Securing the network edge with OpenSIPS

John Quick
Smartvox Limited
Network Edge – a simplistic view

Private Trusted LAN

OpenSIPS

Public Untrusted Internet
Real World

Database

Core Telephony Infrastructure

IVR/PBX/Media Servers

Gateway

Public telephone network

Outbound Call centre

Dialler

Mobile user SIP Client

SIP Trunks

Internet

Hosted PBX

Customer handsets

Customer Premises 1

IP-PBX

Extension phones

Carrier 1

Carrier 2

Private routing

Private routing

OpenSIPS
Who can be trusted?
OpenSIPS: Role separation

- **OpenSIPS Carrier Proxy**
  - Database
  - Core Telephony Infrastructure
  - IVR/PBX/Media Servers
  - LCR

- **OpenSIPS Registrar/Proxy**
  - Far-end NAT handling
  - User authentication

- **Internet**
  - Trusted
  - NOT Trusted

- **Carriers**
  - Customer Premises 1
  - Extension phones
  - IP-PBX

- **SIP Trunks**
  - Hosted PBX
  - Mobile user SIP Client

- **Mobile user**
  - SIP Client

- **Hosted PBX**
  - Extension phones

- **Customer Premises 1**
  - IP-PBX
  - Extension phones

- **Customer handsets**
  - Extension phones
Call Routing: Clear identification of risk
Possible Network Configuration Scenarios

• Dual network interfaces on the OpenSIPS server

• All VoIP servers in a DMZ

• OpenSIPS in the DMZ and Media Servers on the LAN

• OpenSIPS in the DMZ and Media Servers behind 1-to-1 NAT

• OpenSIPS in the Cloud
Network Zoning: Dual interface solution

LAN
- Core Telephony Infrastructure
- IVR/PBX/Media Servers

OpenSIPS Registrar + RTPProxy

Internet
- SIP Trunks
- Hosted PBX

Customer Premises 1
- Extension phones
- IP-PBX

Customer handsets

Requires few public IP addresses
No complex firewall configuration

Requires RTPProxy in bridging mode
RR headers need fixing in both directions
Security vulnerability if server is hacked
Network Edge – traditional zoning model

- **INTERNET**: Controlled access WAN to DMZ
- **DMZ**: OpenSIPS
- **LAN**: Database
- **DMZ to LAN**: Highly controlled access
- **WAN to LAN**: Block WAN to LAN
Network Zoning: OpenSIPS in the DMZ

- **DMZ**
  - OpenSIPS Registrar
  - + A Media Proxy
  - SIP
  - RTP

- **LAN**
  - Core Telephony Infrastructure
  - IVR/PBX/Media Servers
  - SIP
  - RTP

- **Internet**
  - SIP Trunks
  - Hosted PBX

- **Customer Premises 1**
  - Extension phones
  - IP-PBX

- **Customer handsets**

**Requirements:**
- Requires few public IP addresses
- Media Servers on the LAN – convenient
- **No** direct Internet access – secure

**Complexity:**
- Firewall/Router complexity of rules
- Firewall/Router load – throughput x2
Network Zoning: VoIP Servers in DMZ

DMZ

OpenSIPS Registrar

Core Telephony Infrastructure

IVR/PBX/Media Servers

LAN

Database

1-to-1 NAT acceptable

Firewall/Router/UTM Device

SIP Trunks

Customer Premises 1

Extension phones

IP-PBX

Customer handsets

SIP

RTP

1-to-1 NAT acceptable

Media Proxy is not necessary
Only RTP ports on Media Servers are open

Media Proxy is not necessary
Only RTP ports on Media Servers are open

Requires more public IP addresses
Quite complex Access Control rules on Firewall
RR headers may need fixing when 1-to-1 NAT is used
Network Zoning: OpenSIPS in the Cloud

- Tightly controlled Internet access
- Multi-tenant possibilities
- Potential as Value Added solution

Complex OpenSIPS configuration

DMZ
- Core Telephony Infrastructure
- Registrar/PBX

Internet
- OpenSIPS Transparent Proxy
- Mediaproxy

Internet

Mobile user
- SIP Client

Customer handsets
- SIP Clients
A presentation of two halves

• Network Topology

• Application level Security
OPTIONS sip:100@123.45.251.24 SIP/2.0.
Via: SIP/2.0/UDP 127.0.0.1:6241;branch=z9hG4bK-2252719110;rport.
Content-Length: 0.
From: "sipvicious"<sip:100@1.1.1.1>;tag=303539631363413939373134.
Accept: application/sdp.
User-Agent: friendly-scanner.
To: "sipvicious"<sip:100@1.1.1.1>.
Contact: sip:100@127.0.0.1:6241.
CSeq: 1 OPTIONS.
Call-ID: 10971752559881716145531.
Max-Forwards: 70.
Phase 1: Server Discovery

OPTIONS Ping like radar

Response may indicate:

A SIP server is at this address

The type of product and version of software installed
2015 Survey

2015 Estimated Fraud Losses by Type (in $ USD Billions)

- International Revenue Share Fraud (IRSF): $10.8
- Premium Rate Service: $3.8
- Wholesale Fraud: $2.0
- Theft/Stolen Goods: $2.8
- Arbitrage: $2.9
- Device/Hardware Reselling: $2.3
- Domestic Revenue Share (DRSF): $2.1
- Commissions Fraud: $1.5
- Friendly Fraud: $0.9
- Service Reselling (e.g., Call Sell): $0.9
- Private Use: $0.8
- Theft/Compromise of data (e.g., logins): $0.5

Communications Fraud Control Association
$10,800,000,000
International Revenue Share Fraud

$9,890,000,000
Annual Government Expenditure Budget for Estonia for 2016*

IRSF: Fastest growing telecoms fraud

- Estimated Total Global fraud loss: $40.1 Billion to $38.1 Billion
- International Revenue Share fraud: $3.8 Billion to $10.8 Billion

Source: Communications Fraud Control Association
The phases of a typical attack

1. **Server discovery**
   - OPTIONS / INVITE
   - Within 1 hour
   - 2 requests / hour
   - Multiple sources

2. **Account discovery and password guessing**
   - REGISTER / INVITE
   - After approx. 6 hours*
   - 10,800 requests / hour*

3. **Access testing**
   - INVITE
   - Depends on Phase 2
   - 10’s requests / day

4. **Call pumping to IPR destinations**
   - INVITE
   - Could wait days or even weeks for the best opportunity
   - As fast as your system will allow

* Source: Measured rate on an Asterisk server used for honeypot testing
Mechanics of the fraud (phase 4)

Compromised ITSP Service

Internet

$\rightarrow$

International Premium Rate numbers

Somalia; Gambia; Sierra Leone; Lithuania; Latvia; Guinea; Cuba; Afghanistan; Albania; Algeria; Ecuador; Serbia; Azerbaijan; Kosovo; Bosnia and Herzegovina; Inmarsat
Some precautions possible with OpenSIPS

Detection of malicious SIP requests
• Identifying characteristics

Logging and automated inspection of logs
• xlog
• fail2ban

Call traffic analysis
• Automated analysis
• Fraud Detection Module; White and Black lists
Detect and Reject Malicious SIP Requests

Useful for: **Phase 1 and Phase 2**

Principles used: **User-Agent header contains characteristic string**

Coding example:

```plaintext
if ($ua =~ "friendly|sipcli|VaxSIPUser|VoIP.*v1")
    exit;

if ($fd == "1.1.1.1" || $fU == "nm")
    exit;
```
Logging Suspicious SIP Requests

Useful for: **Phase 2**

Principles used: **Write to different log streams depending on category**
- **L_INFO**, **L_NOTICE**, **L_WARN**, **L_ERROR**, **L ALERT**

Coding example:

```
xlog("L_INFO", "--> $rm R-URI=$ru LR Request (Rcvd $si:$sp) Call-ID=$ci\n");
```

```
xlog("L_WARN", "!--> !!WARNING: $rm request to $ru; Sender is NOT Registered."
(From $fu ($si:$sp) UA=$ua) Call-ID=$ci\n");
```
Check return code from authorize functions

```plaintext
if (!www_authorize('', "subscriber")) {
    switch ($retcode) {
    case -1:
        xlog("L_ALERT", "$si:$sp Register_Unknown_User From=$fu\n");
        break;
    case -2:
        xlog("L_ALERT", "$si:$sp Register_Bad_Password From=$fu\n");
    }
    www_challenge('', "0");
    exit;
}
```
Mapping log_level parameter to log files

/etc/rsyslog.conf

# Send opensips messages to their own log files
$template MyTimeStamp,"%TIMESTAMP:::date-pgsql% %msg\n"
local0.=warn,local0.=notice        -/var/log/opensips.log;MyTimeStam
local0.=alert                     -/var/log/osips_alert.log;MyTimeStam
local0.err,local0.=info           -/var/log/osips_error.log;MyTimeStam
<table>
<thead>
<tr>
<th>Date</th>
<th>Time</th>
<th>IP Address</th>
<th>Service Type</th>
<th>Event Type</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>2016-04-20</td>
<td>19:04:13</td>
<td>195.154.182.164:5074</td>
<td>Invite_from_unknown_source</td>
<td>From=sip:115@72.15.16.29</td>
<td>RURI=sip:001445209698@72.15.16.29 UA=sipcli/v1.8</td>
</tr>
<tr>
<td>2016-04-20</td>
<td>20:09:02</td>
<td>195.154.182.164:5074</td>
<td>Invite_from_unknown_source</td>
<td>From=sip:115@72.15.16.29</td>
<td>RURI=sip:001445209698@72.15.16.29 UA=sipcli/v1.8</td>
</tr>
<tr>
<td>2016-04-20</td>
<td>21:17:00</td>
<td>195.154.182.164:5087</td>
<td>Invite_from_unknown_source</td>
<td>From=sip:115@72.15.16.29</td>
<td>RURI=sip:9001445209698@72.15.16.29 UA=sipcli/v1.8</td>
</tr>
<tr>
<td>2016-04-20</td>
<td>22:22:52</td>
<td>195.154.182.164:5070</td>
<td>Invite_from_unknown_source</td>
<td>From=sip:115@72.15.16.29</td>
<td>RURI=sip:9001445209698@72.15.16.29 UA=sipcli/v1.8</td>
</tr>
<tr>
<td>2016-04-20</td>
<td>23:29:12</td>
<td>195.154.182.164:5070</td>
<td>Invite_from_unknown_source</td>
<td>From=sip:115@72.15.16.29</td>
<td>RURI=sip:41445209698@72.15.16.29 UA=sipcli/v1.8</td>
</tr>
<tr>
<td>2016-04-21</td>
<td>07:39:26</td>
<td>212.129.2.189:9916</td>
<td>Register_Wrong_Domain</td>
<td>Contact=sip:786542789@212.129.2.189:9916</td>
<td>UA=VoIP v11.2.4</td>
</tr>
<tr>
<td>2016-04-21</td>
<td>07:39:26</td>
<td>212.129.2.189:9916</td>
<td>Register_Wrong_Domain</td>
<td>Contact=sip:786542789@212.129.2.189:9916</td>
<td>UA=VoIP v11.2.4</td>
</tr>
<tr>
<td>2016-04-21</td>
<td>13:43:00</td>
<td>163.172.194.75:6746</td>
<td>Register_Wrong_Domain</td>
<td>Contact=sip:202@163.172.194.75</td>
<td>UA=eyeBeam release 3006o stamp 17551</td>
</tr>
<tr>
<td>2016-04-21</td>
<td>13:43:20</td>
<td>163.172.194.75:6746</td>
<td>Register_Wrong_Domain</td>
<td>Contact=sip:202@163.172.194.75</td>
<td>UA=eyeBeam release 3006o stamp 17551</td>
</tr>
<tr>
<td>2016-04-21</td>
<td>13:44:00</td>
<td>163.172.194.75:6746</td>
<td>Register_Wrong_Domain</td>
<td>Contact=sip:202@163.172.194.75</td>
<td>UA=eyeBeam release 3006o stamp 17551</td>
</tr>
<tr>
<td>2016-04-21</td>
<td>14:53:22</td>
<td>46.166.165.79:5392</td>
<td>Register_Wrong_Domain</td>
<td>Contact= From=sip:100@1.1.1.1 To=sip:100@1.1.1.1</td>
<td>UA=friendly-scanner</td>
</tr>
<tr>
<td>2016-04-21</td>
<td>20:16:19</td>
<td>80.241.209.130:5010</td>
<td>Register_Wrong_Domain</td>
<td>Contact=sip:100@80.241.209.130:5010</td>
<td>UA=VoIP SIP v12.1.0</td>
</tr>
<tr>
<td>2016-04-21</td>
<td>20:16:19</td>
<td>80.241.209.130:5010</td>
<td>Register_Wrong_Domain</td>
<td>Contact=sip:100@80.241.209.130:5010</td>
<td>UA=VoIP SIP v12.1.0</td>
</tr>
<tr>
<td>2016-04-21</td>
<td>20:16:20</td>
<td>80.241.209.130:5010</td>
<td>Register_Wrong_Domain</td>
<td>Contact=sip:100@80.241.209.130:5010</td>
<td>UA=VoIP SIP v12.1.0</td>
</tr>
</tbody>
</table>
Fail2ban configuration

```
/etc/fail2ban/filter.d/osips-alert.conf

# Regexp to block repeat attempts sending invalid requests to OpenSIPS
[Definition]
failregex = <HOST>.*Invite_from_unknown.*$
           <HOST>.*Request_to_unknown_dest.*$
           <HOST>.*Register_Wrong_Domain.*$
           <HOST>.*Register_Bad_Credential.*

ignoreregex =
```

```
/etc/fail2ban/jail.d/osips-alert.conf

[osips-alert]
enabled   = true
filter    = osips-alert
action    = iptables[name=osips-alert, port="5060", protocol=udp]
logpath   = /var/log/osips_alert.log
maxretry  = 3
findtime  = 7200
bantime   = 10800
```
Avoiding the weaknesses of fail2ban

```sql
avp_db_query("SELECT username FROM location WHERE received LIKE 'sip:$si%', '"avp(usrname)');
if (is_avp_set("$avp(usrname)"))
    # Address is being used by a registered device so should not be blocked

avp_db_query("SELECT username FROM location WHERE contact LIKE 'sip:%%si%', '"avp(usrname)');
if (is_avp_set("$avp(usrname)"))
    # Address found in the contact field of the location table – don’t block
```

Avoids blocking a legitimate customer

Might give hackers a way around your security
Why do we need Call Traffic Analysis?

Because Customers are a weak link
- Their equipment may get hacked
- User Account Credentials may get stolen

Because we make silly mistakes
- Maintenance of firewall rules
- Vulnerability when infrastructure is changed

Because the hackers are clever and inventive
- May hack your Provisioning Server
- May set up a customer account with no intention of paying
Fraud Detection through Traffic Analysis

3 qualifying parameters

- Time
- Destination
- User Account
OpenSIPS Fraud Detection Module

**Good**

- Includes Time (hour and day), Destination and User Account
- Measures Total & Concurrent Calls; also Calls/Minute
- Data driven

**Not so good!**

- Only available in v2
- Users or Destinations: Cannot be assigned to a group (categorized)
- Destinations: Prefix only – not a regex
- Metric for Total does not persist through a restart
Concurrent Call Counting using Dialog Module

loadmodule "dialog.so"

modparam("dialog", "log_profile_hash_size", 8) # allows up to 256
modparam("dialog", "profiles_with_value", "concurrent")

route[
    # Remember the User a/c (using $au) and set a flag for onreply
    store_dlg_value("UserAccount", "$au");
    set_dlg_flag("12");
    # Get the number of concurrent calls already active for this user account
    get_profile_size("concurrent", "$au", "$var(numcalls)");
    # Test if number of concurrent calls is within permitted limit..

    onreply_route[]
    if ($rm=="INVITE" && status=="200" && is_dlg_flag_set("12")) {
        fetch_dlg_value("UserAccount", "$var(profvalue)");
        set_dlg_profile("concurrent", "$var(profvalue)");
        reset_dlg_flag("12");
    }
]
Bespoke solution: Cached Categorisation List

**Startup_route**
- Load data into local cache – e.g. from a DB table
- Each record is a value/category pair

**Main route**
- Set ‘User Account’ – e.g. using $au (Auth User) or $si (Source IP)
- Use cached data to set a category – e.g. using $rU (dialed number)
- Create a compound “key” to use in set_dlg_profile()
- Use ‘Concurrent Call Counting’ to track number of calls in selected category from that User
Bespoke solution: Coding ideas/suggestions

loadmodule "cached_local.so"

modparam("cachedb_local", "cache_table_size", 10)  # Allows over 1000 entries

startup_route {
    # Create a cached list of Destination prefixes – e.g. Sierra Leone & Somalia
    cache_store("local", "+616", "HIGHRISK");
    cache_store("local", "+686", "HIGHRISK");
}

route[]
    # Check destination number against cached list of prefixes – for example:
    $var(testval) = $(rU{s.substr,0,4});
    cache_fetch("local","$var(testval)",$avp(retval))
...
    $var(compoundkey) = $au + $avp(retval)
    store_dlg_value("UserAccount", "$var(compoundkey)");
    set_dlg_flag("12");
    get_profile_size("concurrent", "$var(compoundkey)", "$var(numcalls)");
Third Party Solutions

Is fraud protection provided by your Carrier?

How does it work – technically and commercially?

Do they actively research/improve fraud prevention solutions?

  e.g. Simwood

Bolt-on Solutions

  e.g. FRS Labs PRISM database
Please don’t get caught by hackers

The End
Thank you