Formal modeling of authentication in SIP

Anders Moen Hagalisletto and Lars Strand

Norwegian Computing Center

Outline

1. Voice-over-IP - Session Initiation Protocol (SIP)
2. Our test case scenario
3. Method and tools: Analyzing implementation rather than specification
4. Digest Access Authentication
5. Formal modeling PROSA
6. Results
7. Conclusion and further work.
Voice over IP

- Voice over IP (VoIP) protocols and technology is a merge of telecom and data communication.
- Industry have high focus on VoIP today.
- VoIP is known to be unsecure!
- Multiple attacks on SIP based VoIP exists.
- We will focus on authentication in SIP.
- Norwegian Computing Center evaluates various architectures and protocols of Voice over IP.
  - Session Initiation Protocol (SIP) RFC 3261
  - Interasterisk Exchange IAX (RFC draft only)
VoIP case-study - three protocols: SIP, RTP and IAX

Telecommunication provider

Internet

IAX traffic

firewall

Company network

SIP traffic

Asterisk server
Method

1. Experiment
   - "Don't trust the documentation"
   - Lab test stup: Replicate test scenario.
   - Software: Asterisk PBX and X-Lite softphones.

2. Active observation
   - Using the network monitoring tool "Wireshark".

3. Formal protocol analysis.
   - PROSA
Network tool Wireshark

- A network monitoring tool.
  - Sniff the network
  - Parse the result and compare against the standard.
- Why did we use Wireshark?
  - Compare implementation against the SIP standard.
  - Result used as basis for modelling in PROSA.
Wireshark

Method: REGISTER
Expires: 1800

Authorization: Digest username="alice",realm="asterisk",nonce="3b7a1395",response="ccbd1c3c12b3dca14a4d5e3551"

Authentication Scheme: Digest
Username: "alice"
Realm: "asterisk"
Nonce Value: "3b7a1395"

Digest Authentication Response: "ccbd1c3c12b3dca14a4d5e35519d7"

Authentication URI: sip:NR
Algorithm: MD5
Max-Forwards: 70
User-Agent: X-Lite release 1105d
Content-Length: 0

SIP Digest Authentication Response Value: (sip.auth.digest.response), 34,... | P: 855 D: 26 M: 0
Wireshark
SIP REGISTER

Anders Moen Hagalisletto and Lars Strand (Norwegian Computing Center)

Formal modeling of authentication in SIP

SECURWARE08 9 / 23

Client "Alice"

Server (R)

Request: REGISTER
Status: 100 Trying
Status: 401 Unauthorized
nonce = "3b7a1395"

Request: REGISTER
Status: 100 Trying
Status: 200 OK

time

Compute response using Digest Access Authentication:

HA1 = MD5(username, realm, password)
HA2 = MD5(method, digestURI)
response = MD5(HA1, nonce, nonceCount, clientNonce, qop, HA2)
Why use formal methods?

Because

1. the only way to prove or verify that protocols fulfills their goals!
2. has been used to find new attacks on protocols
3. implicitly gives a unambiguous specification of
   - the protocol’s interactions and entities
   - the functional and security goals
4. the protocol specification can be analyzed automatically
The Dolev Yao model

A Dolev Yao attacker

1. controls the entire network
2. does not have access to secret entities (keys)
3. can intercept any message
4. can send any message (based on her knowledge)

The latter means that it can inject anything into a concrete message, even the entire message content can be changed.
Formal language PROSA contains all necessary primitives and operators for cryptography contains operator: *Agent A believes that* ...

The PROSA tool includes a static validation module

- *automated refinement*
- *validation of refined specs*

simulation and analysis

Note: Both tools and theory rely on the Dolev Yao model.
The PROSA tool - specification of protocols

- Specification Language
  - Protocol Algebra
  - Substitution
  - Matching
  - Generate Latex

- Operational semantics (simulation engine)
  - Standard Semantics
    - Communication
    - Protocol Machine
  - Dolev Yao Semantics
    - Corrupt Communication
    - Attack Machine

- Static Analysis
  - Automated Refinement
  - Validation
  - Static Security Analysis

- Local Deduction
  - Propositional Logic
  - Second order Logic

Anders Moen Hagalisletto and Lars Strand (Norwegian Computing Center)

Formal modeling of authentication in SIP

SECURWARE08
A protocol clause is written:

\[
(P) \quad A \rightarrow B : M
\]

meaning "agent A sends a message M to the agent B"

- \( A, B, C, S, I, I(A) \): agent terms
- \( K_{AB} \): symmetric key shared by A, B
- \( K_A \): A’s public key
- \( K_A^{-1} \): A’s private key
- \( N_A \): nonce generated by agent A
- \( W_A^Y \): string containing the text Y related to agent A
- \( X_A \): miscellaneous entities

Composition operators:
- concatenation of message content denoted by "," (comma),
- hashing \( H[M] \), and
- encryption \( E(K : M) \), where \( K \) is a key and \( M \) a message content.
Digest Access authentication specified precisely

Digest access authentication is then given by

\[
\begin{align*}
H_1 &= H[W_C^\text{uname}, W_C^\text{realm}, K_{CR}^\text{pwd}] \\
H_2 &= H[W_C^\text{meth}, W_C^\text{URI}] \\
\text{response} &= H[H_1, N_R, X_{nc}, N_C, W_qop, H_2]
\end{align*}
\]

Written out explicitly the response yields:

\[
H[H[W_C^\text{uname}, W_C^\text{realm}, K_{CR}^\text{pwd}], N_R, X_{nc}, N_C, W_qop, H[W_C^\text{meth}, W_C^\text{URI}]]
\]

A typical application is then given by a challenger \( R \) requesting a client \( C \) to authenticate as described in the following protocol skeleton:

\[
\begin{align*}
(D_1) \quad R &\rightarrow C : N_R \\
(D_2) \quad C &\rightarrow R : W_C^\text{uname}, W_C^\text{realm}, N_R, W_C^\text{URI}, X_{nc}, N_C, W_qop, H[H_1, N_R, X_{nc}, N_C, W_qop, H_2]
\end{align*}
\]
Establish a (possibly new) contact point,

\[ W^\text{Contact}_C \]

a phone number, email address, etc.
We analyzed the registration sub-protocol in Case Study. SIP authentication on registration =

\[ registration \oplus Digest\ authentication \]

But what exactly means the composition \( \oplus \)?

The exact behaviour not specified explicitly: We used RFC and Wireshark to find out!
Registration with Digest Access authentication (Wireshark)

\( (P^D_1) \quad C \rightarrow R : W^{\text{REGISTER}}, W^C_{\text{Contact}}, N_C^\text{callid} \)

\( (P^D_2) \quad R \rightarrow C : W^{\text{Trying}}, N_C^\text{callid} \)

\( (P^D_3) \quad R \rightarrow C : W^{\text{Unauth}}, W^\text{auth}, W^\text{realm}, N_R, N_C^\text{callid} \)

\( (P^D_4) \quad C \rightarrow R : W^{\text{REGISTER}}, N_C^\text{callid}, W^C_{\text{uname}}, W^\text{realm}, N_R, W^\text{URI}_C, X_{nc}, N_C, W^{qop} \)

\[ H[H[W^\text{uname}_C, W^\text{realm}, K_{CR}^{\text{pwd}}], N_C, X_{nc}, N_R, W^{qop}, H[W^{\text{REGISTER}}, W^\text{URI}_C]] \]

\( (P^D_5) \quad R \rightarrow C : W^{\text{Trying}}, W^C_{\text{Contact}}, N_C^\text{callid} \)

\( (P^D_6) \quad R \rightarrow C : W^{\text{OK}}, W^C_{\text{Contact}}, N_C^\text{callid} \)
Typical Workflow: Analysis of implementation

1. Sniffing network traffic
   Implementation
2. Understand protocol implementation
3. Formal Specification
4. Revision of formal specification
5. Protocol Analyzer Tool
   
   - Attacker
6. Output
   - Analysis report
   - Secure?
7. Revision of implementation
8. Not secure

Anders Moen Hagalisletto and Lars Strand (Norwegian Computing Center)
Formal modeling of authentication in SIP
SECURWARE08
A large picture on the attack
Attack on registration

\[
\begin{align*}
(R_{1.1.a}^D) & \quad C \rightarrow I(R) : W \text{REGISTER}, W_C^{\text{Contact}}, N_C^{\text{callid}} \\
(R_{1.1.b}^D) & \quad I(C) \rightarrow R : W \text{REGISTER}, W_I^{\text{Contact}}, N_C^{\text{callid}} \\
(R_{1.2.a}^D) & \quad R \rightarrow I(C) : W \text{Trying}, N_C^{\text{callid}} \\
(R_{1.2.b}^D) & \quad I(R) \rightarrow C : W \text{Trying}, N_C^{\text{callid}} \\
(R_{1.3.a}^D) & \quad R \rightarrow I(C) : W \text{Unauth}, W_{\text{auth}}, W_{\text{realm}}, N_R, N_C^{\text{callid}} \\
(R_{1.3.b}^D) & \quad I(R) \rightarrow C : W \text{Unauth}, W_{\text{auth}}, W_{\text{realm}}, N_R, N_C^{\text{callid}} \\
(R_{1.4.a}^D) & \quad C \rightarrow I(R) : W \text{REGISTER}, N_C^{\text{callid}}, W_{\text{uname}}, \\
& \quad W_{\text{realm}}, N_R, W_{\text{URI}}, X_{\text{nc}}, N_C, W_{\text{qop}} \\
& \quad H[H[W_{\text{uname}}, W_{\text{realm}}, K_{\text{CR}}^{\text{pwd}}], N_C, X_{\text{nc}}, \\
& \quad N_R, W_{\text{qop}}, H[W \text{REGISTER}, W_{\text{URI}}]] \\
(R_{1.4.b}^D) & \quad I(C) \rightarrow R : W \text{REGISTER}, N_C^{\text{callid}}, W_{\text{uname}}, \\
& \quad W_{\text{realm}}, N_R, W_{\text{URI}}, X_{\text{nc}}, N_C, W_{\text{qop}} \\
& \quad H[H[W_{\text{uname}}, W_{\text{realm}}, K_{\text{CR}}^{\text{pwd}}], N_C, X_{\text{nc}}, \\
& \quad N_R, W_{\text{qop}}, H[W \text{REGISTER}, W_{\text{URI}}]] \\
(R_{1.5.a}^D) & \quad R \rightarrow I(C) : W \text{Trying}, W_I^{\text{Contact}}, N_C^{\text{callid}} \\
(R_{1.5.b}^D) & \quad I(R) \rightarrow C : W \text{Trying}, W_I^{\text{Contact}}, N_C^{\text{callid}} \\
(R_{1.6.a}^D) & \quad R \rightarrow I(C) : W \text{OK}, W_I^{\text{Contact}}, N_C^{\text{callid}} \\
(R_{1.6.b}^D) & \quad I(R) \rightarrow C : W \text{OK}, W_I^{\text{Contact}}, N_C^{\text{callid}} \\
\end{align*}
\]

\[
\begin{align*}
\text{Bel}_C(\text{Bel}_R(\text{Bel}_C(W_{\text{Contact}}^{\text{Contact}}))) & \quad \text{TRUE} \\
\text{Bel}_R(\text{Bel}_C(W_{\text{Contact}}^{\text{Contact}})) & \quad \text{FALSE}
\end{align*}
\]
Contact address of Alice is compromised (attack on authenticity/integrity)

Easy to spot security errors when we have a precise specification

Easy to fix attack in theory:

The attack can be prevented by changing the Digest response to include the contact address(es):

\[ H[H[H[W^\text{name}_C, W^\text{realm}_C, K^\text{pwd}_CR], W^\text{Contact}_C, N_R, X_{nc}, N_C, W^\text{qop}, H[W^\text{REGISTER}_C, W^\text{URI}_C]]] \]

Hence: the specification must be changed!
SIP is a huge and feature-rich protocol standard

But SIP REGISTRATION Digest authentication = leads to REGISTRATION attack

This attack can be prevented by modifying the Digest.

Formalizing protocols with tools support aids in discover new attacks

Future work: Deploy same procedure for IAX protocol - compare SIP and IAX