HACKMANIT

RUB

Return Of Bleichenbacher's Oracle Threat (ROBOT)

Hanno Böck

Juraj Somorovsky (Ruhr University Bochum / Hackmanit)

Craig Young (Tripwire VERT)

Recent Attacks on TLS

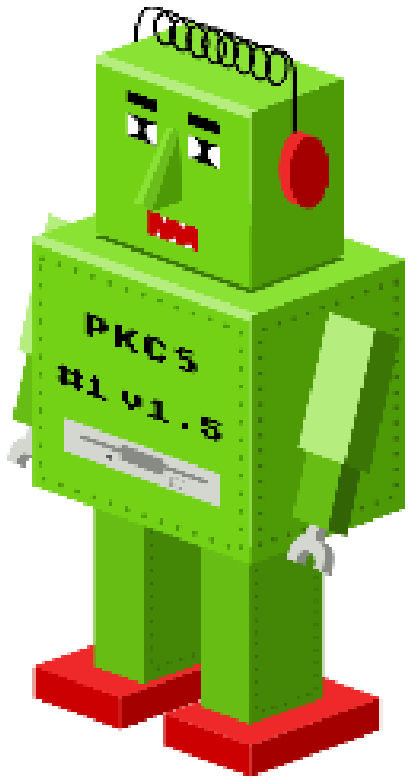
- CRIME, BEAST, Lucky 13, Heartbleed, Early CCS

- 20 years ago: Bleichenbacher's attack
  - Applied to RSA PKCS#1 v1.5 in SSL/TLS
  - Decrypt SSL/TLS traffic
  - Implementations applied ad-hoc fixes
  - Everything is secure, right?
- Return of Bleichenbacher's Oracle Threat – ROBOT*

* Name idea shamelessly stolen from ROCA ☺

Overview

1. **Bleichenbacher's attack**
2. **How we started – Attack on Facebook**
3. **Performing the scans**
4. **Responsible disclosure**
5. **Conclusions**

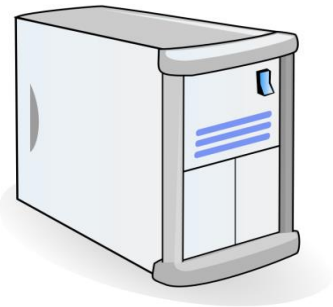


Designed by Ange Albertini

TLS Protocol (High Level Overview)

1. TLS Handshake
   - Selection of algorithm, version, extensions
   - Key exchange: **RSA**, (EC)DH, (EC)DHE
2. Encrypted and authenticated data transport

TLS RSA Handshake



ClientHello →

ServerHello
Certificate
ServerHelloDone →

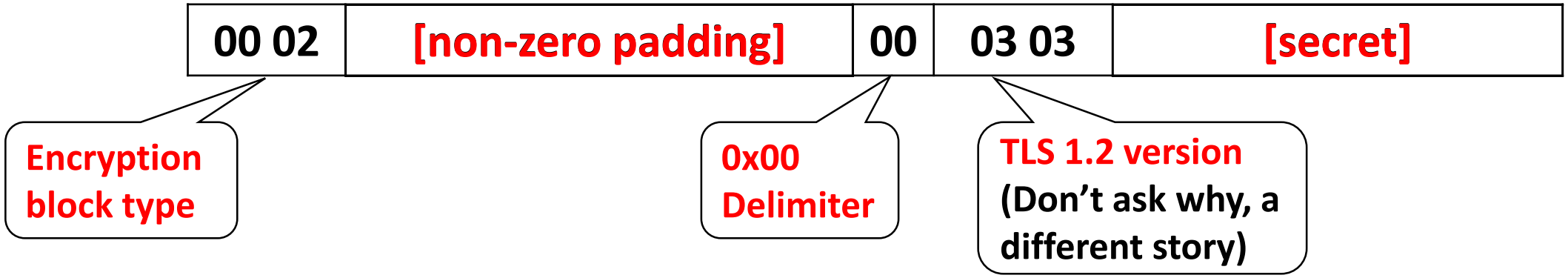
ClientKeyExchange
ChangeCipherSpec
(Client-) Finished →

RSA encrypted premaster secret

ChangeCipherSpec
(Server-) Finished →

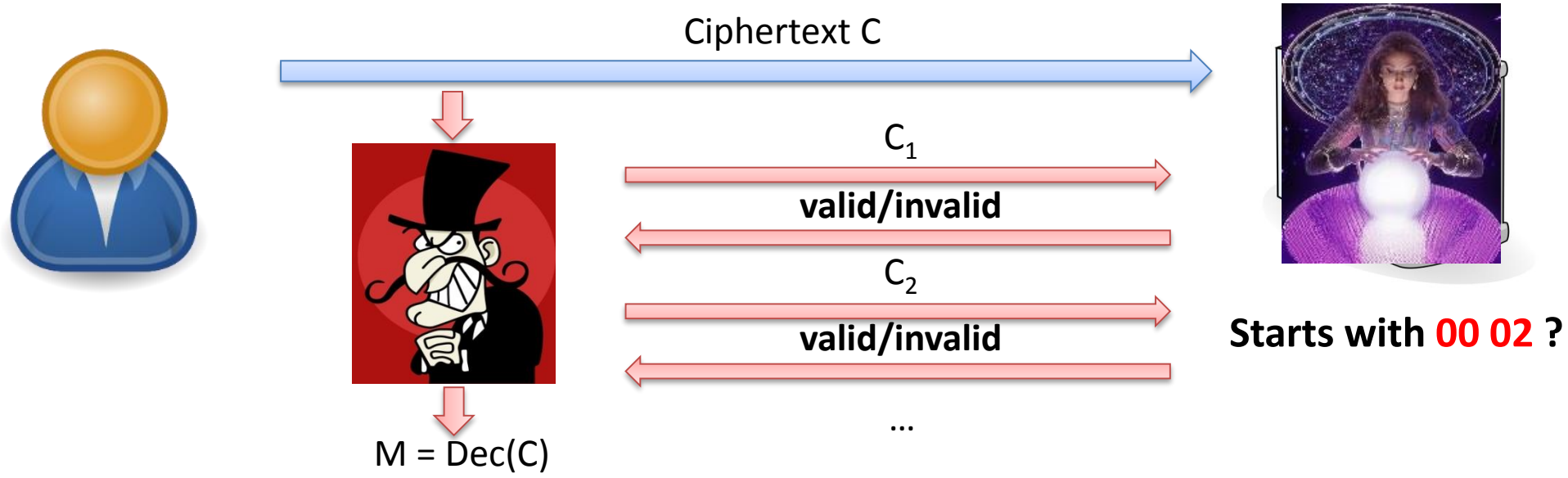
RSA PKCS#1 v1.5

- Used to pad and encrypt the premaster secret:
  - To pad it to the RSA key length
  - To add randomization
- Example for TLS 1.2:



Bleichenbacher's Attack

- 1998: Adaptive chosen-ciphertext attack
- Exploits strict **RSA PKCS#1 v1.5** padding validation



Bleichenbacher's Attack

- The attack needs some math (Not going into details here)

- "Million message attack"

(In general performance depends on the oracle properties)

Step 1: Blinding. Given an integer c, choose different random integers s_0; then check, by accessing the oracle, whether $c(s_0)^e \bmod n$ is PKCS conforming. For the first successful value s_0, set

$$c_0 \leftarrow c(s_0)^e \bmod n$$
$$M_0 \leftarrow \{[2B, 3B-1]\}$$
$$i \leftarrow 1.$$

Step 2: Searching for PKCS conforming messages.
Step 2.a: Starting the search. If $i = 1$, then search for the smallest positive integer $s_1 \geq n/(3B)$, such that the ciphertext $c_0(s_1)^e \bmod n$ is PKCS conforming.
Step 2.b: Searching with more than one interval left. Otherwise, if $i > 1$ and the number of intervals in M_{i-1} is at least 2, then search for the smallest integer $s_i > s_{i-1}$, such that the ciphertext $c_0(s_i)^e \bmod n$ is PKCS conforming.

Step 2.c: Searching with one interval left. Otherwise, if M_{i-1} contains exactly one interval (i.e., $M_{i-1} = \{[a,b]\}$), then choose small integer values r_i, s_i such that

$$r_i \geq 2\frac{bs_{i-1} - 2B}{n} \tag{1}$$

and

$$\frac{2B + r_i n}{b} \leq s_i < \frac{3B + r_i n}{a}, \tag{2}$$

until the ciphertext $c_0(s_i)^e \bmod n$ is PKCS conforming.
Step 3: Narrowing the set of solutions. After s_i has been found, the set M_i is computed as

$$M_i \leftarrow \bigcup_{(a,b,r)} \left\{ \left[\max\left(a, \left\lceil \frac{2B + rn}{s_i} \right\rceil\right), \min\left(b, \left\lfloor \frac{3B - 1 + rn}{s_i} \right\rfloor\right)\right] \right\} \tag{3}$$

for all $[a,b] \in M_{i-1}$ and $\frac{as_i - 3B + 1}{n} \leq r \leq \frac{bs_i - 2B}{n}$.

Step 4: Computing the solution. If M_i contains only one interval of length 1 (i.e., $M_i = \{[a,a]\}$), then set $m \leftarrow a(s_0)^{-1} \bmod n$, and return m as solution of $m \equiv c^d \pmod n$. Otherwise, set $i \leftarrow i + 1$ and go to step 2.

...skipped if c is already PKCS conforming (i.e., when

Return Of Bleichenbacher's Oracle Threat (ROBOT). USENIX Security 2018

Creating Bleichenbacher's Oracle

Server

ClientHello →

← ServerHello

← Certificate

← ServerHelloDone

Modified ciphertext

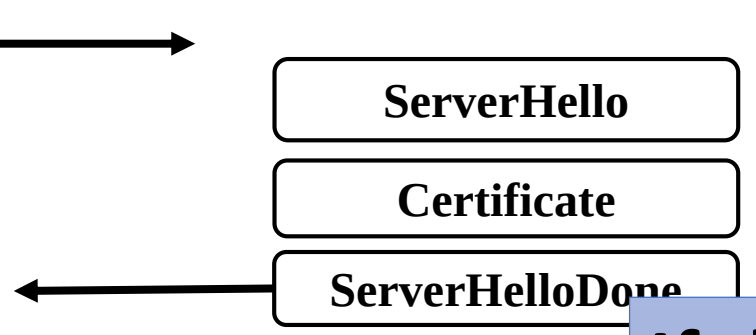
ClientKeyExchange'

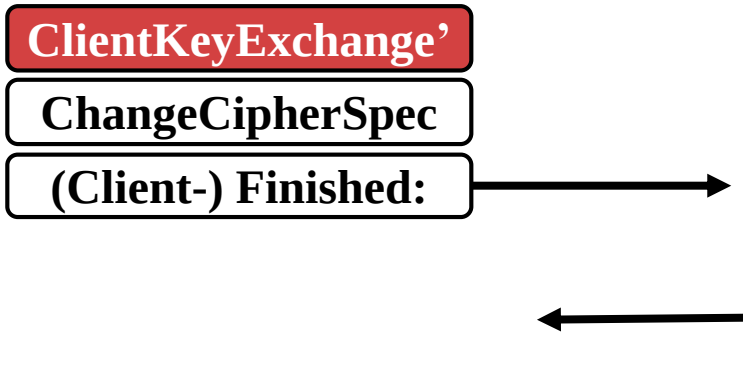
ChangeCipherSpec

(Client-) Finished: →

← Decrypt Error / Bad Record MAC Alert

TLS Countermeasure

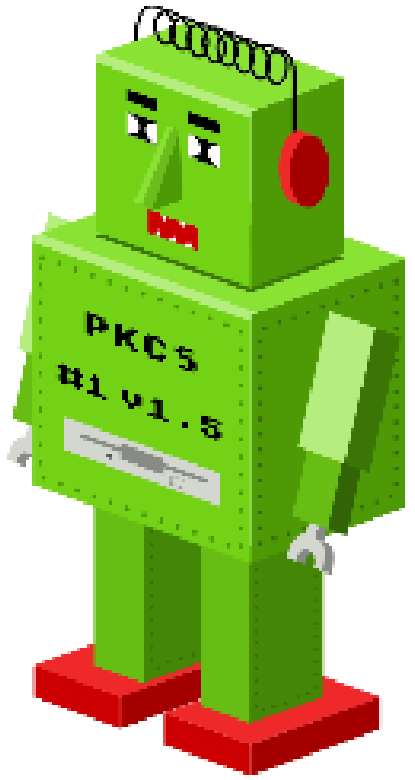




If the attacker can distinguish valid / invalid PKCS#1 messages, he wins

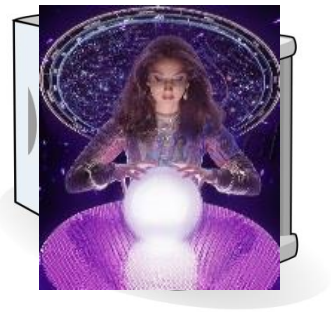
Overview

1. **Bleichenbacher's attack**
2. **How we started – Attack on Facebook**
3. **Performing the scans**
4. **Responsible disclosure**
5. **Conclusions**



Designed by Ange Albertini

Hanno Found a Weird Behavior of Facebook



Server

ClientHello →

← **ServerHello**

← **Certificate**

← **ServerHelloDone**

ClientKeyExchange'

ChangeCipherSpec

(Client-) Finished: →

← **Illegal Parameter** / **Bad Record MAC Alert**

Can We Exploit It?

- Idea: It would be funny to sign a message with Facebook's private key
  - Yes, **signing is possible** as well
- Millions of queries needed…would Facebook block us?
- Successful after several tries:

  "We hacked Facebook with a Bleichenbacher Oracle (JS/HB)."

- Facebook fixed

```
echo 799e4353 5a4da709 80fada33 d0fbf51a e60d32c1
115c87ab 29b716b4 9ab06377 33f92fc9 85f280fa 569e41e2
847b09e8 d028c0c2 a42ce5be eb640c10 1d5cf486 cdffc5be
116a2d5b a36e52f4 195498a7 8427982d 50bb7d9d 938ab905
40756535 8b1637d4 6fbb60a9 f4f093fe 58dbd251 2cca70ce
842e74da 078550d8 4e6abc83 ef2d7e72 ec79d7cb 2014e7bd
8debbd1e 313188b6 3a2a6aec 55de6f56 ad49d32a 1201f180
82afe3b4 edf02ad2 a1bce2f5 7104f387 f3b8401c 5a7a8336
c80525b0 b83ec965 89c36768 5205623d 2dcdbe14 66701dff
c6e768fb 8af1afdb e0a1a626 54f3fd08 175069b7 b198c471
95b63083 9c663321 dc5ca39a bfb45216 db7ef837 | xxd -r
-p > sig
curl
https://crt.sh/?d=F709E83727385F514321D9B2A64E26B1A195
751BBCAB16BE2F2F34EBB084F6A9|openssl x509 -noout -
pubkey > pubkey.key
openssl rsautl -verify -pubin -inkey pubkey.key -in
sig
```

Facebook: New Attempt

Server

ClientHello →

← ServerHello

← Certificate

← ServerHelloDone

ClientKeyExchange' →

[illegible] →

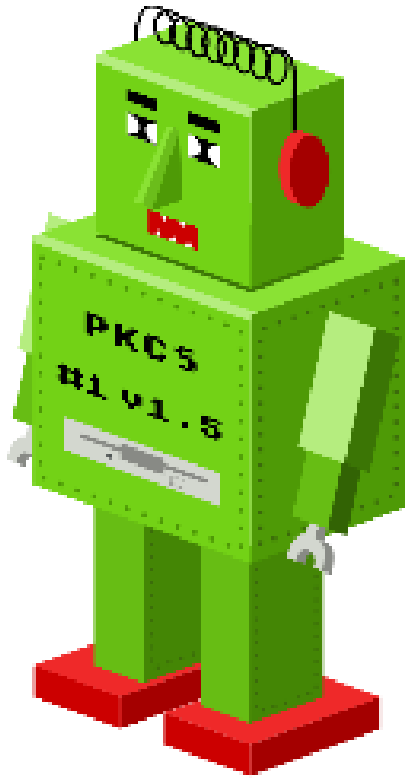
/

Facebook Fixed Again

- This is interesting. So how about other servers?

Overview

1. **Bleichenbacher's attack**
2. **How we started – Attack on Facebook**
3. **Performing the scans**
4. **Responsible disclosure**
5. **Conclusions**

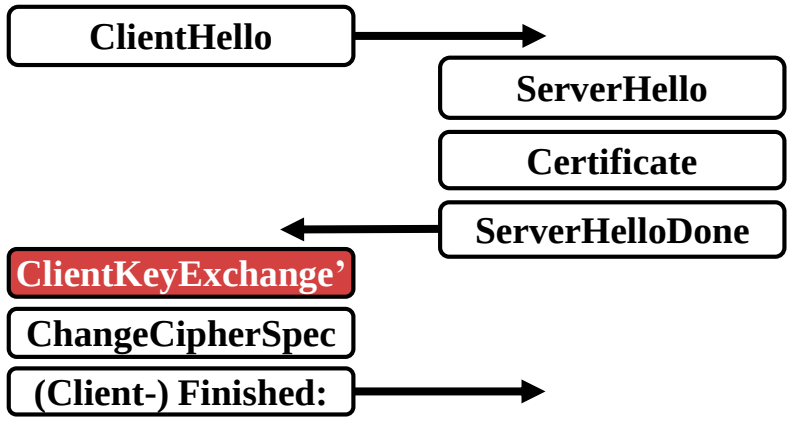


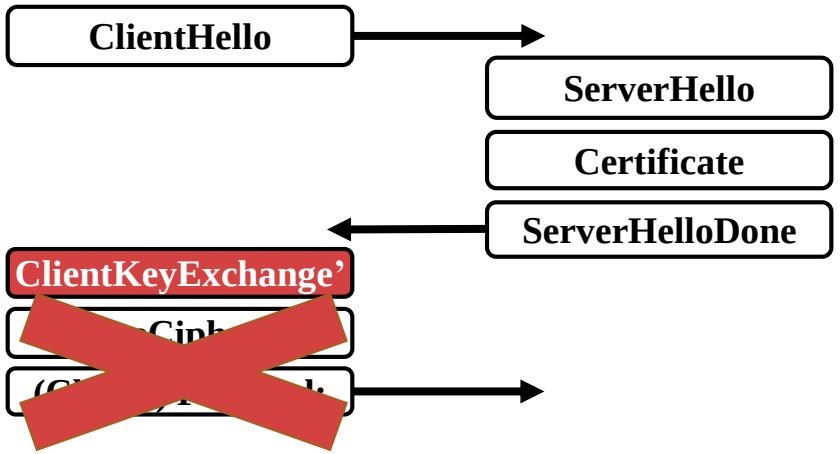
Designed by Ange Albertini

Let's Start Scanning

- Careful selection of ClientKeyExchange messages:
  - Wrong TLS version
  - Wrong padding length
  - Not starting with 0x00 02

- Full / Shortened TLS handshakes:





Alexa Top 1 Million Scan

- 2,8 % vulnerable
- PayPal, Apple, ebay, Cisco, …
- Different behaviors…different combinations:

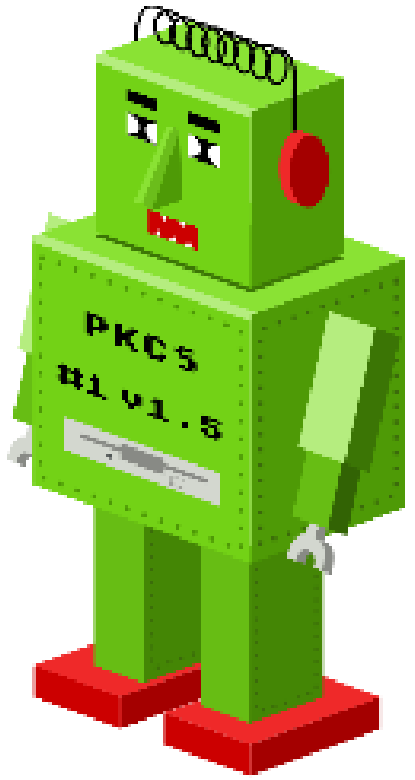
TCP connection resets  Timeouts 

Different alerts **Illegal Parameter** / **Bad Record MAC Alert** / **Handshake Failure** / **Internal Error** /..

Duplicate alerts **Alert** / **Alert** **Alert**

Overview

1. **Bleichenbacher's attack**
2. **How we started – Attack on Facebook**
3. **Performing the scans**
4. **Responsible disclosure**
5. **Conclusions**



Designed by Ange Albertini

Who Is Responsible for These Mistakes?

- Reporting is not always that easy …

Your server is vulnerable to Bleichenbacher‘s attack.

No worries, we use **millitary** grade encryption.

Don’t Fix for Some Vendors … Cisco ACE

- Supports only TLS RSA
- Cisco: We won't fix it, it's out of support for several years

- But there were plenty of webpages still running with these devices

Like cisco.com

Identified (Most of) Them

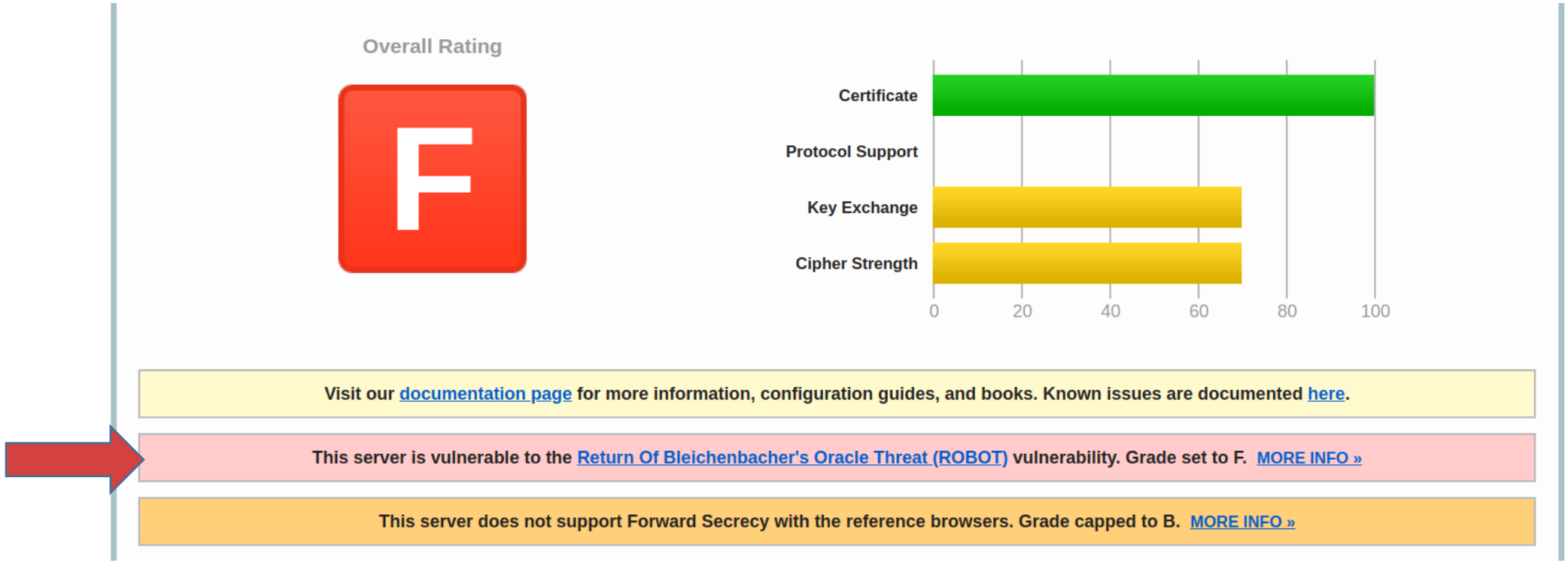
Implementation	Server response		TLS flow	Oracle	Reference / ID
	Valid message	Invalid message			
Facebook					
1st vulnerability	20	47	full	strong	-
2nd vulnerability	20	TCP FIN	shortened	strong	-
F5					
Variant 1	TCP timeout	40	shortened	strong	CVE-2017-6168
Variant 2	One alert (40)	Two alerts (40)	full	strong	CVE-2017-6168
Variant 3	TCP timeout	40	shortened	weak	CVE-2017-6168
Variant 4	One alert (40)	Two alerts (40)	full	weak	CVE-2017-6168
Variant 5	20	80	full	strong	CVE-2017-6168
Citrix Netscaler					
with CBC cipher suites	Connection reset	TCP timeout	full	strong	CVE-2017-17382
with GCM cipher suites	51	TCP timeout	full	strong	CVE-2017-17382
Radware					
Radware Alteon	51	TCP reset	full	strong	CVE-2017-17427
Cisco					
Cisco ACE	20	47	full	strong	CVE-2017-17428
Cisco ASA	TCP timeout	TCP reset	full	weak	CVE-2017-12373
Erlang					
Erlang version 19 and 20	10	51	full	strong	CVE-2017-1000385
Erlang version 18	20	51	full	strong	CVE-2017-1000385
Palo Alto Networks					
PAN-OS	One alert (40)	Two Alerts (40)	full	weak	CVE-2017-17841
IBM					
IBM Domino	20	47	full	weak	(unfixed)
IBM WebSphere MQ	?	?	?	?	CVE-2018-1388

F5 had 5 different vulnerabilities!

Return Of Bleichenbacher's Oracle Threat (ROBOT). USENIX Se

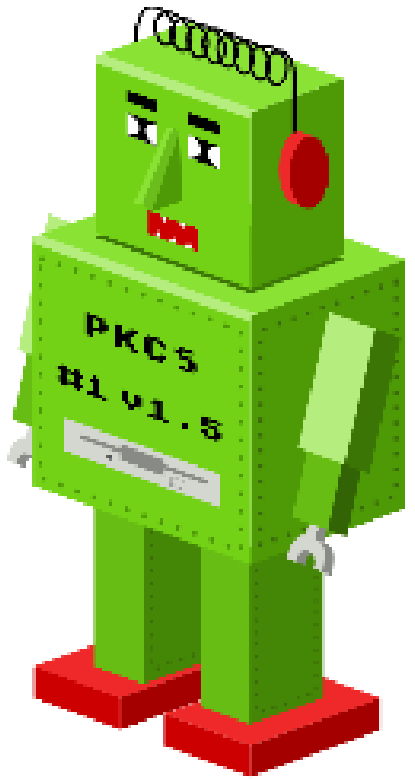
Test Tools

- No easily usable test tool for Bleichenbacher attacks available
- Currently implemented in SSL Labs, testssl.sh, TLS-Attacker, tlsfuzzer



Overview

1. **Bleichenbacher's attack**
2. **How we started – Attack on Facebook**
3. **Performing the scans**
4. **Responsible disclosure**
5. **Conclusions**



Designed by Ange Albertini

Future Work

- Timing attacks
- Fingerprinting

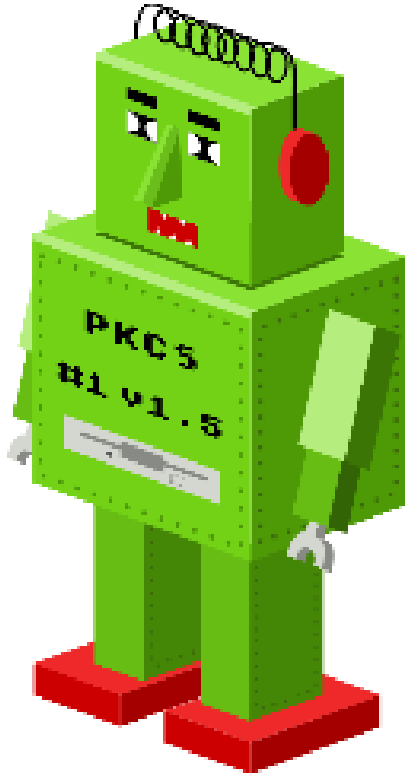
Illegal Parameter / **Bad Record MAC Alert** /..

- Some servers send certificates or "garbage bytes"
  - **Bleedinbacher**? There could be a Heartbleed-style memory disclosure waiting to be found

Conclusions

- Old **20** year attacks still work
- New side-channels (timeouts, TCP resets, …)
- Crypto attack countermeasures are hard to apply

- Disable TLS_RSA cipher suites (not used in TLS 1.3)
- Stop using RSA PKCS#1 v1.5, use elliptic curves (or RSA-OAEP if RSA needed)



https://robotattack.org/