Attacking encrypted VoIP protocols
1. Name> Ivica [Eeveetsa] Stipovic
2. Work> Ward Solutions, Dublin, Ireland
3. Job> Information Security Consultant
4. Contact> Ivica.Stipovic@ward.ie
5. >EOF
What is VoIP protocol?

- VOIP stands for (Voice Over IP) - Voice /Video/messaging that uses IP-based transport protocols for transmission
- What is SIP?
  
  SIP stands for Session Initiation Protocol (it is a voice control protocol), developed by IETF
- SIP is one of the predominant VOIP control protocols

SIP, MGCP, H.323, XMPP…

There are other VoIP protocols as well:

- **MGCP** Media Gateway Control Protocol (MGCP), connection management for media gateways.
- **H.323** - one of the first VoIP call signaling and control protocols that found widespread implementation.
- **XMPP** - Extensible Messaging and Presence Protocol, instant messaging, presence information, and contact list maintenance.
- **Skype protocol**, proprietary Internet telephony protocol suite based on peer-to-peer architecture.

...etc
What applications are using VoIP?

This is a small snapshot of the most popular VoIP applications.

Not all of them use SIP, though: Skype uses proprietary protocol.

Viber does use SIP, so as Blink, Cisco IP Communication, Jitsi, Bria, Empathy, Linphone, X-Lite, Zoiper, Yahoo! Messenger...
Let’s talk about SIP

• SIP by default runs as a UDP service over the port 5060
• By default, it is not encrypted

• Full description of SIP protocol is given in the https://tools.ietf.org/html/rfc3261

• SIP structure is very similar to HTTP session structure (both request and response paradigm)
Structure of a SIP session

Similar to HTTP

Request (REGISTER vs GET)

Response (Unauthorised)
Return code 401

Request (REGISTER vs GET)
Attributes of a SIP session

We will focus on attributes required to facilitate password attacks. These are: algorithm, method, username, realm, uri, response, nonce, {nc}, {cnonce} and {qop}.
Why TLS/encryption?

• wrapping SIP into TLS makes it more secure (HTTP vs HTTPS, POP3 vs POP3S, LDAP vs LDAPS etc.)

• interception of encrypted SIP will show obfuscated application layer payload

• only ip/tcp/udp header level information intelligible
Why TLS/encryption?

-challenges to pentesters that TLS presents – we want the application layer

Lower OSI layers

Application layer

obfuscated
SIP – two aspects of attacks

1\textsuperscript{st} aspect: interception + decryption
- for unencrypted SIP sessions, one focuses only on interception
- in our case, we need to do both

2\textsuperscript{nd} aspect: SIP password cracking
- for unencrypted SIP sessions, off-the-shelf tools available in Kali
- in our case, we need to develop either:
  - manual preparation of a file for sipcrack
  - our own tool to streamline the cracking (<= chosen approach)
Attacking plaintext SIP passwords

-Kali with sipdump and sipcrack

-sipdump takes a pcap of a SIP session as input and generates a text file output for sipcrack

-sipcrack takes the text output from sipdump and performs password dictionary attack
Solution design-1st part (interception and decryption)

An idea of MITM occurred as one plausible attack vector

This is what we need to achieve:

- build a mechanism capable of intercepting and decrypting the TLS wrapped session
- search for some kind of protocol-agnostic proxy capable of decrypting TLS
- forward the traffic from this protocol-agnostic proxy to Burp so we can play with packets
Solution design-1st part (interception and decryption)

- Burpsuite does that job No. 1, but only for HTTP(S) – it does not speak SIP or any other non-HTTP(S) protocols for that matter

- The Solution: mitm_relay.py (https://github.com/jrmdev/mitm_relay)
Solution design-1st part (interception and decryption)

- Topology design developed as below

- Key component: mitm_relay.py + Burpsuite running as SIP proxy on Kali
  - you can have them on separate VMs too, also, no particular need for Kali – any Linux will do, I guess
Chaining mitm_relay.py with Burp

-l 0.0.0.0 mitm_relay.py listens on all interfaces

-p 127.0.0.1:8080 Burp runs on localhost, port 8080

-r tcp:5061:192.168.0.101:5061 relay all traffic on tcp 5061 port to final SIP server running on 192.168.0.101, port 5061

-c mitm.pem digital certificate for mitm_relay.py

-k privatemitm.key private key for mitm_relay.py
Chaining mitm_relay.py with Burp

- If we switch over to Burp, we can see decrypted SIP negotiation

- POST requests have embedded "/CLIENT_REQUEST/to/<IP>" and "/SERVER_RESPONSE/from/<IP>

- that allows Burp to process SIP Sessions (it only transports stuff, does not really speak SIP)
Chaining mitm_relay.py with Burp

• Mechanics of the interception
• SIP client ->mitm_relay.py->Burpsuite->SIP server->SIP client 2

• NOTE: remember that now that we have Burp reading the SIP, several other attacks can be mounted:
  - send SIP request to Burp Repeater,
  - change call destinations,
  - brute force destination numbers,
  - change user agent fingerprint,
  - inject some funky headers/establish covert channel attack,
  - spoof calling ID...

• Tampering in BURP is interesting, but out of scope of this research
Solution design - 2nd part (coding new app)

• Why is new app required – what about sipdump and sipcrack?
  - sipdump will not work here – we have dumped the decrypted session into a non-pcap format

  - sipdump has no way of importing the private key to decrypt the captured wireshark pcap session

  - we can decrypt wireshark pcap, remember? – we possess the private key for mitm_relay.py

• What is new app doing?
  - new app will process the mitm_relay.py dump which is more-less text based file and will extract all authentication attributes and perform password dictionary-based attack
Solution design - 2nd part (coding new app)

- Functional mapping of sipdump+sipcrack into sipcrack2
- Streamline mitm_relay dump parsing with password cracking
- How do we crack digest algorithm?

- https://www.ietf.org/rfc/rfc2069.txt later amended by RFC 2617

- {qop}, {nc}, {cnonce} increase the variability of hashing but will still not protect against our attack and all these attributes can be intercepted and decrypted
Solution design - 2nd part (coding new app)

• Digest authentication algorithm

• If \{qop\}, \{cnonce\} and \{nc\} are not defined (RFC2069) then
  \[ H1 = MD5(\text{username:realm:password}) \]
  \[ H2 = MD5(\text{method:uri}) \]
  \[ \text{Response} = MD5(H1:nonce:H2) \]

Else if \{qop\}, \{cnonce\} and \{nc\} are defined (RFC2617) then

  \[ \text{Response} = MD5(H1:nonce:nc:cnonce:qop:H2) \]

https://github.com/adenosine-phosphatase/sipcrack2
Final thoughts

• Development
  sipcrack2 is for now just a linux version, hope to release Windows version with CUDA/multithreading & parallel processing in a near future

• How realistic/difficult is the attack?
  The attack shown is relatively difficult to implement (ARP poisoning required to redirect the traffic from legitimate proxy to the attacker)

• Recommendations
  - use strong passwords
  - do not use self-signed certificates
  - use client side certificate in addition to server

• Risk rating proposed: Medium
Demo
Questions?

Telling INIT to go to single user mode.
init: rc main process (2205) killed by TERM signal
[root@centos-4 /]# _

Shutting down