Suricata’s approaching VoIP networks

Suricata’s approaching VoIP networks

Agenda

1. Introduction

2. Voice over IP

3. SIP protocol

4. SIP security

4. Suricata SIP parser

5. Live demo

Voice over IP

- Group of technologies used for the delivery of voice and multimedia traffic over Internet Protocol (IP) networks like the Internet.

- It works like a phone service using the Internet rather than the traditional telephone service offering lesser rates as compared to many phone companies.

Voice over IP – Network elements

- User agent: endpoint able to send and receive SIP messages and manage SIP sessions. The user agent client (UAC) sends SIP requests, the user agent server (UAS) receive requests and returns a SIP response

- Proxy server: component with UAC and UAS capabilities that functions as an intermediary entity for the purpose of performing requests between other network elements

- Registrar: SIP endpoint that provides a location service, it accepts REGISTER requests and records the address and other parameters from the user agent

What SIP is

- Signaling protocol used to manage a multimedia session
  - Session is nothing but a simple call between two endpoints
- Defined in RFC 3251
- It can be used for two-party (unicast) or multiparty (multicast) sessions

What SIP isn‘t

- It’s not an audio protocol
- It’s not a video protocol
- It’s not a data protocol

- **It’s mostly applied to VoIP but it’s not a VoIP protocol**

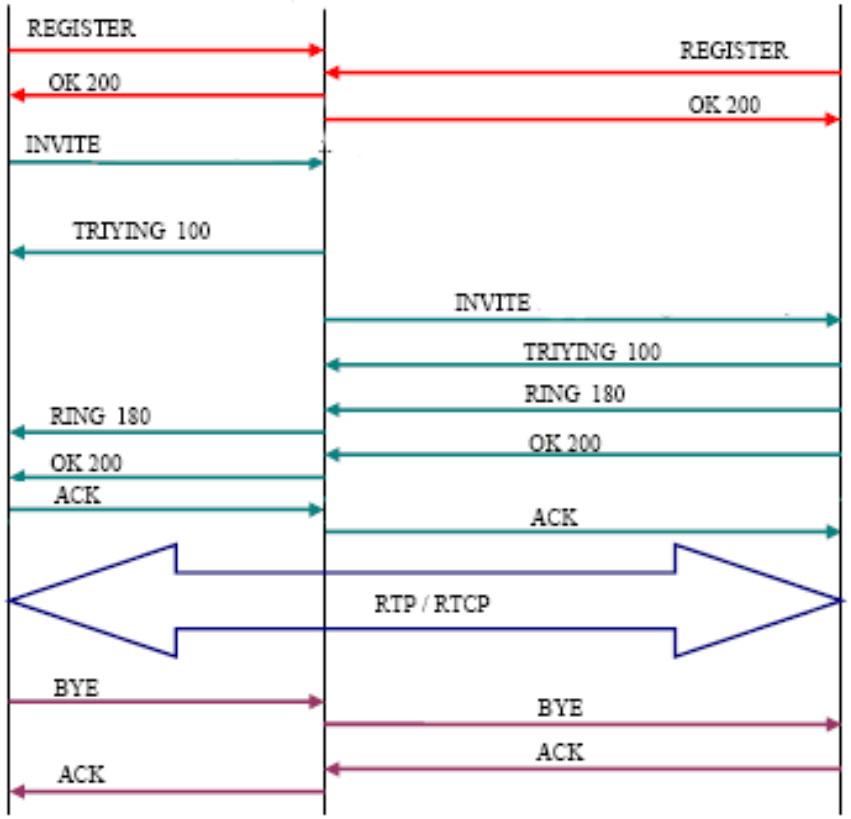
What does SIP do?

- “It tells you the presence of the other party, makes a connection and lets you do whatever you want over the connection, but it has no idea of what’s going over the connection.”

- **SIP is a media-independent protocol.**

VoIP Alliance

- Real-Time Protocol (RTP) is a specialized application layer for transporting audio and video when real-time streaming is needed.
- The RTP control protocol (RTCP) works alongside RTP to provide information about RTP packet delivery and manage the quality of voice service
- Session Description Protocol (SDP) works alongside SIP to specify which types of media the SIP clients in the session can actually support.



SIP messages

- It’s based around request/response transactions (like HTTP)
  - Each transaction is made up of a request and at least one response
- SIP messages are similar to HTTP and SMTP
- It reuses most of the header fields, encoding rules and status code of HTTP
- The URI syntax is similar to SMTP

SIP Request

Method SP Request-URI SP Version CRLF
Headers CRLF

```
REGISTER sip:sip.cybercity.dk SIP/2.0

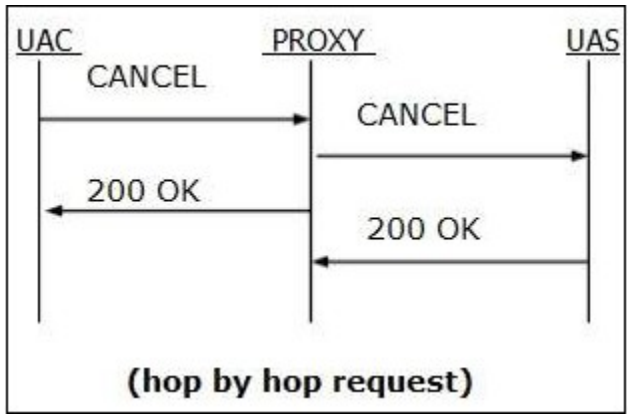
Via: SIP/2.0/UDP 192.168.1.2;branch=z9hG4bKnp151248737-46ea715e192.168.1.2;rport
From: <sip:voi18063@sip.cybercity.dk>;tag=903df0a
To: <sip:voi18063@sip:cybercity.dk>
Call-ID: 578222729-4665d775@578222732-4665d772
Contact:
<sip:voi18063@192.168.1.2:5060;line=9c7d2dbd8822013c>;expires=1200;q=0.500>
Expires: 1200
Cseq: 68 REGISTER
Content-Length: 0
Max-Forwards: 70
User-Agent: Nero SIPPS IP Phone Version 2.0.51.16
```

SIP Request - Methods

- INVITE
  - Establish a media session between the user agents
  - A session is considered established if an INVITE has received a success response (2xx) or an ACK has been sent
- BYE
  - Terminate an established session, it can be sent either by the caller or the callee to end a session
  - It cannot be sent by a proxy server
  - It cannot be sent to a pending INVITE or an unestablished session

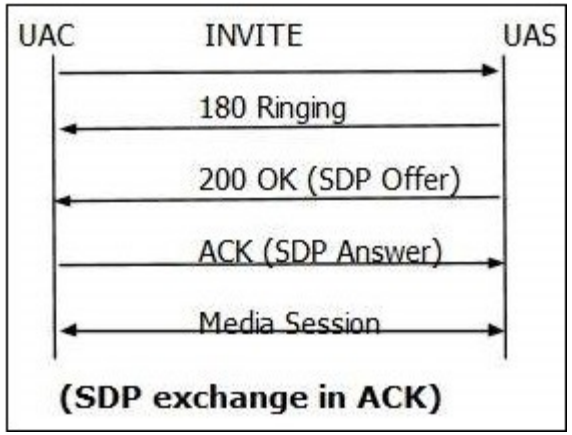
SIP Request – Methods (2)

- REGISTER
  - Performs the registration of a user agent, it's sent by a user agent to a registrar server
  - It carries the AOR (Address of Record) in the **To** header of the user that is being registered
- CANCEL
  - Used to terminate a session which is not established
  - It can be sent either by a user agent or a proxy server



SIP Request – Methods (3)

- ACK
  - Used to acknowledge the final responses to an INVITE request
  - It always goes in the direction of INVITE



- OPTIONS
  - Used to query an user agent or proxy about its capabilities
  - A proxy never generates an OPTIONS request

SIP Request – URI

- **sip(s):user:password@host:port;uri-parameters?headers**
- user
  - The identifier of a particular resource at the host being addressed
- password
  - Password associated with the user
- Host
  - The host providing the SIP resource
- Uri-parameters
  - Parameters affecting a request constructed from the URI
- Headers
  - Header fields to be included in a request constructed from the URI

SIP Request – Version

- **“SIP/2.0”**

- SIP version is always mandatory
- String is case-insensitive
- But implementations must send upper-case

SIP Response

SIP-Version SP Status-code SP Reason-phrase CRLF
Headers CRLF

```
SIP/2.0 401 Unauthorized

Call-ID: 578222729-4665d775@578222732-4665d772
Cseq: 68 REGISTER
From: <sip.voi18063@sip.cybercity.dk>;tag=903df0a
To: <sip:voi18063@sip:cybercity.dk>
Via: SIP/2.0/UDP
192.168.1.2;received=80.230.219.70;rport=5060;branch=z9hG4bKnp151248737-
46ea715e192.168.1.2
WWW-Authenticate: Digest
realm=”sip.cybercity.dk”,nonce=”...”,opaque=”...”,state=false,algorithm=MD5
Content-Length: 0
```

SIP Response – Status codes

- 3-digit integer result code that indicates the outcome of an attempt to understand and satisfy a request.
- First digit defines the class of response
- Last two digits do not have any categorization role

- 1xx: Provisional – request received, continuing to process the request
- 2xx: Success – the action was successfully received, understood, and accepted
- 3xx: Redirection – further action needs to be taken in order to complete the request
- 4xx: Client error – the request contains bad syntax or cannot be fulfilled at this server
- 5xx: Server Error – the server failed to fulfill an apparently valid request
- 6xx: Global failure – the request cannot be fulfilled at any server

SIP Response – Response phrase

- It’s intended to give a short textual description for the human user

- 100: Trying
- 180: Ringing
- 200: OK
- 202: Accepted
- 400: Bad request
- 500: Server Internal Error

SIP – why insecure?

- Text-based protocol
  - Messages are sent in clear
  - An attacker can collect, modify and replay them
- Integrity not supported
  - Modification and replay attacks are not detected
- Authentication
  - Often not required
  - If present, it’s weak (basic authentication)

SIP Attacks classification

- Denial Of Services
  - INVITE, REGISTER, … flooding
- Extensions enumeration
  - Identify live SIP extensions (brute-force)
- Eavesdropping
  - The art of secretly listening to a VoIP conversation without the consent
- Message tampering
  - Intercepts and modifies SIP messages

SIP Attacks classification (2)

- Registration Hijacking
  - An attacker impersonates a valid UA to a registrar and replaces the legitimate registration with his own address
- Session tear-down
  - It occurs when an attacker observes the signaling for a call, and then sends spoofed SIP “BYE” messages to the participating UAs.

SIP security countermeasures

- Encrypt all the things:
  - SIPS and SRTP
- Authentication
  - Provide strong authentication between SIP components
- Monitor inbound/outbound SIP messages
  - Extensions scanning, malformed messages, malicious teardown requests
- Monitor unusual calling patterns

Suricata SIP parser

- Available since Suricata 5.0
- Written in Rust
- Work over udp/5060
- Basic logging:
  - Request/Response fully parsed and logged
  - Headers parsed but not logged (yet)
  - Extended/custom logging not implemented (yet)
- Sticky buffers:
  - sip.method
  - sip.protocol
  - sip.request_line / sip.response_line
  - sip.stat_code / sip.stat_msg
  - sip.uri

SIP sticky buffers

- **sip.method:** matches on the method found in a SIP request

```
alert sip any any → any any (sip.method; content:"INVITE"; sid:1;)
```

- **sip.uri:** matches on the URI found in a SIP request

```
alert sip any any → any any (sip.uri; content:"sip:sip.cybercity.dk"; sid:1;)
```

- **sip.request_line:** forces the whole SIP request line to be inspected
- `alert sip any any → any any (sip.request_line; content:"REGISTER sip:sip.cyberciti.dk"; sid:1;)`

- **sip.response_line:** forces the whole SIP response line to be inspected
- `alert sip any any → any any (sip.response_line; content:"SIP/2.0 200 OK"; sid:1;)`

SIP sticky buffers (2)

- **sip.stat_code:** matches on the status code found in a SIP response

```
alert sip any any → any any (sip.stat_code; content:"200"; sid:1;)
```

- **sip.protocol:** matches the protocol field from a SIP request or response
- `alert sip any any → any any (sip.protocol; content:"SIP/2.0"; sid:1;)`

SIP attacks detection

- **INVITE flooding**
  - Too many calls?

- `alert sip any any → $SIP_IP $SIP_PORT (msg:”INVITE flooding”;`
  `sip.method; content:”INVITE”;`
  `threshold: type both, track by_src, count 100, seconds 60;`
  `rev:1; sid:1;)`

- **REGISTER flooding**
  - Brute-force?

- `alert sip any any → $SIP_IP $SIP_PORT (msg:”REGISTER flooding”;`
  `sip.method; content:”REGISTER”;`
  `threshold: type both, track by_src, count 100, seconds 60;`
  `rev:1; sid:1;)`

SIP attacks detection (2)

- **Unauthorized responses**
  - A high number of “401 Authorized messages” can be a brute force authentication attack on the SIP proxy
  - Response generated when a user tries to send REGISTER messages with wrong credentials

- ```
  alert sip $SIP_IP $SIP_PORT → any any (msg:”REGISTER brute-force”;
  sip.response_line; content:”SIP/2.0 401 Unauthorized”;
  threshold: type both, track by_src, count 100, seconds 60;
  rev:1; sid:1;)
  ```
  ```

SIP attacks detection (3)

- **SQL injections**

- alert sip any any → $SIP_IP $SIP_PORT (msg:”DROP sql injection”; **sip.request_line; content:”drop”;**
- pcre:”/\’drop/ix”;
  threshold: type both, track by_src, count 100, seconds 60;
  rev:1; sid:1;)

  alert sip any any → $SIP_IP $SIP_PORT (msg:”DELETE sql injection”; **sip.request_line; content:”delete”;**
- pcre:”/\’delete/ix”;
  threshold: type both, track by_src, count 100, seconds 60;
  rev:1; sid:1;)

  alert sip any any → $SIP_IP $SIP_PORT (msg:”SELECT sql injection”; **sip.request_line; content:”select”;**
- pcre:”/\’select/ix”;
  threshold: type both, track by_src, count 100, seconds 60;
  rev:1; sid:1;)

Time for sexy charts

END!

Questions?

- Mail: giuseppe.longo@dreamlab.net
- Twitter: @theglongo
- IRC: glongo @ irc.freenode.org/#suricata