PADDING ORACLE FOR THE MASSES
What is this presentation about?

• This presentation is a scrap book from our experience developing a reliable exploit against ASP.Net
• It tooks 2 people working full time to create a reliable and working exploit for this vulnerabilities
• All the kudos to Juliano Rizzo and Thai Duong for finding such a clever technique and teaching the world about the risks
Padding Oracle: it’s not a vulnerability it’s an *Attack*
What is the vulnerability?

- The vulnerability is a bad crypto implementation when using cbc mode of operation.
- A block cipher by itself allows encryption of only a single data block of the cipher lengths.
- IBM came out with CBC (Cipher Block chaining), this mode of operation causes the decryption of a block of ciphertext to depend on all the preceding ciphertext blocks (i.e., If you encrypt again the same block, the ciphertext will be different.)
I AM PROTECTED.
IM USING AES
CONFIDENCIALITY != AUTHENTICITY
HOW DOES CBC WORKS?
Decryption process

C0

IV

dk(C0)

IMVO

+ 

P0
Decryption process

\[
\begin{align*}
\text{IMV0} & \quad \text{IMV1} \\
\text{P0} & \quad \text{P1}
\end{align*}
\]
HOW DOES PADDING WORKS?
PKCS7

Seven bytes data, padded with one \x01 byte

Six bytes data, padded with two \x02 bytes
PKCS7

Eight bytes data, padded with eight \x08 bytes
PADDING ORACLES 101
Evil Hacker

QWERTYBLAHBLAH

CORRECT DECRIPTION

WRONG PADDING
WRONG PADDING

The difference between a correct decryption or a wrong padding came in different flavours:

- Time difference
- Error Code
- Stack trace
- HTML length
- Strawberry
captcha.php?val=IV

IV

Encrypted Data

ABCDEFGH IJKLMNOP
captcha.php?val=ABCDEFGHIJKLMNOPQRSTUVWXYZ

IJKLMNOP

ABCDEF

PYWTL333
Controlled DATA

ABCDEFGH

IJKLMNOP

dk(...)

PYWTL333
plaintext

IMV

IJKLMNOP

dk(...)

X

1 2 3 4 5 6 7 8

+ ABCDEFGH

IV

= PYWTL333
So, if by some means we can know the IMV for a specific block, we can forge custom captchas.
HOW DOES THE ATTACK WORKS?
We modify the last byte from the IV
We leave the enc block Fixed
This will make the final result work or fail, in this case we can see that 33Z is not a valid padding.
We keep changing the iv’s last character until we find the correct padding (the web will behave differently).

Padding Ok!
\[ X_8 = 1 + T = 0x55 \]
Now that we know the 8th byte of the IMV, we go for the 7th
First we set the last IV byte such as the last decrypted byte is a 2.
Now we try to find the 7th IMV byte by keep changing the 7th IV byte.
IJKLMNOP

Fixed Padding Ok!
If you want to make your encrypted buf say something, you already have the pieces!
We calculate an IV in order to produce our message :)
It's Fixed on the Server

IV

C0

dk(C0)

IMV0

P0

C1

dk(C1)

IMV1

P1
Your target is ScriptResource.axd, this will allow you to download any file on the www root, including web.config (machine password, etc).

It has its own “protocol” for doing stuffs.
R|~/web.config
\texttt{R#XXXXXXXXXXX}|||\sim\texttt{/web.config}
• After a lot of brain cells burning, we came out with the following:

- RANDOM
- IV for next
- IV for next
- ABCDEFGHI
- R#xxxxxx
- TRASHED
- |||~/web
- .config
RANDOM

IV for next

Obtained via randomly modifying the block
12k to 100k average (could be more).

R#xxxxxx

IV for next

Obtained via Padding Oracle.
Around 2k request.

TRASHED

|||~/web

ABCDEFGHI

$config$
Workarounds

- They don’t work, the only way to fix the bug is the patch
- Examples:
  - Redirecting all the logs to the same web
  - Adding a random sleep to each request
MAGIC "T" BLOCK
Randomly change a block, to obtain a T. This will decrypt all the other blocks. 50-1000 hits
```csharp
private static void ProcessRequestInternal(HttpResponse response, NameValueCollection queryString, virtual FileReader fileReader)
{
    bool flag;
    string str = DecryptParameter(queryString);
    switch (str[0])
    {
    case 'q':
    case 'Q':
        flag2 = false;
        flag = true;
        break;
    case 'r':
    case 'R':
        flag2 = false;
        flag = false;
        break;
    case 'u':
    case 'U':
        flag2 = true;
        flag = false;
        break;
    case 'z':
        
    case 'T':
        OutputEmptyPage(response, str.Substring(1));
    return;
    
    Throw404();
    return;
    }
    str = str.Substring(1);
    if (string.IsNullOrEmpty(str))
    {
        Throw404();
    }
    string[] strArray = str.Split(new char[] { '!' });
    if (flag2)
```
MAGIC “T” Block

- The magic “T” Block will replace padding oracle
- It will also allow you to do CBC-R
- This will bypass all types of workarounds
- Can speed up the QR-Block lookup!!
First objective: find a QRBlock
Bruteforce!!!
Send a lot of random blocks

T-BLOCK  random1  random2  RandomN  Padding blocks
Txxxxxx  trashhh  trashhh  trashhh  Trash + padding
Until we find our QR-Block
Life it's no so easy...
We need to set a correct IV for each brute force block
We need to mimic the first block

- **IV**
- **randomX**
- **IMV**
- **$$...$$**

- **Padding blocks**
  - blah
  - blah

- **Trash + padding**
  - blah
  - blah
Simplification: 
default IV =
\"\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\"
If instead of sending 8 random bytes you send `\x00\x00 + 6` random bytes, everything is simpler!
Life is easy again :)
but...
we can be out of phase

T-BLOCK  random1  random2  ...  randomX  ...  randomN  Padding blocks

Txxxxx  trashhh  ✅✅✅R.  #......  ashhhtr  Trash + padding

False negative
The encoding “eat” some characters
Solution:
Every n blocks we send a mark
The Padding Oracle attack took 35,000 requests in avg.
The **Magic-T** takes 700 request
(\(t\)block: 400, \(q\)\(_r\)block: 300)
Is that all?

- ASP.net is just one wrong implementation, there are more.
- As a consultant you should be looking for:
  - Session keys that looks like base64 (ASP.NET Uses UrlEncoded base64, it is a bit different)
  - Encrypted cookies
  - Persisted information such as viewstate
  - "Any encrypted information that is stored client-side and returned to the server"
Conclusion

• Workarounds are useless. PATCH!
• Exploits once again show themselves to be a necessary tool to prove server risks
• This is a error of implementation, even if you fix asp.net, your own developers' software could have made their own crypto and be vulnerable
More information

- Security Flaws Induced by CBC Padding – S. Vaudenay
- Padding Oracle Attacks on the ISO CBC Mode Padding Standard – K.G. Paterson and A. Yau
- Practical Padding Oracle Attacks – J. Rizzo and T. Duong
THANKS

NICO WAISMAN
nico@immunityinc.com
@nicowaisman

Matias Soler (@gnuler)
Agustín Gianni (@agustingianni)