Tamarin Workshop
Automated Protocol Verification

Felix Linker
PhD Student, ETH Zurich
Who are we?

Felix Linker
Department of Computer Science
ETH Zürich

https://felixlinker.de
@felixlinker

Alexander Dax
CISPA Helmholtz Center for Information Security

https://alexanderdax.com

Jonathan Hoyland
Cloudflare
Part 1: An Introduction to Tamarin
The EMV Standard: Break, Fix, Verify

- S&P21 paper showed how to:
  - Pay with stolen credit card
  - Without ever needing the PIN
Attack Video
The EMV Standard: Break, Fix, Verify

• S&P21 paper showed how to:
  – Pay with stolen credit card
  – Without ever needing the PIN
• How did they find this attack?
• Used Tamarin!
What is Tamarin?

• Our world is powered by security-critical protocols
  – You want certain things to not happen
    – *NSA reads your WhatsApp messages*
  – You want certain things to always happen
    – *Merchant receives payment upon confirmation*

• Protocols are complex!

• People make mistakes!

*With Tamarin, you can prove that a protocol (model) guarantees security properties*
Workshop Goals

1. Go to github.com/felixlinker/tamarin-workshop/
2. Clone or download
3. Install Tamarin

- Get your hands on Tamarin
- Tamarin is easy! (except when it isn’t)
Example: TCP

3-Way TCP Handshake
Example: TCP – What happens under the hood?

rule SYN:
[ ]
-- [ Begin() ] -->
[ St_AliceWait(), Out('SYN') ]

rule SYNACK:
[ In('SYN') ]
--> 
[ St_BobWait(), Out('SYNACK') ]
Example: TCP – What happens under the hood?

rule SYN:
[]
--[ Begin() ]--> [ St_AliceWait(), Out('SYN') ]

rule SYNACK:
[ In('SYN') ]
--> [ St_BobWait(), Out('SYNACK') ]
Example: TCP – What happens under the hood?

rule SYN:
[ ]
--[ Begin() ]->
[ St_AliceWait(), Out('SYN') ]

rule SYNACK:
[ In('SYN') ]
--> 
[ St_BobWait(), Out('SYNACK') ]
Example: TCP – What happens under the hood?

```
rule SYN:
    []
    --[ Begin() ]->
    [ St_AliceWait(), Out('SYN') ]

rule SYNACK:
    [ In('SYN') ]
    --[
    St_BobWait(), Out('SYNACK') ]
```

In('SYN')
Example: TCP – What happens under the hood?

\[
\text{rule SYN:} \\
\quad [] \\
\quad \quad \rightarrow \quad \text{[ Begin() ]} \rightarrow \quad \text{[ St_AliceWait(), Out('SYN') ]}
\]

\[
\text{rule SYNACK:} \\
\quad [ \text{In('SYN')} ] \\
\quad \rightarrow \quad \text{[ St_BobWait(), Out('SYNACK') ]}
\]

St_AliceWait()
Example: TCP – What happens under the hood?

```
rule SYN:
    []
    --[ Begin() ]->
    [ St_AliceWait(), Out('SYN') ]

rule SYNACK:
    [ In('SYN') ]
    -->
    [ St_BobWait(), Out('SYNACK') ]
```
Values in Tamarin

- Values can be:
  - Constants: ‘constant’
  - Unguessable (fresh) values: ~k
  - Public values: $P$
  - Function application: f(t1, t2)

- A variable $x$ can be any of the above (also called message)

- **Equational theory** gives symbols semantics

  \[
  \text{functions: sign/2, verify/3, pk/1, true/0} \\
  \text{equations: verify(sign(m, sk), m, pk(sk)) = true}
  \]
Take-Aways

Cheatsheet!

Exercise 1 + 2!

functions: $f/1$

rule Memorize:

\[
\text{MemorizeSomething}(f(x)) \quad \rightarrow \quad \text{Out}(v)
\]

rule LookUpAndSend:

\[
\text{MemorizeSomething}(v) \quad \rightarrow \quad \text{Out}(v)
\]

Pattern-match

State read

Message in

State write

Message out

Fact

Function

Pattern-match

Department of Computer Science
Summary – Part 1

• So far you learned
  – Modelling in Tamarin
  – State-read/message-in + state-write/message-out pattern
  – The symbolic model

• Interested in more? Documentation is quite good

• Also:
  – Manual proofs
  – Custom proof heuristics
  – Induction
Part 2: Analyzing Specifications with Tamarin
What is Tamarin?

- Our world is powered by security-critical protocols
  - You want certain things to not happen
    - \textit{NSA reads your WhatsApp messages}
  - You want certain things to always happen
    - \textit{Merchant receives payment upon confirmation}
- Protocols are complex!
- People make mistakes!

\begin{tamarin}
\textbf{With Tamarin, you can prove that a protocol (model) guarantees security properties}
\end{tamarin}
What is Tamarin?

• Our world is powered by security-critical protocols
  – You want certain things to not happen
    – *NSA reads your WhatsApp messages*
  – You want certain things to always happen
    – *Merchant receives payment upon confirmation*
• Protocols are complex!
• People make mistakes!

*With Tamarin, you can prove that a protocol (model) guarantees security properties*
What is Tamarin?

• Our world is powered by security-critical protocols
  – You want certain things to not happen
    – *NSA reads your WhatsApp messages*
  – You want certain things to always happen
    – *Merchant receives payment upon confirmation*

• Protocols are complex!
• People make mistakes!

*With Tamarin, you can prove that a protocol (model) guarantees certain security properties under certain assumptions*
Specifications vs Formal Analysis

**Specification**
- Designed to foster compatible implementations
- Often deliberately underspecified
- Security considerations often ad-hoc

**We did X so attack Y is not possible**

**Formal Analysis**
- A structured way to approach security
- A positive definition of security properties
- A list of explicit assumptions

**Ideally…**
Case Study: OAuth 2.0

10. Security Considerations

10.1. Client Authentication

10.2. Client Impersonation

10.3. Access Tokens

10.4. Refresh Tokens

10.5. Authorization Codes

10.6. Authorization Code Redirection URI Manipulation

10.7. Resource Owner Password Credentials

10.8. Request Confidentiality

10.9. Ensuring Endpoint Authenticity

10.10. Credentials-Guessing Attacks

10.11. Phishing Attacks

10.12. Cross-Site Request Forgery

10.13. Clickjacking


10.15. Open Redirectors

Case Study: OAuth 2.0 – Prior Work

Fett, Küsters, Schmitz. CCS’16.

- But: Also doesn’t list desired properties
Case Study: OAuth 2.0 – But how analyze a specification?

1. Implement an initial specification
2. Model security properties
   - It’s okay if they are trivially true
3. Make your model more realistic
   - Now the properties are hopefully false
4. Refine everything
   - Let your understanding guide you
   - Let Tamarin tell you why your understanding is wrong

Use the GUI
But how analyze a specification?
But how analyze a specification?

**Lemma SecrecyMessage:**

"All m #t. MessageWasSent(m) @ #t

==\not\exists x. K(m) @ #x"

Message was sent...

...and adversary knows it.
But how analyze a specification?

Adversary knows key! We expect this to be impossible!

Questions?
Case Study: OAuth 2.0 – Authorization Code Flow

Let’s do it!
Further Reading
