Criptografie și Securitate

- Prelegerea 16.1 -
Atacuri de timp pentru MAC-uri

Adela Georgescu, Ruxandra F. Olimid

Facultatea de Matematică și Informatică
Universitatea din București

Cuprins

Atac de timp

- Prezentăm un atac general care afectează multe implementări ale algoritmilor MAC;

Atac de timp

- Prezentăm un atac general care afectează multe implementări ale algoritmilor MAC;
- Studiem verificarea unui HMAC implementată în libraria criptografică KeyCzar (Python);

Atac de timp

- Prezentăm un atac general care afectează multe implementări ale algoritmilor MAC;
- Studiem verificarea unui HMAC implementată în libraria criptografică KeyCzar (Python);
- Codul de mai jos verifică un tag generat de HMAC

```
def Verify(key,msg,tag_bytes):
    return HMAC(key,msg) == tag_bytes
```

Atac de timp

- Prezentăm un atac general care afectează multe implementări ale algoritmilor MAC;
- Studiem verificarea unui HMAC implementată în libraria criptografică KeyCzar (Python);
- Codul de mai jos verifică un tag generat de HMAC

```
def Verify(key,msg,tag_bytes):
    return HMAC(key,msg) == tag_bytes
```

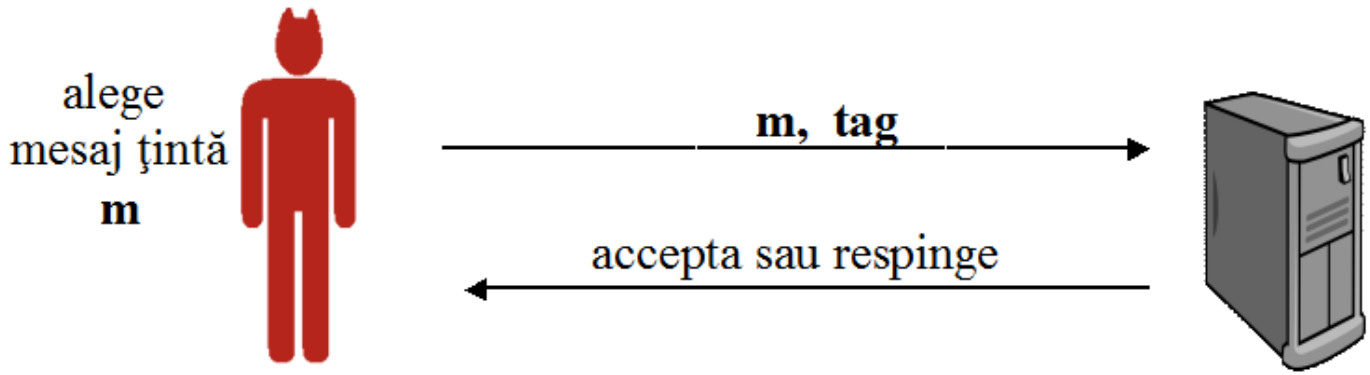
- Problemă: comparația "==" se face octet cu octet și întoarce *false* atunci când prima inegalitate este găsită;

Atac de timp

- Scopul atacului: Adversarul $\mathcal{A}$ vrea să calculeze un tag pentru un anume mesaj m

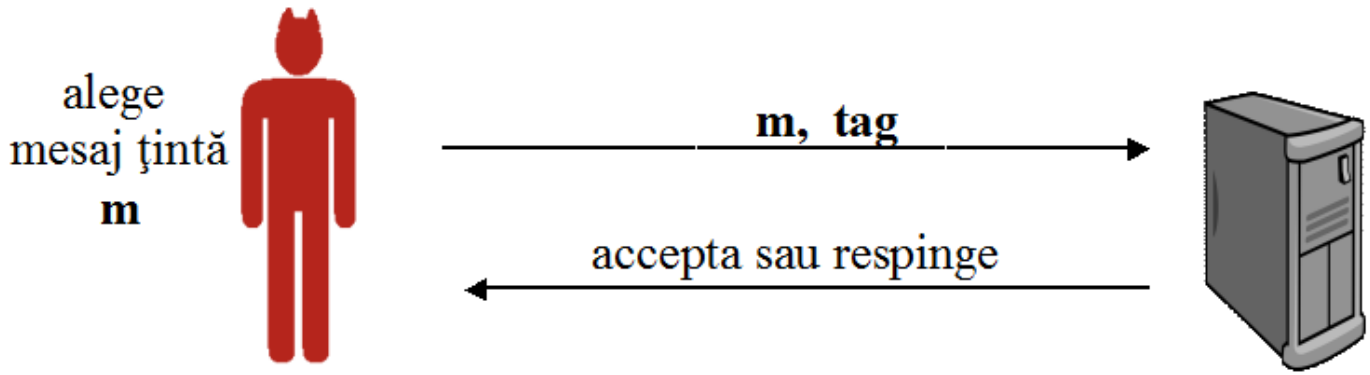
Atac de timp

- Scopul atacului: Adversarul $\mathcal{A}$ vrea să calculeze un tag pentru un anume mesaj m



Atac de timp

- Scopul atacului: Adversarul $\mathcal{A}$ vrea să calculeze un tag pentru un anume mesaj m



- $\mathcal{A}$ trimite multe cereri către server pentru același mesaj m și tag-uri diferite.

Atac de timp

Pașii atacului:

Atac de timp

Pașii atacului:

1. $\mathcal{A}$ trimite cerere către server pentru m si un tag aleator $t = B_1\ B_2 ... B_n$ (t este reprezentat pe octeți) și măsoara timpul până server-ul răspunde;

Atac de timp

Pașii atacului:

1. $\mathcal{A}$ trimite cerere către server pentru m si un tag aleator $t = B_1\ B_2 ... B_n$ (t este reprezentat pe octeți) și măsoara timpul până server-ul răspunde;

2. $\mathcal{A}$ trimite cerere către server pentru fiecare $t_i{}' = i\ B'_2 ... B'_n$ pentru $i = 1, 2, 3...$ ($B'_2, ..., B'_n$ sunt octeti arbitrari);

Atac de timp

Pașii atacului:

1. $\mathcal{A}$ trimite cerere către server pentru m si un tag aleator $t = B_1\ B_2 ... B_n$ (t este reprezentat pe octeți) și măsoara timpul până server-ul răspunde;

2. $\mathcal{A}$ trimite cerere către server pentru fiecare $t_i' = i\ B_2' ... B_n'$ pentru $i = 1, 2, 3...$ ($B_2', ..., B_n'$ sunt octeti arbitrari);

3. $\mathcal{A}$ se oprește când timpul de verificare e puțin mai mare decât la pasul 1;

Atac de timp

Pașii atacului:

1. $\mathcal{A}$ trimite cerere către server pentru m si un tag aleator $t = B_1\ B_2 ... B_n$ (t este reprezentat pe octeți) și măsoara timpul până server-ul răspunde;

2. $\mathcal{A}$ trimite cerere către server pentru fiecare $t_i' = i\ B_2' ... B_n'$ pentru $i = 1, 2, 3...$ ($B_2', ..., B_n'$ sunt octeti arbitrari);

3. $\mathcal{A}$ se oprește când timpul de verificare e puțin mai mare decât la pasul 1;

4. In acest caz, a găsit primul octet și continuă atacul pentru următorii octeți, pe rând, până când găsește tag-ul valid.

Soluții posibile

- Soluția nr. 1: comparația pe cele două stringuri trebuie să necesite mereu același timp:

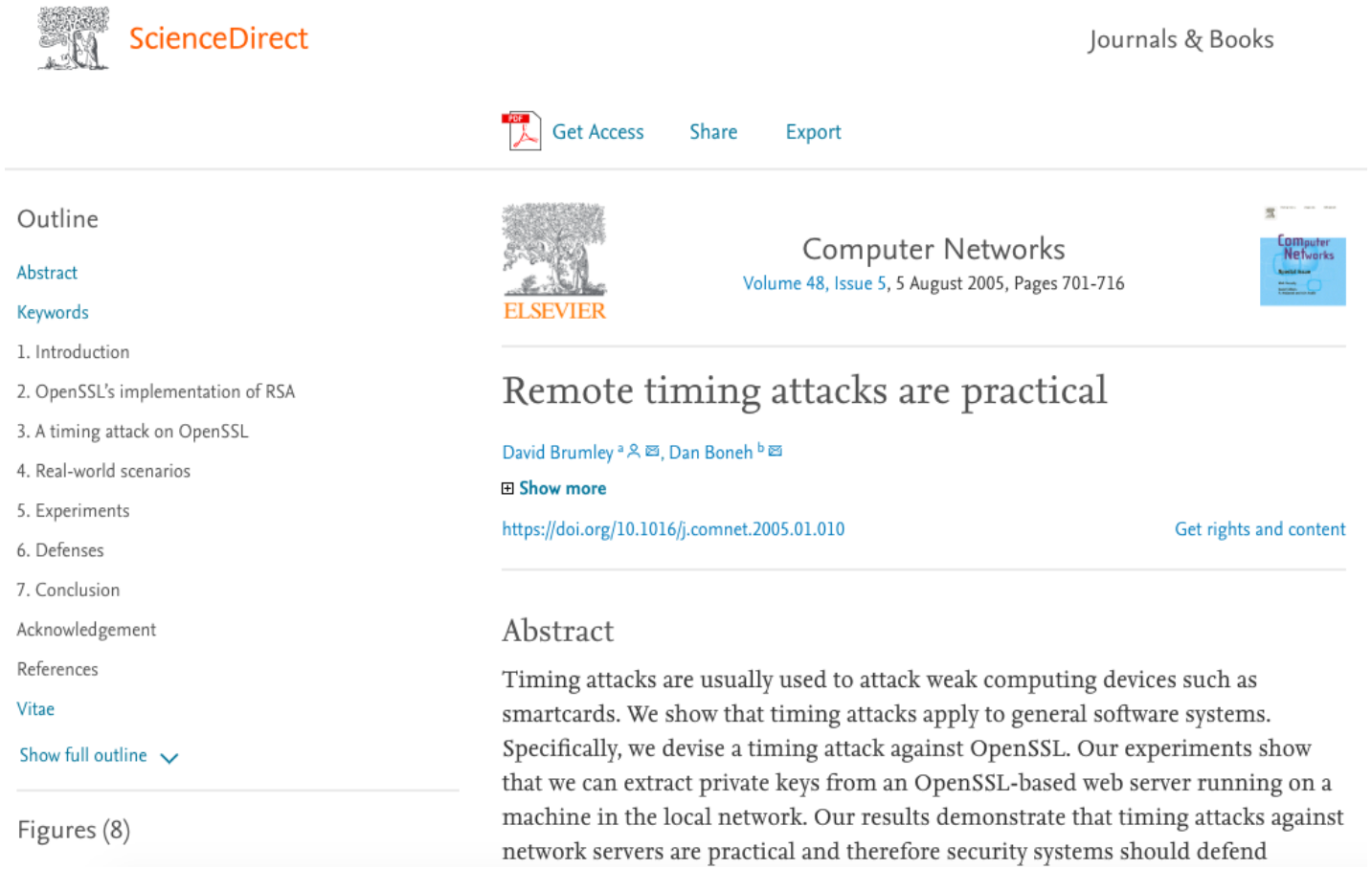
```
def Verify(key,msg,tag_bytes):
   if len(tag_bytes) != len(correct_tag) return false;
   result = 0
   for x, y in zip( HMAC(key,msg), tag_bytes):
      result |= ord(x) ^ ord(y)
   return result == 0
```

Soluții posibile

- Soluția nr. 2: comparația se face pe valorile HMAC ale celor 2 stringuri - adversarul pierde accesul direct la valorile comparate:

```
def Verify(key,msg,tag_bytes):
    mac = HMAC(key, msg)
    return HMAC(key,mac) == HMAC(key,tag_bytes)
```

Exemple


ScienceDirect
Journals & Books
Get Access
Share
Export
Outline
Abstract
Keywords
1. Introduction
2. OpenSSL's implementation of RSA
3. A timing attack on OpenSSL
4. Real-world scenarios
5. Experiments
6. Defenses
7. Conclusion
Acknowledgement
References
Vitae
Show full outline
Figures (8)
ELSEVIER
Computer Networks
Volume 48, Issue 5, 5 August 2005, Pages 701-716
Remote timing attacks are practical
David Brumley [a], Dan Boneh [b]
Show more
https://doi.org/10.1016/j.comnet.2005.01.010
Get rights and content
Abstract
Timing attacks are usually used to attack weak computing devices such as smartcards. We show that timing attacks apply to general software systems. Specifically, we devise a timing attack against OpenSSL. Our experiments show that we can extract private keys from an OpenSSL-based web server running on a machine in the local network. Our results demonstrate that timing attacks against network servers are practical and therefore security systems should defend

Exemple

Springer Link

European Symposium on Research in Computer Security

ESORICS 2011: Computer Security – ESORICS 2011 pp 355-371 | Cite as

Remote Timing Attacks Are Still Practical

Authors | Authors and affiliations

Billy Bob Brumley, Nicola Tuveri

Conference paper

45 Citations | 114 Readers | 2.1k Downloads

Part of the Lecture Notes in Computer Science book series (LNCS, volume 6879)

Abstract

For over two decades, timing attacks have been an active area of research within applied cryptography. These attacks exploit cryptosystem or protocol implementations that do not run in constant time. When implementing an elliptic curve cryptosystem with a goal to provide side-channel resistance, the scalar multiplication routine is a critical component. In such instances, one attractive method often suggested in the literature is Montgomery's ladder that

Exemple

Springer Link

Annual International Cryptology Conference

CRYPTO 1996: Advances in Cryptology — CRYPTO '96 pp 104-113 | Cite as

Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems

Authors | Authors and affiliations

Paul C. Kocher

Conference paper
First Online: 13 July 2001

564 Citations | 15 Mentions | 2 Readers | 10k Downloads

Part of the Lecture Notes in Computer Science book series (LNCS, volume 1109)

Abstract

By carefully measuring the amount of time required to perform private key operations, attackers may be able to find fixed Diffie-Hellman exponents, factor RSA keys, and break other cryptosystems. Against a vulnerable system, the attack is computationally inexpensive and

Important de reținut!

- Nu folosiți propriile voastre implementări criptografice, ci doar pe cele standardizate!