Attacking RSA-based Sessions in SSL/TLS

Vlastimil Klíma, Ondřej Pokorný, and Tomáš Rosa
v.klima@volny.cz, ondrej.pokorny@i.cz, t_rosa@volny.cz

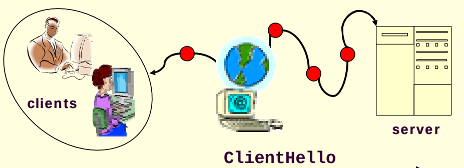
Presents: Tomáš Rosa, Dept. of computer science, FEE, CTU in Prague, CZ

The research on this project was supported by ICZ, Prague, CZ

Part I

Attack description

SSL/TLS
Session Setup



ClientHello →

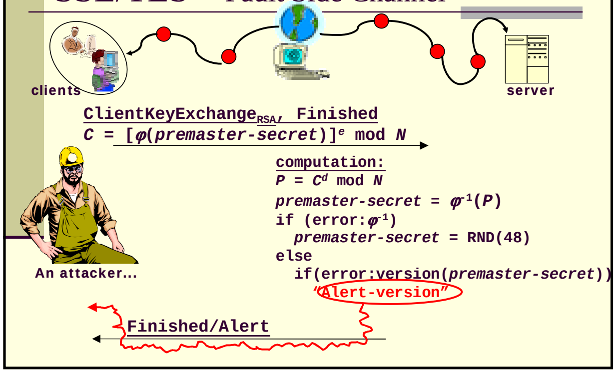
← ServerHello, ..., ServerHelloDone

..., ClientKeyExchange, Finished →

← Finished

← Transport channel →

SSL/TLS – Fault Side Channel



Mathematical Basis of the Attack

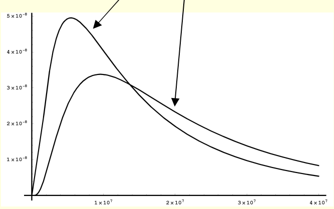
- Since φ = **EME-PKCS1-v1_5**, we may write Im(φ) $\subset$ $<E, F>$, $E, F \in \mathbf{Z}$.
  - note that for any x, $\varphi(x)$ = **00 || 02 ||…**
- Seeing "Alert-version" we know that $P \in$ Im(φ), therefore $P \in <E, F>$.
- Let C_0 be the ciphertext we want to invert (with respect to RSA),

  $C_0 = P_0^e \bmod N$.
- Let $P = C^d \bmod N$, $C = C_0 s^e \bmod N$, $s \in \mathbf{Z}$.
  - note that P is still an unknown plaintext, $P = P_0 s \bmod N$
- Now, seeing "Alert-version" we know that $E \leq sP_0 \bmod N \leq F$.
- From here, we get a non-trivial information on P_0, since there is $r \in \mathbf{Z}$, such that:
  - $\mathbf{(E+rN)/s \leq P_0 \leq (F+rN)/s}$
- Searching for various s producing "Alert-version" we can narrow the set of solutions for P_0 to get one particular value which is then the inverse of C_0.
  - each such s roughly halves our uncertainty on P_0

Queries Distribution

1024 bit RSA key
min: 815 835
median: 13 331 256

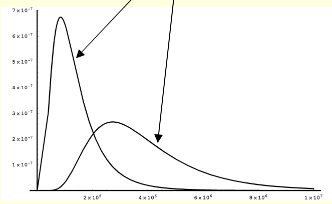
2048 bit RSA key
min: 2 824 986
median: 19 908 079



Queries Distribution

1025 bit RSA key
min: 630 589
median: 1 197 380

2049 bit RSA key
min: 1 413 005
median: 3 462 557



Experimental Time Measurements

- General intranet server:
  - 2x Pentium III/1.4 GHz, 1 GB RAM, 100 Mb/s Ethernet
  - OS RedHat 7.2, Apache 1.3.27
  - moderately loaded network connection
  - speed: 67.7 queries per second
  - median obtained: cca 54 h 42 min

Illustration of the Attack Scenario

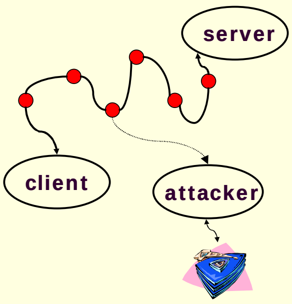
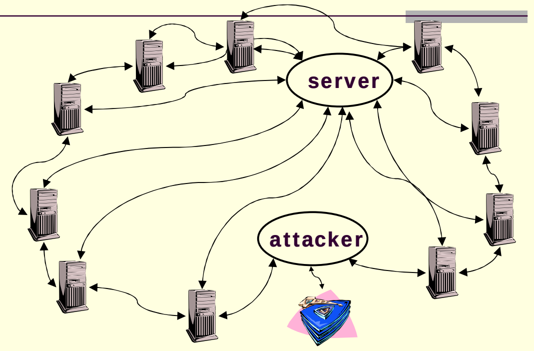


Illustration of the Attack Scenario

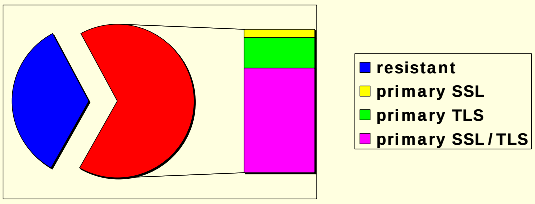



```
...
...
goods ID: hf582de4
remark: XXL, natural color
-------------
CARD-ID: 1456 2265 5554 5468
NAME: Mr. George Doubal
EXPIRES: 02/2006
-------------
Address:
       U stromu 8
110 00 Praha 10
       Czech Republic
...
...
```

Cross-attacking

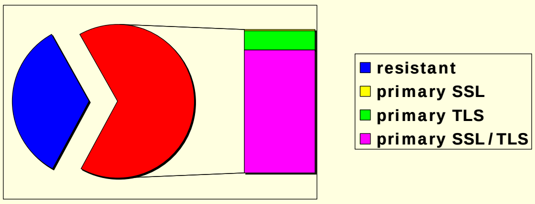
- The core components allowing the attack are nearly the same for both SSL and TLS
- Private keys are often shared between SSL and TLS running on the same server
- Therefore, we can discover the *premaster-secret* for a SSL connection by attacking <u>primarily</u> TLS implementation and vice versa

Internet Servers Vulnerability



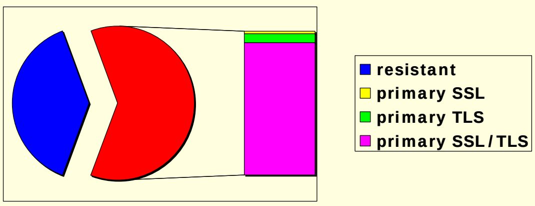
10.3. 2003; 611 randomly selected servers

Internet Servers Vulnerability (2)



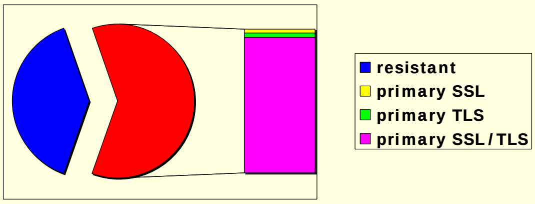
31.3. 2003; 586 randomly selected servers

Internet Servers Vulnerability (3)



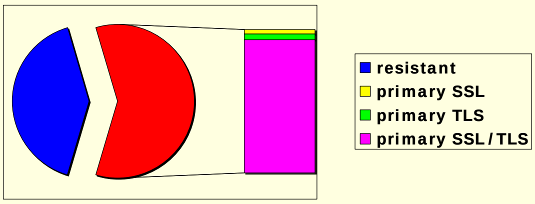
2.5. 2003; 547 randomly selected servers

Internet Servers Vulnerability (4)



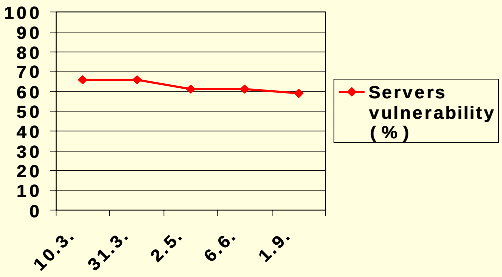
6.6. 2003; 544 randomly selected servers

Internet Servers Vulnerability (5)



1.9. 2003; 533 randomly selected servers

Vulnerability Trend



Security Management...?

Part II

Countermeasures

General Assumption-Condition

- Let C be an RSA ciphertext corresponding to an **unknown** *premaster-secret*.
  - $C = [\varphi(\textit{premaster-secret})]^e \bmod N$, where φ is a EME-PKCS1-v1_5 encoding
- AC1 We assume that it is **infeasible** for an attacker **to distinguish** whether the server uses the value of *premaster-secret* or if it uses a randomly generated value of *premaster-secret'* instead.
- A2 Furthermore, we assume that using the randomly generated value of *premaster-secret'* makes the handshake procedure fail with a probability close to one.

A Wrong Way (WW)

```
1. RSA decryption: C → P, P = C^d mod N
2. if P is PKCS-conforming
     then pms ← last_48_bytes(P)
     else pms ← rand(48)
3. proceed with premaster-secret = pms
   (this includes version number check, etc.)
```

- **Why is it bad?** It focuses solely on repairing the fact that the version number check was done only for PKCS-conforming plaintexts.
- **It conflicts with assumption AC1:** Sending many oracle queries with the same value of *C*, an attacker can distinguish between using decoded or randomly generated *premaster-secret*. She uses results from the version number check to do so.

A Better Way #1 (BW1)

```
1. RSA decryption: C → P, P = C^d mod N
2. if P is S-PKCS-conforming and version number
   is OK
     then pms ← last_48_bytes(P)
     else pms ← rand(48)
3. proceed with premaster-secret = pms
   (version number check is not repeated)
```

- **Problems with AC1 from WW are solved.**
- **Theoretical vulnerability:** An attacker can control the condition in step 2 by manipulating the expected version number. It might be perhaps helpful together with some power or electromagnetic side channels – the attacker can learn how to break assumption A1.

A Better Way #2 (BW2)

```
1. RSA decryption: C → P, P = C^d mod N
2. if P is S-PKCS-conforming
     then pms ← last_48_bytes(P)
     else pms ← rand(48)
3. first_2_bytes(pms) ← expected version number
4. proceed with premaster-secret = pms
   (explicit version number check is omitted)
```

- **Problems with AC1 seem to be solved**, even for some other side channel attacks. An attacker has a lower chance to learn how to break assumption A1.

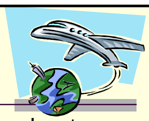
Part III

Concluding remarks

General Characteristics Repeated

- Based on fault side channel
  - an attacker observes server's reaction on incorrectly structured data
- Allows the attacker to compute RSA decryption with the server's private key
  - works for arbitrary input value
  - main target is a value of *premaster-secret*
- Extends Bleichenbacher's attack from 1998 (presented at CRYPTO '98)
- Feasibility depends on a concrete implementation

Lessons learned



- Any possible source of information about RSA plaintext must be carefully investigated
  - also – it's worth it to read several lines bellow a patch we make
- We can hardly say that all internet servers are maintained properly
  - better of preaching that security is mainly about its management is to really start to manage it

Thank you for your attention