How to prevent type-flaw guessing attacks on password protocols

Sreekanth Malladi and Jim Alves-Foss
University of Idaho, USA.
June 22, 2003

Password protocols

Protocols using passwords for authentication and key establishment.

Guessing attacks: Introduction

$$n_a$$

$$\{n_a\}_{passwd(a,b)}$$

Guesses: `monday, tuesday, wednesday`, ...

$$\{n_a\}_{\texttt{monday}}$$
$$\{n_a\}_{\texttt{tuesday}}$$
$$\{n_a\}_{\texttt{wednesday}} \cdots$$

Type-flaws

Data of one type, later interpreted as different type.

Examples: na as ka, $\{nb, a\}_k$ as na etc.

Type-glaw guessing attacks

Guessing attacks using type-flaws. i.e.

1. Induce type-flaws in *on-line* communication;

2. Verify a guess *off-line* using the messages.

Continued ...

Example:

$\mathbf{P1}$ (Lomas et al.'s protocol [GLNS93]):

Msg 1. $A \rightarrow B$: $\{C, N\}_{pk(B)}$
Msg 2. $B \rightarrow A$: $\{f(N)\}_{PAB}$.

C - confounder, N - integer, f - invertible function, PAB - $passwd(A, B)$.

$\mathbf{P2}$:

Msg 1. $A \rightarrow B$: $\{N, C\}_{pk(B)}$
Msg 2. $B \rightarrow A$: $\{f(N)\}_{PAB}$.

$\mathbf{P1}$ and $\mathbf{P2}$ are combined:

On-line phase:

Msg P1.1. $a \rightarrow b$: $\{c, n\}_{pk(b)}$
Msg P1.2. $b \rightarrow a$: $\{f(n)\}_{pab}$
Msg P2.1. $I(a) \rightarrow b$: $\{c, n\}_{pk(b)}$
Msg P2.2. $b \rightarrow I(a)$: $\{f(c)\}_{pab}$.

Off-line phase:

guess pab,
decrypt $\{f(n)\}_{pab}, \{f(c)\}_{pab}$
encrypt (c, n) with $pk(b)$
verify.

Heather et al.'s solution:

- Tag each field with it's type;
- eg. na as (nonce, na);
- Problem when used in password protocols: Tags verify guess directly!!
- eg. $\{na\}_{passwd(a,b)}$ as $\{\text{nonce}, na\}_{passwd(a,b)}$;
- Decrypt with guess, check for "nonce";
- If found, verifies the guess rightaway!!
- Conclusion: Heather et al.'s solution *cannot* be used in password protocols.

Hypothesis:

> *Avoiding* tags inside password encryptions prevent most type-flaw guessing attacks.

Proof Strategy:

Aim:

> Protocol is secure without type-flaws $\Rightarrow$ Protocol secure under tagging

Therefore, prove that

> Attack on tagging scheme $\Rightarrow$ Attack when all fields correctly tagged.

Proof parts

1. On-line communication;

2. Off-line guessing and verification.

Part 1:

Using Heather et al.’s results in [HLS00].

Part 2:

Using our def of guessing attacks [CMAE03].

Protocol model

- Based on Heather et al.'s [HLS00];
- *Message structure* – Tags, Facts and Taggedfacts.
- Tags – agent, nonce, . . . ;
- Facts – *Atom, Pairs, Encryptions*;
- TaggedFacts – (tag,fact);
- Well-tagged fact – Tag is *indeed* the type of the fact.

Protocol model – Continued ...

- Our change: Treat password encryptions as subset of *Atoms*;

- i.e. Password encryptions as an “abstract type”;

- Possible because we disallow attacker operations on them;

- Do not consider password learnt by breaching secrecy;.

- *Framework* – Strand spaces

  1. *Strands* – Communications of honest agent or penetrator;

  2. *Bundle* – Partial or complete protocol run.

- *Honest strands* – Modelled using “strand templates”;

- Strand templates output honest strands after instantiation.

- *Penetrator strands* – Dolev-Yao attacker with standard inference rules.

Transforming arbitrarily tagged bundles to well-tagged bundles

- Define a renaming function ϕ;

- ϕ changes an aribtrary bundle C to a well-tagged bundle;

- Possible because, if honest agent accepts ill-tagged fact, it should accept any value in it's place;

- Show if s is an honest strand, so is $\phi(s)$ (from [HLS00,Lemma 3.2]);

- Show if s is a penetrator strand, so is $\phi(s)$ (from [HLS00,section 3.3]).

Part 1: Results in [HLS00,Theorem 1]

If C is a bundle (under the tagging scheme) then there is a renaming function ϕ and a bundle C'', such that:

- C'' contains the tagged facts of C (considered as a set), renamed by ϕ;
- C'' contains the same honest strands as C, modulo some renaming;
- facts are uniquely originating in C'' if they were uniquely originating in C;
- all tagged facts in C'' are well-tagged.

Part 2: Guessing attacks

Off-line attacker capabilities:

- Use a guess to encrypt and decrypt password encryptions;
- Split and concatenate facts;
- Tag facts and untag taggedfacts.
- Given bundle C and taggedfact tf;
  - Define $\models$ on C and tf such that $C \models tf$ if,
  - There exist a valid sequence of attacker actions to produce tf from C;

Defining guessing attacks

- Attacker must synthesize a term in two ways using a guess;

- But in *atmost* one way without using the guess;

Formally,

Definition 1. g is *verifiable* from C and tf is a *verifier* for g iff:

1. $\widehat{C} \cup \{g\} \models tf \ \wedge \ \widehat{C} \cup \{g\} \models \widehat{tf}$; and

2. $\widehat{C} \not\models tf \vee \widehat{C} \not\models \widehat{tf}$.

where $\widehat{tf}$ is a fresh constant and $\widehat{C}$ is obtained by replacing the particular occurrence of tf in C, with $\widehat{tf}$.

Lemma 1.

$$C \cup \{g\} \models_{tr} tf \Rightarrow C^{''} \cup \{g\} \models_{\phi(tr)} \phi(tf).$$

i.e attacker can derive a term from C'' if the corresponding term is derivable from C.

Corollary 1.

$$C \not\models tf \Rightarrow C^{''} \not\models \phi(tf).$$

i.e. attacker *cannot* derive a term from C'' if the corresponding term is not derivable from C.

Main result

- Let C'' be denoted as $\mathcal{C}$.

- If guessing attack on C, then,

  1. $\widehat{C} \cup \{g\} \models tf \;\wedge\; \widehat{C} \cup \{g\} \models \widehat{tf}$; and

  2. $\widehat{C} \not\models tf \vee \widehat{C} \not\models \widehat{tf}$.

- Rewrite above expressions,

  1′. $\widehat{\mathcal{C}} \cup \{g\} \models tf' \;\wedge\; \widehat{\mathcal{C}} \cup \{g\} \models \widehat{tf}'$; and

  2′. $\widehat{\mathcal{C}} \not\models tf' \vee \widehat{\mathcal{C}} \not\models \widehat{tf}'$.

- Possible because of Lemma 1 and Corollary 1;

- Therefore, attack on $C \Rightarrow$ attack on C''.

Conclusion

1. Type-flaws can be used to launch guessing attacks;

2. Most of them can be prevented by type-tagging;

3. Proof consisted of two parts;

4. Similar on-line communication is possible on two protocol runs with and without type-flaws;

5. Above point follows from Heather et al.'s results;

6. Corresponding guessing attack on both or on none;

7. Indirectly proves attack not due to type-flaws.

Future work

1. Limitation: replaying $\{t_1\}_{passwd(a,b)}$ in $\{t_2\}_{passwd(a,b)}$.

2. Tagging can be simplified – just use component numbers inside encryptions;

3. Proving "protocol-ids" inside strong encryptions prevent multi-protocol guessing attacks;

4. Effects of the solutions on secrecy and authentication;

5. Effects when using non-standard inference rules;

6. Decidability of guessing attacks (tagging helps);

7. Limitations of Heather et al.'s and our proofs:

   (a) Do not consider all possible constructed keys (only sequence of atoms);

   (b) Message elements without algebraice properties (eg. XOR and products);

   (c) Above two required for "real-world" protocols.

Acknowledgments

Paper dedicated to Late Professor Roger Needham.

Thanks to

1. Iliano Cervesato;

2. Anonymous referees;

3. Ricardo Corin;

4. Vitaly Shmatikov; and

YOU ALL !!!

Bibliography

CMAE03 R. Corin, S. Malladi, J. Alves-Foss and S. Etalle. Guess what? Here is a new tool that finds some new guessing attacks. *Workshop on Issues in the Theory of Security (WITS'03)*, Warsaw, March 2003.

GLNS93 L. Gong, T. M. Lomas, R. M. Needham, and J. H. Saltzer. Protecting poorly chosen secrets from guessing attacks. *IEEE Journal on Selected Areas in Communications*, 11(5):648–656, 1993.

HLS00 J. Heather, G. Lowe, and S. Schneider. How to prevent type flaw attacks on security protocols. In *Proceedings, 13th Computer Security Foundations Workshop*, 255-268, , July 2000.