more the scan process. Here is an example of a scan session with and without the SCTP kernel extension:

```
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.

3 Technology discussion

3.1 Packets & Chunks

3.1.1 SCTP Packets

Each SCTP packets sits directly on top of IP with the following packet format (from RFC2960):

```
3.  SCTP packet Format

   An SCTP packet is composed of a common header and chunks. A chunk
   contains either control information or user data.

   The SCTP packet format is shown below:

       0                   1                   2                   3
       0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |                        Common Header                          |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |                          Chunk #1                             |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |                           ...                                 |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |                          Chunk #n                             |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

3.1.2 Header

At the top of packet, you’ll always have header looking like this following packet(from RFC2960). This header contains Source Port and Destination Port which will be the main thing SCTPscan will be manipulating in the packet. Then the checksum which was originally Adler32 and now is CRC32.

```
3.1 SCTP Common Header Field Descriptions

                         SCTP Common Header Format

       0                   1                   2                   3
       0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |     Source Port Number        |     Destination Port Number   |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |                      Verification Tag                         |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |                           Checksum                            |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

3.1.3 SCTP Chunks

Each SCTP packet can contain one or more chunks. Here is a chunk type list from RFC 2960:

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)
15 to 62	- reserved by IETF
63	- IETF-defined Chunk Extensions
64 to 126	- reserved by IETF
127	- IETF-defined Chunk Extensions
128 to 190	- reserved by IETF
191	- IETF-defined Chunk Extensions
192 to 254	- reserved by IETF
255	- IETF-defined Chunk Extensions

The types in bold are the chunk types we’ll be using when scanning.

There are rules about which chunks can be bundled together except a few ones as RFC 2960 specifies:

```
Multiple chunks can be bundled into one SCTP packet up to the MTU
size, except for the INIT, INIT ACK, and SHUTDOWN COMPLETE chunks.
These chunks MUST NOT be bundled with any other chunk in a packet.
See Section 6.10 for more details on chunk bundling.
```

That’s an interesting vector for fuzzing by combining different chunks in a single packet.

3.2 INIT vs SHUTDOWN_ACK Packet Scanning

To scan host, INIT may not be the best candidate in order to generate a response by the remote host.

In RFC 2960, there’s an interesting specification about “out of the blue” packets (some people will catch the funny reference to Out Of Band signaling of the blue box era):

```
“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.”
```

That’s a new way to elicit answers even if not answering ABORTs to INITs targeted at not-opened port. According to specification, hosts implementing SCTP should send back an SHUTDOWN COMPLETE packet. In practice, this rarely happens. This would enable better autoportscan results.

3.3 Checksums

SCTPscan can use either Adler-32 or CRC32 checksums and send both packet for each attempt, so that you can target very early implementation of SCTP stack on old machines.

3.4 Collaborative scanning

By default, results of the scans are sent to a central collaborative platform so that we can make statistics on ports usage and versions usage. This reporting capability can be disabled with the zombie argument.

3.5 SCTPscan Improvements

There’s lots of room for improvement in SCTPscan such as:

- Complete implementation using libpcap
- Identification of scan response packet (so that display does not mix up several instance of SCTPscan scanning different hosts)
- Better parameter checking

4 Results

SCTP is scare in the wild. Very few Signaling Gateways actually are open to any connection and exposed on the Internet. We see a huge different when actually auditing telecommunication internal networks with a lot of hosts using SCTP there. Another interesting result is that SCTP is completely below the radar. A lot of firewalls filter TCP and UDP but have no rule for SCTP and let it be treated by default. Default being often considered only like “control” protocols such as ICMP, an interesting number of firewall actually let SCTP pass. We found very few firewall that logged dropped SCTP packets when they did. In term of IDS, no current IDS do any detection on SCTP level. The idea of SCTP based backdoor has been raised during comments and emails to TSTF research team. Most notably, after our initial tests of SCTPscan, we noticed that for example, dshield.org would never pick up our IP as a “scanner IP” which constitute an interesting statement about detection.

5 Conclusions

SCTPscan is one of the first “open” tool (open source, aimed at any network engineer) that targets the protocol suite that’s used on telecommunication backbones. It will help examine networks and assess more precisely the nature and quality of these backbones, but also of the equipments being used in these telecommunication environment. Thus, it can be considered as one “full disclosure” tool, but at the same time an attack-vector toward the “walled garden” or “security through obscurity” attitude that has been around telecommunication backbones.

The current situation and dynamic of this field is that telecom backbone world is opening up. This means that there is a commoditization of the field and that some status of the past will be challenged. There are also power monopolies that are being changed, and a subtle fight can be understood. For example, between incumbent operators and emerging ADSL operators aiming at phone service for example. A more compelling example of power struggle is the one between ITU and IETF. The SIGTRAN initiative actually defines the migration from ITU, a centrally controlled, formerly closed-source minded (remember you had to pay a lot of money to access ITU recommendation still recently) to IETF, a decentralized work force composed of engineers and scientists from the biggest corporations of the globe. It’s also a shift from a European-born entity to a US-born entity, from a telecom-centric organization to a network-centric organization.

We will see how the whole SIGTRAN initiative will affect the telecom world, and within this context, will affect the view of security in telecom environment. SCTPscan is only one small –but fundamental– part of a broader set of tools and services. I think we will benefit from the same openness and discovery spirit in higher-level protocols of the telecommunication world.

6 References

[1] ITU / CCITT Recommendation Q.140-Q.180 (11/88), Specifications of Signalling System No. 5, Initially published in Blue Book Fascicle VI.2 (1988).

[2] ITU / CCITT Recommendation Q.700 (03/93), Introduction to CCITT Signalling System No. 7

[3] RFC2960, Stream Control Transmission Protocol, by R. Stewart, Q. Xie, K. Morneault, C. Sharp, H. Schwarzbauer, T. Taylor, I. Rytina, M. Kalla, L. Zhang, V. Paxson, October 2000, Updated by RFC3309.

[4] RFC4166, Telephony Signalling Transport over Stream Control, Transmission Protocol (SCTP) Applicability Statement, by L. Coene, J. Pastor-Balbas, February 2006.

[5] RFC4666, Signaling System 7 (SS7) Message Transfer Part 3 (MTP3) - User Adaptation Layer (M3UA), by K. Morneault, Ed., J. Pastor-Balbas, Ed, September 2006. Obsoletes RFC3332.

[6] Emmanuel Gadaix presentation “NGN Security - Next Generation Nightmare?”, Bellua CyberSecurity Conference 2006

[7] RFC4233, Integrated Services Digital Network (ISDN) Q.921-User Adaptation Layer, by K. Morneault, S. Rengasami, M. Kalla, G. Sidebottom, January 2006, Obsoletes RFC3057