Métodos Formais Aplicados a Segurança da Informação

Uma Pequena Introdução

Jean Everson Martina, Ph.D.

Laboratório de Segurança em Computação
Universidade Federal de Santa Catarina

jean.martina@gmail.com

2013

LabSEC
Laboratório de Segurança em Computação

Sumário

Introdução

Introdução

Podemos Tornar as Rede Seguras através do uso da Criptografia?

- Objetivos:
  - Autenticidade: Quem enviou?
  - Sigilo: Quem pode receber?
- Ameaças:
  - Um atacante ativo
  - Agentes descuidados ou comprometidos
  - Não consideramos a força dos algorítimos

Notação

A, B	Nome dos Agentes (Alice e Bob)
N_a	Numero aleatório escolhido por Alice (Nonce)
K_a	Chave Pública da Alice
$\{\|X\|\}_{K_a}$	Mensagem cifrada usando Ka
	• Qualquer pessoa pode cifrar
	• Somente Alice pode recuperar X

Um Pequeno Protocolo

O Protocolo Needham-Schroeder

Apresentação

1. A $\rightarrow$ B: $\{|N_a, A|\}_{K_b}$
   Alice manda pra Bob um Nonce cifrado
2. B $\rightarrow$ A: $\{|N_a, N_b|\}_{K_a}$
   Bob retorna para Alice o Nonce dela junto com o Nonce dele
3. A $\rightarrow$ B: $\{|N_b|\}_{K_b}$
   Alice Retorna pra Bob o Nonce dele

O Protocolo Needham-Schroeder

Explicação

1. A $\rightarrow$ B: $\{|N_a, A|\}_{K_b}$
   Somente Alice conhece Na antes da Mensagem 1
   Somente Bob pode decifrar a Mensagem 1
2. B $\rightarrow$ A: $\{|N_a, N_b|\}_{K_a}$
   Somente Bob conhece Nb antes da Mensagem 2
   Bob conhece Na porque ele pode decifrar
   Somente Alice pode decifrar a Mensagem 2
3. A $\rightarrow$ B: $\{|N_b|\}_{K_b}$
   Alice conhece Nb porque ela pode decifrar
   Somente Bob pode decifrar a Mensagem 3
   Pra que a mensagem 3?

LabSEC
Laboratório de Segurança em Computação

O Protocolo Needham-Schroeder

Interpretação

1. A $\rightarrow$ B: $\{|N_a, A|\}_{K_b}$
   Alice inicia a sessão, Na é o controle da sessão
   A identidade de Alice (A) serve para Bob saber pra quem cifrar a mensagem 2
2. B $\rightarrow$ A: $\{|N_a, N_b|\}_{K_a}$
   Bom manda Na de volta para manter a sessão
   Bob Nb para poder autenticar Alice na mensagem 3
   Ao receber a Mensagem 2 Alice sabe que só Bob poderia te-la criado porque ela contém Na
3. A $\rightarrow$ B: $\{|N_b|\}_{K_b}$
   Alice ja autenticou Bob. Agora ela quer se autenticar para Bob
   Ao receber a Mensagem 3 Bob sabe que so Alice poderia te-la criado porque ela contém Nb

LabSEC
Laboratório de Segurança em Computação

O Protocolo Needham-Schroeder

Obtenção dos Objetivos

1. A $\rightarrow$ B: $\{|N_a, A|\}_{K_b}$
2. B $\rightarrow$ A: $\{|N_a, N_b|\}_{K_a}$
3. A $\rightarrow$ B: $\{|N_b|\}_{K_b}$

- O Protocolo autentica Alice para Bob
- O Protocol autentica Bob para Alice
- Pelo uso de Nonces novos (fresh), obtemos a propriedade de que a outra parte esta viva no protocolo
- Mas e ai? Isso é seguro?

Um Grandioso Ataque

Regras do Jogo

Modelo de ameaça

- Charlie é um atacante muito poderoso.
- Ele pode:
  - Interceptar qualquer coisa na rede
  - Bloquear qualquer coisa na rede
  - Repetir Mensagens
  - Forjar mensagens como tudo que ele aprendeu monitorando a rede
  - Se comportar como um agente normal
- Ele não pode:
  - Quebrar algoritmos criptográficos
  - Adivinhar números aleatórios

O Protocolo Needham-Schroeder

O ataque de Gavin Lowe

1. $A \rightarrow C: \{|N_a, A|\}_{K_c}$
1'. $C(A) \rightarrow B: \{|N_a, A|\}_{K_b}$
2'. $B \rightarrow C(A): \{|N_a, N_b|\}_{K_a}$
2. $C \rightarrow A: \{|N_a, N_b|\}_{K_a}$
3. $A \rightarrow C: \{|N_b|\}_{K_c}$
3'. $C \rightarrow B: \{|N_b|\}_{K_b}$

- Bob acredita estar falando com Alice, quando na verdade está falando com Charlie.
- Charlie usa Alice como um oráculo para responder aos desafios de Bob.
- Charlie pode usar Nb para provar para Bob que ele é Alice

O ataque de Gavin Lowe

Fatos

- Gavin Lowe era um "mero" professor de teoria da computação em Oxford
- Parece fácil, mas levaram 15 anos para descubrir este ataque.
- O ataque funciona porque o modelo de ameaça é mais forte.
- Mas o ataque só foi descoberto através do uso de um método formal, neste caso um checador de modelos.

Metodos Formais Aplicados a Segurança

Pequena Revisão de Lógica

Antes de falar de métodos formais vamos revisar um pouco de lógica matemática

- Lógica Proposicional
- Lógica de Primeira Ordem (FOL)

Lógica Proposicional

- É o system lógico maps basico. Ele estuda os argumentos e sua estrutura.
  - Um argumento é uma sentença declarativa em linguagem natural (ex. Português)
  - Por exemplo: *"O ônibus está atrasado"*
- Foi descoberta por Aristóteles na Grecia Antiga.
- Cada Sentença recebe um valor de V (verdadeiro) ou F (falso).
- Existem regras bem definidas para extrair significado de argumentos complexos. (Modus Ponens, Modus Tolens, Negação da Implicação entre outros).
- É uma lógica clássica e fácil de entender.

Um exemplo de Fórmula em Lógica Proposicional

Está chovendo.	:	P
Jane tem seu guarda-chuva consigo.	:	Q
Jane se molha.	:	R

$$(P \wedge \neg Q) \rightarrow R, \neg R, P \vdash Q$$

- **Símbolos**: $\wedge$ – "e"; $\vee$ – "ou"; $\not$ "não"; $\rightarrow$ – "implica"; $\leftrightarrow$ – "equivalente a"; $\vdash$ – "prova"; and $\nvdash$ – "não prova".

LabSEC
Laboratório de Segurança em Computação

Lógica de Primeira Ordem (FOL)

- Também conhecida como *lógica de predicados* ou *lógica quantificacional*.
- Estende a expressividade da lógica proposicional.
  - É difícil expressar sentenças como *"alguma coisa* **é** *uo* **têm** *..."* em Lógica Preposicional.
- A grande diferença para com a Lógica Preposicional é a existência de quantificadores:
  - $\exists$ (existe), and $\forall$ (para todos).
- Outros conceitos são: predicados, variáveis, funções e constantes.
- Essa lógica é expressiva o suficiente para modelarmos protocolos

Um exemplo de uma Fórmula FOL

$S(x, y)$: x é filho de y (S é um predicado)
$B(x, y)$: x é irmão de y (B é um outro predicado)
$f(x)$: retorna o pai de x (f é uma função)

$$\forall x[S(x, f(m) \rightarrow B(x, m))]$$

(m é uma constante, e x é uma variável)

Nossos protocolos podem ser modelados desta forma.

LabSEC
Laboratório de Segurança em Computação

Definindo os Predicados

- $E(x)$: x é uma entidade (um agente) no protocolo.
- $Stores(x, y)$: o dado x é armazenado pela entidade y.
- $Knows(x, y)$: o dado x é conhecido pela entidade y.
- $M(x)$: a mensagem x é enviada no protocolo.

Definindo as Funções

- Agrupamento de mensagens:
  - $pair(x, y)$; $triple(x, y, z)$.
- Troca de mensagens:
  - $sent(x, y, z)$: o agente x envia ao agente y a mensagem z.
- Funções de chave:
  - $krkey(x, y)$: a chave privada x pertence ao agente y;
  - $kukey(x, y)$: a chave pública x pertence ao agente y; e
  - $kp(x, y)$: a chave privada x e a chave pública y formam um par de chaves.

Definindo as Funções

- Funções de *Nonce*:
  - $nonce(x, y)$: o *nonce* x é gerado pela entidade y.
- Primitivas criptográficas:
  - $encr(x, y)$: o dado x é cifrado usando a chave y; e
  - $sign(x, y)$: o dado x é assinado usando a chave y.

Definindo as Constantes

- Agentes participantes do protocolo:
  - *a* (Alice); *b* (Beto); *c* (Charlie).
- Chaves privadas e chaves públicas:
  - *kra*; *kua*: chave privada e chave pública de Alice
  - *krb*; *kub*: chave privada e chave pública de Beto
  - *krc*; *kuc*: chave privada e chave pública de Charlie
- *Nonces*:
  - *na*; *nb*; *nc*.

Definindo a Base de Conhecimento Inicial

- O primeiro passo é definir o conhecimento pertencente a cada agente. Por exemplo, parte do conhecimento inicial de Alice é:

Exemplo

$E(a)$
$Knows(kp(krkey(kra, a), kukey(kua, a)), a)$
$Knows(kukey(kub, b), a)$
$Knows(kukey(kuc, c), a)$
$Knows(nonce(na, a), a)$

A mesma coisa se faz para os outros agentes Beto e Charlie

LabSEC
Laboratório de Segurança em Computação

Descrevendo o Protocolo

- Em seguida, modelamos cada passo da troca de mensagens. Por exemplo, o primeiro passo é modelado da seguinte forma:

Exemplo

$$
\begin{aligned}
&Knows(kukey(kua, a), a) \wedge \\
&Knows(kp(krkey(kra, a), kukey(kua, a)), a) \wedge \\
&Knows(kukey(kub, b), a) \wedge \\
&Knows(nonce(na, a), a) \\
&\rightarrow \\
&M(sent(a, b, encr(pair(na, a), kub)) \wedge \\
&Stores(pair(na, b)a)
\end{aligned}
$$

Descrevendo o Protocolo

- O segundo passo:

Exemplo

$$\forall x[Knows(kukey(kub, b), b) \wedge$$
$$Knows(kp(krkey(krb, b), kukey(kub, b)), b) \wedge$$
$$Knows(kukey(kua, a), b) \wedge$$
$$Knows(nonce(nb, b), b) \wedge$$
$$M(sent(x, b, encr(pair(na, a), kub))$$
$$\rightarrow$$
$$M(sent(b, a, encr(pair(na, nb), kua)) \wedge$$
$$Stores(pair(nb, a), b)]$$

Descrevendo o Protocolo

- O terceiro passo:

Exemplo

$$\forall x[$$
$$Stores(pair(na, b), a) \wedge$$
$$M(sent(x, a, encr(pair(na, nb), kua))$$
$$\rightarrow$$
$$M(sent(a, b, encr(nb), kub))]$$

Modelo Lógico do Atacante

- O modelo do atacante adiciona alguns elementos lógicos:
  - A constante c que representa o próprio atacante;
  - Os dados do atacante ao personificar um usuário válido no protocolo; e
  - O predicado $Im(x)$ que indica o conhecimento aprendido pelo atacante pela manipulação das mensagens trocadas. Este predicado possui funcionamento idêntico ao predicado $M(x)$.

Conhecimento Inicial

1. O atacante é uma entidade no protocolo e tem seus dados à sua própria disposição:
   - $E(c)$
2. Conhece os dados públicos dos agentes legítimos:
   - $Knows(kukey(kua, a), c)$
   - $Knows(kukey(kub, b), c)$
3. Pode gravar todas as mensagens:
   - $\forall x, y, w[M(sent(x, y, w)) \rightarrow Im(w)]$

Transformações de Mensagens

1. Pode decompor mensagens em pedaços menores:
   - $\forall u, v[Im(pair(u, v)) \rightarrow Im(u) \wedge Im(v)]$
   - $\forall u, v, w[Im(triple(u, v, w)) \rightarrow Im(u) \wedge Im(v) \wedge Im(w)]$
2. Pode fabricar mensagens a partir do conteúdo aprendido:
   - $\forall u, v[Im(u) \wedge Im(v) \rightarrow Im(pair(u, v))]$
   - $\forall u, v, w[Im(u) \wedge Im(v) \wedge Im(w) \rightarrow Im(triple(u, v, w))]$
3. Pode enviar mensagens falsas:
   - $\forall u, x, y[Im(u) \wedge E(x) \wedge E(y) \rightarrow M(sent(x, y, u))]$

Capacidades Criptográficas

1. Qualquer coisa pode potencialmente ser uma chave:
   - $\forall u, v[Im(u) \wedge E(v) \rightarrow Knows(krkey(u, v), c)]$
   - $\forall u, v[Im(u) \wedge E(v) \rightarrow Knows(kukey(u, v), c)]$
2. Qualquer coisa pode potencialmente ser um *nonce*:
   - $\forall u, v[Im(u) \wedge E(v) \rightarrow Knows(nonce(u, v), c)]$
3. Gerar mensagens cifradas ou assinadas com as chaves conhecidas:
   - $\forall u, v, x[Im(u) \wedge Knows(kukey(v, x), c) \wedge E(x) \rightarrow Im(encr(u, v))]$
   - $\forall u, v, x[Im(u) \wedge Knows(krkey(v, x), c) \wedge E(x) \rightarrow Im(sign(u, v))]$

LabSEC
Laboratório de Segurança em Computação

Mais em Capacidades Criptográficas

1. Decifrar mensagens com chaves conhecidas:
   - $\forall u, v, w, x[Im(encr(u, v)) \wedge Knows(kp(krkey(w, x), kukey(v, x)), c) \wedge E(x) \rightarrow Im(u))]$
2. Decifrar mensagens com *nonces* conhecidos:
   - $\forall u, v, w[Im(encr(u, v)) \wedge Knows(nonce(v, w), c) \wedge E(w) \rightarrow Im(u))]$
3. Ter acesso ao conteúdo das mensagens assinadas:
   - $\forall u, v[Im(sign(u, v)) \rightarrow Im(u)]$

O Provador de Teoremas SPASS

- A busca por provas pode ser feita manualmente, com papel e caneta.
- Porém, um modo mais conveniente (e prático) é o uso de provadores de teoremas como suporte à obtenção das provas.
- O provador de teoremas escolhido foi o SPASS.
  - Lida com Lógica de Primeira Ordem.
  - Prova por contradição (negação da conjectura).
  - Provador de propósito geral.

Testando os Modelos Lógicos

- O teste de conjecturas sobre os modelos lógicos criados, permite a extração de fatos sobre nosso protocolo.
  - Conjecturas são afirmações que não sabemos se são verdadeiras ou falsas a partir dos axiomas (premissas).
  - Por sua vez, fatos são as afirmações extraídas a partir do teste de conjecturas.
- O ataque de Lowe pode ser facilmente verificado nesta especificação formal

Resultados e Objetivos

Resultados até agora no LabSEC

- Verificação dos protocolos da Nota Fiscal Eletrônica usando FOL
- Verificação de Protocolos de Autenticação Biométrica usando FOL
- Verificação de Protocolos de Multicast usando HOL

LabSEC
Laboratório de Segurança em Computação

Objetivos

- Desenvolver uma comunidade de métodos formais para a segurança
- Dar mais garantias aos protocolos amplamente usados
- Formar pessoas capacitadas na arte:
  - Alta empregabildade: Intel, Arm, Nvidia, Microsoft, etc usam as mesmas técnicas para outros problemas.

LabSEC
Laboratório de Segurança em Computação

Projetos Futuros

- Atacar protocolos usando técnicas mais elaboradas em lógicas com mais expressividade (HOL)
- Gerar métodos de verificação formal de iteração humana com protocolos

LabSEC
Laboratório de Segurança em Computação

Agradecimentos

- Eduardo dos Santos, M.Sc. pela grande ajuda na confecção dos slides e na execução de experimentos.

LabSEC
Laboratório de Segurança em Computação