Padding Oracles Everywhere

T. Duong\textsuperscript{1} J. Rizzo\textsuperscript{2}

\textsuperscript{1}VNSEC/HVA

\textsuperscript{2}NETIFERA

EKOPARTY 2010
Outline

1. Introduction
   - Review of CBC mode
   - Padding oracle attack

2. Basic PO attacks
   - POET vs CAPTCHA
   - POET vs JavaServer Faces

3. Advanced PO attacks
   - Distributed cross-site PO attacks
   - Using PO to encrypt

4. 0-day: POET vs ASP.NET
   - ASP.NET’s design problems
   - Padding oracles in ASP.NET
CBC Mode

- CBC mode is a cryptography mode of operation for a block cipher.
- Allows encryption of arbitrary length data.
- Encryption and decryption are defined by:

\[ C_i = e_K(P_i \oplus C_{i-1}) \]

\[ P_i = d_K(C_i) \oplus C_{i-1} \]
CBC Mode

Typical block size $n$: 64 bits (DES, triple DES) or 128 bits (AES).

Typical key size: 56 bits (DES), 168 bits (triple DES), 128, 192 or 256 bits (AES).
Padding

Hello world
11 bytes of plaintext

PKCS5 Padding

Hello world

r l d 05 05 05 05 05

Encryption
Padding oracle attack

Introduction

- First introduced by Vaudenay at Eurocrypt 2002.

- Two assumptions:
  - Adversary can intercept padded messages encrypted in CBC mode.
  - Adversary has access to a padding oracle.
Padding oracle attack

What is a padding oracle?
Padding oracle attack

What is a padding oracle?

- Adversary submits a CBC mode ciphertext $C$ to oracle $\tilde{o}$.
- Oracle decrypts under fixed key $K$ and checks correctness of padding.
- Oracle outputs VALID or INVALID according to correctness of padding:

$$\tilde{o}(C) = \begin{cases} 
0, & \text{invalid} \\
1, & \text{valid}
\end{cases}$$
Padding oracle attack
How does it work?

- For a long message, decrypt block by block. It’s easy to parallelize the attack.
- For a block, decrypt the last byte first, then decrypt the next to last byte, and so on.
- How?
Padding oracle attack

How to decrypt a block

1. Cipher-text
2. No
3. Cipher-text
4. No
5. Cipher-text
6. No
7. ... 256 tries max
8. Cipher-text
9. Yes
Padding oracle attack

How to decrypt a block

Oracle CBC decryption process

Oracle query cipher-text

1. Decrypts control block

2. XOR with IV
Padding oracle attack

How to decrypt a block

3. Decrypt target

4. XORs with control

Final “plain-text”
Padding oracle attack
Last byte decryption algorithm

Last byte decryption algorithm

- pick a few random bytes $r_1, \ldots, r_b$, and take $i = 0$.
  - pick $r = r_1 r_2 \ldots r_{b-1} (r_b \oplus i)$.
  - if $\bar{d}(r|y) = 0$ then increment $i$ and go back to previous step.
  - replace $r_b$ by $r_b \oplus i$.
- for $n = b$ down to 2
  1. take $r = r_1 \ldots r_{b-n} (r_{b-1+1} \oplus 1) r_{b-n+2} \ldots r_b$
  2. if $\bar{d}(r|y) = 0$ then stop and output $(r_{b-n+1} \oplus n) \ldots (r_b \oplus n)$
- output $r_b \oplus 1$.
A broken CAPTCHA system

- $ERC = e_{K, IV}(\text{rand}())$.
  - $...<\text{img src="/captcha?token=ERC" /}>...$
  - $ERC$ is stored as either a hidden field or a cookie in the CAPTCHA form.

- Once a user submits, the server decrypts $ERC$, and compares it with the code that the user has entered. If equal, the server accepts the request; it denies the request otherwise.
POET vs CAPTCHA
Bypass the broken CAPTCHA system

- Since the system decrypts any ERC sent to it, it is vulnerable to Padding Oracle attack.

- The only remaining problem now is to know when padding is VALID, and when it’s not.

- Fortunately, most CAPTCHA systems would send back an error notification when they fail to decrypt ERC, i.e. padding is INVALID.

- In addition, when we modify ERC so that the padding is VALID, most systems would display an image with a broken code.

- Now we have a padding oracle, and we can use it to decrypt any ERC, thus bypass the CAPTCHA completely.
JavaServer Faces (JSF) is a popular Java-based standard for building server-side user interfaces.

Like ASP.NET, JSF stores the state of the view in a hidden field.

Although JSF specification advises that view state should be encrypted and tamper evident, but no implementation follows that advice.

In other words, we can use padding oracle attacks to decrypt the view states of most JSF frameworks.
By default, all JSF frameworks would display a very detailed error message if it fails to decrypt a view state.

Padding oracle in default installations of JSF frameworks

- if we see `javax.crypto.BadPaddingException`, then it’s INVALID padding
  - it’s VALID padding otherwise.
POET vs JavaServer Faces

Apache MyFaces error-page

An Error Occurred:

javax.crypto.BadPaddingException: Given final block not properly padded

Caused by:
javax.crypto.BadPaddingException - Given final block not properly padded

- Stack Trace

javax.faces.FacesException: javax.crypto.BadPaddingException: Given final block not properly padded
at org.apache.myfaces.shared_impl.util.StateUtils.symmetric(StateUtils.java:373)
at org.apache.myfaces.shared_impl.util.StateUtils.symmetric(StateUtils.java:411)
at org.apache.myfaces.shared_impl.util.StateUtils.decrypt(StateUtils.java:291)
at org.apache.myfaces.shared_impl.util.StateUtils.reconstruct(StateUtils.java:240)
at org.apache.myfaces.renderkit.html.HtmlResponseStateManager.getSavedState(HtmlResponseStateManager.java:209)
at org.apache.myfaces.renderkit.html.HtmlResponseStateManager.getState(HtmlResponseStateManager.java:177)
at com.sun.faces.lifecycle.RestoreViewPhase.execute(RestoreViewPhase.java:48)
at org.apache.myfaces.lifecycle.RestoreViewExecutor.execute(RestoreViewExecutor.java:85)
at org.apache.myfaces.lifecycle.LifecycleImpl.executePhase(LifecycleImpl.java:103)
at org.apache.myfaces.lifecycle.LifecycleImpl.execute(LifecycleImpl.java:76)
at com.sun.faces.personal.framework.lifecycle.CpcLifecycleImpl.execute(CpcLifecycleImpl.java:3)
at javax.faces.webapp.FacesServlet.service(FacesServlet.java:148)
at weblogic.servlet.internal.StubSecurityHelper$StubSecurityServiceAction.run(StubSecurityHelper.java:12)
at weblogic.servlet.internal.StubSecurityHelper.invokeServlet(StubSecurityHelper.java:1283)
at weblogic.servlet.internal.ServletStubImpl.execute(ServletStubImpl.java:283)
at weblogic.servlet.internal.TailFilter.doFilter(TailFilter.java:26)
Basic PO attacks

POET vs JavaServer Faces

Padding Oracle in JSF frameworks

- Most JSF frameworks allow developers to turn off error messages. Then we can use the following simple trick:

Padding oracle in JSF frameworks when error-page is turned off

- Say we want to decrypt block $C_i$ of an encrypted view state $C_0|C_1|\ldots|C_{n-1}$, then we send $C_0|C_1|\ldots|C_{n-1}|C_{random}|C_i$ to the target.
  - Since Java ignores those extra blocks while decrypting and deserializing view states, it’s VALID padding if the target returns the same page as when the view state is unaltered.
  - And it’s probably INVALID padding if we see something else, e.g. a HTTP 500 error message.
Demo
POET vs Apache MyFaces

- Apache MyFaces latest version.
- This also works with SUN Mojarra and probably other JSF implementations.
Distributed cross-site PO attacks

- Only a single bit of information is necessary to exploit a padding oracle.
- Cross-domain information leakage bugs in web browsers can help.
- One example: `<img>` + `onerror()`/`onload()` events.
- `onLoad()` called: VALID padding; `onError()` called: INVALID padding.
Distributed cross-site PO attacks

- We’ve been able to exploit CAPTCHA schemes using a single Javascript program running in the local browser.
- Creating a distributed attack is as simple as injecting javascript code into popular websites.
- Distributed attacks allows easy creation of code books.
Demo
Distributed cross-site PO attacks

- Cracking CAPTCHA using Javascript running locally.
Using PO to encrypt

An introduction to CBC-R

- CBC-R turns a decryption oracle into an encryption oracle.

- We all know that CBC decryption works as following:

\[ P_i = d_K(C_i) \oplus C_{i-1} \]

\[ C_0 = IV \]

- We can use a padding oracle to get \( d_K(C_i) \), and we control \( C_{i-1} \). In other words, we can produce any \( P_i \) as we want.
Using PO to encrypt

How CBC-R works

CBC-R pseudocode

- choose a plaintext message $P_0|...|P_{n-1}$ that you want to encrypt.
  - pick a random $C_{n-1}$.
  - for $i = n-1$ down to 1: $C_{i-1} = P_i \oplus d_\delta(C_i)$
  - $IV = P_0 \oplus d_\delta(C_0)$
  - output $IV|C_0|C_1|...|C_{n-1}$. This ciphertext would be decrypted to $P_0|...|P_{n-1}$. 

T. Duong, J. Rizzo (VNSEC/HVA, NET) Padding Oracles Everywhere
Using PO to encrypt

CBC-R Without Controlling IV

- CBC-R allows us to encrypt any message, but if we cannot set the IV, then first plaintext block $P_0$ will be random and meaningless.

- If the victim expects the decrypted message to start with a standard header, then it will ignore the forged message constructed by CBC-R.

- We have not found generic way to overcome this limitation. However, we have found workarounds for particular cases.
Using captured ciphertexts as prefix

\[ P_{valid} = d_K(C_{captured} | IV_{CBC-R} | P_{CBC-R}). \]

- The block at the position of \( IV_{CBC-R} \) is still garbled.
- We can make the garbled block becomes part of some string that doesn’t affect the semantic of the message such as comment or textbox label.
Using PO to encrypt

CBC-R Without Controlling IV

Brute-forcing $C_0$

- CBC-R can produce many different ciphertexts that decrypted to the same plaintext block chain $P_{n-1},...,P_1$. The only difference is the first plaintext block which is computed as following:

$$P_0 = d_K(C_0) \oplus IV$$

- A valid header means that the first few bytes of $P_0$ must match some magic numbers. There are also systems that accept a message if the first byte of its $P_0$ matches its size.

- If this is the case, and if the message is short enough, we can try our luck by brute-forcing $C_0$. 
Using PO to encrypt

CBC-R Applications

sudo make me a CAPTCHA
Creating malicious JSF view states

- Which view states to create?
  - How to solve the garbled block problem?
ASP.NET’s design problems
Web.config (We steal this slide from Paul Craig)


- Web.config is the most important and sensitive file in ASP.NET.

- Guess what? It’s just a normal file inside the document root!
  - Usernames, passwords, connection strings.
  - MachineKey: validationKey (HMAC key) and decryptionKey (DES, 3DES, or AES key).
  - A lot of configuration information.

- All it takes is one file disclose vulnerability.
ASP.NET’s design problems

Cryptography

- MAC-then-Encrypt -> Decrypt-then-Verify -> still leak padding validity information.
- Crypto API does not authenticate messages by default -> there are some encryptions w/o using MAC at all.
- Fixed known IV.
- Same keys use to encrypt a lot of different things -> one padding oracle leads to full compromise.
- No easy way to generate keys:
  - People don’t change keys during the lifetime of applications.
  - People don’t change default keys in downloaded applications.
  - People even generate keys using online tools.
Padding oracles in ASP.NET

MAC-then-Encrypt: FAILED

- ASP.NET MAC-then-Encrypt these things:
  - ViewStates.
  - Form Authentication Tickets.
  - Anonymous Identification.
  - Role Cookies.

- In other words, universal padding oracles in every ASP.NET application!
Padding oracles in ASP.NET

No MAC at all: EPIC FAILED

- ASP.NET does not use MAC at all when encrypting:
  - WebResource

- Even better universal padding oracle!
Padding oracles in ASP.NET
How to detect padding oracles in ASP.NET

- Nice error messages, often turned on by default.
- No error message? Nice HTTP response statuses.
- Always the same 404 status? Nice timing information.
0-day: works for the latest versions of ASP.NET.

Target application: DotNetNuke (over 600,000 public installations).

POET -> remote code execution -> Cesar’s Token Kidnapping -> ROOT privilege on Windows.
What happened?

- This line is worth the price of admission: we found a way to read arbitrary files using CBC-R!

- You may need to optimize your CBC-R attack. Full paper and tools will be released soon!
Padding oracle attacks allow one to decrypt ciphertext without knowing the key.

We can use padding oracle attacks to crack CAPTCHA, and decrypt JSF view state, etc.

Distributed cross-site padding oracle attacks allow one to distributively build a code book to map all ciphertexts to corresponding plaintexts.

CBC-R turns a decryption oracle into an encryption oracle, and allow us to destroy ASP.NET security.