Practical Padding Oracle Attacks

Juliano Rizzo    Thai Duong

Black Hat Europe, 2010
Requisite

\[ \text{XOR} \]

\[ 0 \oplus 0 = 0 \]

\[ 0 \oplus 1 = 1 \]

\[ 1 \oplus 0 = 1 \]

\[ 1 \oplus 1 = 0 \]
Outline

1. Introduction
   - Review of CBC Mode
   - Padding Oracle attack

2. Finding padding oracles
   - Find potential padding oracles
   - Confirm the existence of padding oracles

3. Basic PO attacks
   - Cracking CAPTCHA
   - Decrypting JSF view states

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

- 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

- 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

- 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 wo
11 bytes of plaintext
r l d

PKCS5 Padding

Hello wo
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

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?

[Diagram showing the concept of a padding oracle with a attacker, valid padding question, cipher-text, secret key, and block cipher connections.

Valid Padding?

Happy Attacker

CIPHER-TEXT

YES / NO

Secret Key

IV

Block cipher

Padding Oracle
Padding oracle attack

What is a padding oracle?

- Adversary submits a CBC mode ciphertext \( C \) to oracle \( \mathcal{O} \).

- Oracle decrypts under fixed key \( K \) and checks correctness of padding.

- Oracle outputs VALID or INVALID according to correctness of padding:

\[
\mathcal{O}(C) = \begin{cases} 
0, & \text{invalid} \\
1, & \text{valid}
\end{cases}
\]
Padding oracle attack
What is a padding oracle?

- Adversary submits a CBC mode ciphertext $C$ to oracle $\mathcal{O}$.

- Oracle decrypts under fixed key $K$ and checks correctness of padding.

- Oracle outputs VALID or INVALID according to correctness of padding:

$$\mathcal{O}(C) = \begin{cases} 0, & \text{invalid} \\ 1, & \text{valid} \end{cases}$$
Padding oracle attack
What is a padding oracle?

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

$$\bar{\mathcal{O}}(C) = \begin{cases} 0, & \text{invalid} \\ 1, & \text{valid} \end{cases}$$
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 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 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

Secret Key

IV

Block cipher

... 256 tries max

YES

NO

NO

NO

CIPHER-TEXT

CIPHER-TEXT

CIPHER-TEXT

CIPHER-TEXT
Padding oracle attack

How to decrypt a block

Oracle CBC decryption process

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

- pick a few random bytes $r_1, \ldots, r_b$, and take $i = 0$.
- pick $r = r_1r_2\ldots r_{b-1}(r_b \oplus i)$.
- if $\bar{\delta}(r|y) = 0$ then increment $i$ and go back to previous step.
- replace $r_b$ by $r_b \oplus i$.
- for $n = b$ down to 2
  - take $r = r_1\ldots r_{b-n}(r_{b-1+1} \oplus 1)r_{b-n+2}\ldots r_b$
  - if $\bar{\delta}(r|y) = 0$ then stop and output $(r_{b-n+1} \oplus n)\ldots(r_b \oplus n)$
- output $r_b \oplus 1$. 

Juliano Rizzo, Thai Duong ()
Practical Padding Oracle Attacks
BH Europe 2010
Padding oracle attack

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 $\bar{\delta}(r|y) = 0$ then increment $i$ and go back to previous step.
- replace $r_b$ by $r_b \oplus i$.
- for $n = b$ down to 2
  - take $r = r_1 \ldots r_{b-n} (r_{b-1+n} \oplus 1) r_{b-n+2} \ldots r_b$
  - if $\bar{\delta}(r|y) = 0$ then stop and output $(r_{b-n+1} \oplus n) \ldots (r_b \oplus n)$
- output $r_b \oplus 1$. 
Padding oracle attack

Last byte decryption algorithm

1. pick a few random bytes $r_1, \ldots, r_b$, and take $i = 0$.
2. pick $r = r_1 r_2 \ldots r_{b-1} (r_b \oplus i)$.
3. if $\tilde{d}(r | y) = 0$ then increment $i$ and go back to previous step.
4. replace $r_b$ by $r_b \oplus i$.
5. for $n = b$ down to 2
   1. take $r = r_1 \ldots r_{b-n} (r_{b-n+1} \oplus 1) r_{b-n+2} \ldots r_b$
   2. if $\tilde{d}(r | y) = 0$ then stop and output $(r_{b-n+1} \oplus n) \ldots (r_b \oplus n)$
6. output $r_b \oplus 1$. 
Padding oracle attack

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 $\delta(r | y) = 0$ then increment $i$ and go back to previous step.
- Replace $r_b$ by $r_b \oplus i$.

- For $n = b$ down to 2
  - Take $r = r_1 \ldots r_{b-n}(r_{b-n+1} \oplus 1)r_{b-n+2} \ldots r_b$
  - If $\delta(r | y) = 0$ then stop and output $(r_{b-n+1} \oplus n) \ldots (r_b \oplus n)$

- Output $r_b \oplus 1$. 

Juliano Rizzo, Thai Duong ()
Padding oracle attack

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 \( \tilde{\delta}(r|y) = 0 \) 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 \( \tilde{\delta}(r|y) = 0 \) then stop and output \( (r_{b-n+1} \oplus n) \ldots (r_b \oplus n) \)
- output \( r_b \oplus 1 \).
Padding oracle attack

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 $\tilde{D}(r|y) = 0$ then increment $i$ and go back to previous step.
- replace $r_b$ by $r_b \oplus i$.
- for $n = b$ down to 2
  - take $r = r_1 \ldots r_{b-n}(r_{b-1+1} \oplus 1)r_{b-n+2} \ldots r_b$
  - if $\tilde{D}(r|y) = 0$ then stop and output $(r_{b-n+1} \oplus n) \ldots (r_b \oplus n)$
- output $r_b \oplus 1$. 
Demo
Exploiting RubyOnRails ActiveSupport::MessageEncryptor

- Since RubyOnRails 2.3, to provide a simple way to encrypt information.
- Vulnerability: encrypt and decrypt functions.
- Use encrypt_and_sign and decrypt_and_verify instead.
Demo

Exploiting RubyOnRails ActiveSupport::MessageEncryptor

- Since RubyOnRails 2.3, to provide a simple way to encrypt information.

- Vulnerability: `encrypt` and `decrypt` functions.

- Use `encrypt_and_sign` and `decrypt_and_verify` instead.
Demo
Exploiting RubyOnRails ActiveSupport::MessageEncryptor

- Since RubyOnRails 2.3, to provide a simple way to encrypt information.

- Vulnerability: `encrypt` and `decrypt` functions.

- Use `encrypt_and_sign` and `decrypt_and_verify` instead.
Finding potential padding oracles

Blackbox testing

- Crawl the target to find BASE64 strings that look like a ciphertext.
- Replace a byte in the last block of the ciphertext by a random value, and send to the target.
- See if there is any error message. Even a blank page is enough information.
Crawl the target to find BASE64 strings that look like a ciphertext.

Replace a byte in the last block of the ciphertext by a random value, and send to the target.

See if there is any error message. Even a blank page is enough information.
Finding potential padding oracles

Blackbox testing

- Crawl the target to find BASE64 strings that look like a ciphertext.

- Replace a byte in the last block of the ciphertext by a random value, and send to the target.

- See if there is any error message. Even a blank page is enough information.
Finding potential padding oracles

Google hacking
Finding potential padding oracles
Source code auditing

- Look for code that imports low level cryptography libraries.
- Look for known source code keywords like `javax.crypto.BadPaddingException`.
- Look for routines that perform encryption and decryption that have some code to handle error while decrypting.
Finding potential padding oracles

Source code auditing

- Look for code that imports low level cryptography libraries.

- Look for known source code keywords like `javax.crypto.BadPaddingException`.

- Look for routines that perform encryption and decryption that have some code to handle error while decrypting.
Finding potential padding oracles

Source code auditing

- Look for code that imports low level cryptography libraries.
- Look for known source code keywords like `javax.crypto.BadPaddingException`.
- Look for routines that perform encryption and decryption that have some code to handle error while decrypting.
Confirm the existence of padding oracles

Determine the block size $b$

- All padding oracle attacks need a correct $b$.

- Most common block sizes are 8 and 16 bytes. Of course we can use trial and error.

How to determine the block size

- if $\text{len}(C) \% 16 = 8$, then stop and output 8.

- take $y = C[-16:]$, i.e. $y$ is the last sixteen bytes of $C$.

- if $\delta(C|y) = 1$, then stop and output 8.

- output 16.
Confirm the existence of padding oracles

Determine the block size $b$

- All padding oracle attacks need a correct $b$.
- Most common block sizes are 8 and 16 bytes. Of course we can use trial and error.

How to determine the block size

- if $\text{len}(C) \% 16 = 8$, then stop and output 8.
- take $y = C[-16:]$, i.e. $y$ is the last sixteen bytes of $C$.
- if $\delta(C|y) = 1$, then stop and output 8.
- output 16.
Confirm the existence of padding oracles

Determine the block size $b$

- All padding oracle attacks need a correct $b$.

- Most common block sizes are 8 and 16 bytes. Of course we can use trial and error.

**How to determine the block size**

- if $\text{len}(C) \% 16 = 8$, then stop and output 8.

- take $y = C[-16:]$, i.e. $y$ is the last sixteen bytes of $C$.

- if $\delta(C|y) = 1$, then stop and output 8.

- output 16.
Confirm the existence of padding oracles
Determine the block size $b$

- All padding oracle attacks need a correct $b$.
- Most common block sizes are 8 and 16 bytes. Of course we can use trial and error.

How to determine the block size
- if $\text{len}(C) \equiv 16 \equiv 8$, then stop and output 8.
- take $y = C[-16:]$, i.e. $y$ is the last sixteen bytes of $C$.
- if $\delta(C|y) = 1$, then stop and output 8.
- output 16.
Confirm the existence of padding oracles

Determine the block size $b$

- All padding oracle attacks need a correct $b$.
- Most common block sizes are 8 and 16 bytes. Of course we can use trial and error.

How to determine the block size

- if $\text{len}(C) \% 16 = 8$, then stop and output 8.
- take $y = C[-16:]$, i.e. $y$ is the last sixteen bytes of $C$.
- if $\delta(C|y) = 1$, then stop and output 8.
- output 16.
Confirm the existence of padding oracles

Determine the block size $b$

- All padding oracle attacks need a correct $b$.
- Most common block sizes are 8 and 16 bytes. Of course we can use trial and error.

How to determine the block size

- if $\text{len}(C) \% 16 = 8$, then stop and output 8.
- take $y = C[-16:]$, i.e. $y$ is the last sixteen bytes of $C$.
- if $\delta(C|y) = 1$, then stop and output 8.
- output 16.
Confirm the existence of padding oracles

- We want the target to reveal as many different reactions to the modified ciphertexts as possible.

- Most important: know when the padding is VALID, and when it’s INVALID.

- POET a.k.a Padding Oracle Exploitation Tool will be released right after BH Europe 2010.
Confirm the existence of padding oracles

- We want the target to reveal as many different reactions to the modified ciphertexts as possible.

- Most important: know when the padding is VALID, and when it’s INVALID.

- POET a.k.a Padding Oracle Exploitation Tool will be released right after BH Europe 2010.
Confirm the existence of padding oracles

- We want the target to reveal as many different reactions to the modified ciphertexts as possible.

- Most important: know when the padding is VALID, and when it’s INVALID.

- POET a.k.a Padding Oracle Exploitation Tool will be released right after BH Europe 2010.
Confirm the existence of padding oracles

- Want to write your own tool to detect padding oracle? Follow this guideline (which is based on the algorithm in slide 22):
  - Determine the block size $b$.
  - Pick a few random words $r_1, \ldots, r_b$, and take $i = 0$.
  - Pick $r = r_1 r_2 \ldots r_{b-1} (r_b \oplus i)$.
  - Send $r|y$ to the target, where $y$ is a valid ciphertext block. Record the value of $i$, content length, and content type of the response. Increment $i$, and go back to step 3 until $i > 255$.
  - Now you have 256 responses. If all of them are the same, then the target is not easily showing you that it is vulnerable to padding oracle attack.
  - Otherwise, look at each value of $i$ where the responses are different from the rest. Examine carefully each response to see what happened.
Confirm the existence of padding oracles

- Want to write your own tool to detect padding oracle? Follow this guideline (which is based on the algorithm in slide 22):
  - Determine the block size $b$.
  - Pick a few random words $r_1, ..., r_b$, and take $i = 0$.
  - Pick $r = r_1r_2...r_{b-1}(r_b \oplus i)$.
  - Send $r|y$ to the target, where $y$ is a valid ciphertext block. Record the value of $i$, content length, and content type of the response. Increment $i$, and go back to step 3 until $i > 255$.
  - Now you have 256 responses. If all of them are the same, then the target is not easily showing you that it is vulnerable to padding oracle attack.
  - Otherwise, look at each value of $i$ where the responses are different from the rest. Examine carefully each response to see what happened.
Confirm the existence of padding oracles

- Want to write your own tool to detect padding oracle? Follow this guideline (which is based on the algorithm in slide 22):
  - Determine the block size $b$.
  - Pick a few random words $r_1, \ldots, r_b$, and take $i = 0$.
  - Pick $r = r_1 r_2 \ldots r_{b-1} (r_b \oplus i)$.
  - Send $r | y$ to the target, where $y$ is a valid ciphertext block. Record the value of $i$, content length, and content type of the response. Increment $i$, and go back to step 3 until $i > 255$.
  - Now you have 256 responses. If all of them are the same, then the target is not easily showing you that it is vulnerable to padding oracle attack.
  - Otherwise, look at each value of $i$ where the responses are different from the rest. Examine carefully each response to see what happened.
Confirm the existence of padding oracles

- Want to write your own tool to detect padding oracle? Follow this guideline (which is based on the algorithm in slide 22):
  - Determine the block size $b$.
  - Pick a few random words $r_1, \ldots, r_b$, and take $i = 0$.
  - Pick $r = r_1 r_2 \ldots r_{b-1} (r_b \oplus i)$.
  - Send $r|y$ to the target, where $y$ is a valid ciphertext block. Record the value of $i$, content length, and content type of the response. Increment $i$, and go back to step 3 until $i > 255$.
  - Now you have 256 responses. If all of them are the same, then the target is not easily showing you that it is vulnerable to padding oracle attack.
  - Otherwise, look at each value of $i$ where the responses are different from the rest. Examine carefully each response to see what happened.

Juliano Rizzo, Thai Duong (Practical Padding Oracle Attacks BH Europe 2010)
Confirm the existence of padding oracles

- Want to write your own tool to detect padding oracle? Follow this guideline (which is based on the algorithm in slide 22):
  - Determine the block size $b$.
  - Pick a few random words $r_1,...,r_b$, and take $i = 0$.
  - Pick $r = r_1r_2...r_{b-1}(r_b \oplus i)$.
  - Send $r|y$ to the target, where $y$ is a valid ciphertext block. Record the value of $i$, content length, and content type of the response. Increment $i$, and go back to step 3 until $i > 255$.
  - Now you have 256 responses. If all of them are the same, then the target is not easily showing you that it is vulnerable to padding oracle attack.
  - Otherwise, look at each value of $i$ where the responses are different from the rest. Examine carefully each response to see what happened.
Confirm the existence of padding oracles

- Want to write your own tool to detect padding oracle? Follow this guideline (which is based on the algorithm in slide 22):
  - Determine the block size $b$.
  - Pick a few random words $r_1, \ldots, r_b$, and take $i = 0$.
  - Pick $r = r_1 r_2 \ldots r_{b-1} (r_b \oplus i)$.
  - Send $r|y$ to the target, where $y$ is a valid ciphertext block. Record the value of $i$, content length, and content type of the response. Increment $i$, and go back to step 3 until $i > 255$.
  - Now you have 256 responses. If all of them are the same, then the target is not easily showing you that it is vulnerable to padding oracle attack.
  - Otherwise, look at each value of $i$ where the responses are different from the rest. Examine carefully each response to see what happened.
A broken CAPTCHA system

- \( ERC = e_{K,IV}(rand()) \).

- \(...<\text{img src}="/captcha?token=ERC" />...\)

- \( ERC \) 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.
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.
Cracking CAPTCHA

A broken CAPTCHA system

- \( ERC = e_{K,IV}(rand()) \).

- \( \ldots <img \text{ src="/captcha?token=ERC"} /> \ldots \)

- \( ERC \) 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.
Cracking CAPTCHA

A broken CAPTCHA system

- \( ERC = e_{K,IV}(rand()) \).
- \( \ldots <\text{img src="captcha?token=ERC"} /> \ldots \)
- \( ERC \) 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.
Cracking 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.
Cracking 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.
Cracking 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.
Cracking 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.
Cracking 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.
Cracking CAPTCHA

CAPTCHA with secret IV

- Since $P_0 = IV \oplus d(\theta(C_0))$, we need to know the IV to get $P_0$.

- If the IV is secret, we can’t know $P_0$, thus can’t crack CAPTCHA systems whose $P_0$ contains part of the random code.

- The solution is: $IV = Human \oplus d(\theta(C_0))$, where $Human$ denotes that somebody reads $P_0$ from the CAPTCHA image.
Cracking CAPTCHA
CAPTCHA with secret IV

- Since $P_0 = IV \oplus d_\delta(C_0)$, we need to know the IV to get $P_0$.

- If the IV is secret, we can’t know $P_0$, thus can’t crack CAPTCHA systems whose $P_0$ contains part of the random code.

- The solution is: $IV = Human \oplus d_\delta(C_0)$, where Human denotes that somebody reads $P_0$ from the CAPTCHA image.
Cracking CAPTCHA

CAPTCHA with secret IV

- Since $P_0 = IV \oplus d_\theta(C_0)$, we need to know the IV to get $P_0$.

- If the IV is secret, we can’t know $P_0$, thus can’t crack CAPTCHA systems whose $P_0$ contains part of the random code.

- The solution is: $IV = Human \oplus d_\theta(C_0)$, where $Human$ denotes that somebody reads $P_0$ from the CAPTCHA image.
Demo
Cracking CAPTCHA

- Target: http://www.bidz.com
- We can control the IV.
Demo
Cracking CAPTCHA

- Target: http://www.bidz.com
- We can control the IV.
Decryption of JSF view states

Introduction

- 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.
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.
Decryption of JSF view states

Introduction

- 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.
Decryption JSF view states

Introduction

- 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.
Decrypting JSF view states
Padding oracle in 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.
Decrypted JSF view states
Padding oracle in 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.
Decrypting JSF view states

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:78)
at org.apache.myfaces.renderkit.html.HtmlResponseStateManager.getSavedState(HtmlResponseStateManager.java:77)
at com.sun.faces.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.cpc.personal.framework.lifecycle.CpcLifecycleImpl.execute(CpcLifecycleImpl.java:3)

javax.faces.webapp.FacesServlet.service(FacesServlet.java:148)

 ...

...
Decryption of JSF view states

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 $C_0|C_1|...|C_{n-1}$, then we send $C_0|C_1|...|C_{n-1}|C_{\text{random}}|C_i$ 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.
Decryption of JSF view states

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 $C_0|C_1|...|C_{n-1}$, then we send $C_0|C_1|...|C_{n-1}|C_{random}|C_i$ 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.
Decryption of JSF view states

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 $C_0|C_1|...|C_{n-1}$, then we send $C_0|C_1|...|C_{n-1}|C_{random}|C_i$ 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
Decryption JSF view states

- Apache MyFaces latest version.
- This also works with SUN Mojarra and probably other JSF implementations.
Demo
Decrypted JSF view states

- Apache MyFaces latest version.
- This also works with SUN Mojarra and probably other JSF implementations.
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
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
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: $C_{i-1} = P_i \oplus d_0(C_i)$
- $IV = P_0 \oplus d_0(C_0)$
- output $IV|C_0|C_1|...|C_{n-1}$. This ciphertext would be decrypted to $P_0|...|P_{n-1}$. 
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 $P_0|...|P_{n-1}$. 
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 $P_0|...|P_{n-1}$. 
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 $P_0|...|P_{n-1}$.
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: $C_{i-1} = P_i \oplus d_0(C_i)$
- $IV = P_0 \oplus d_0(C_0)$
- output $IV | C_0 | C_1 | ... | C_{n-1}$. This ciphertext would be decrypted to $P_0 | ... | P_{n-1}$.
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 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 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 PO to encrypt

CBC-R Without Controlling IV

Using captured ciphertexts as prefix

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

- 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

Using captured ciphertexts as prefix

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

- 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

Using captured ciphertexts as prefix

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

- 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 follows:

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

- 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 $C_0$. 

Juliano Rizzo, Thai Duong ()

Practical Padding Oracle Attacks

BH Europe 2010 35 / 41
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 Without Controlling IV

Brute-forcing $C_0$

- CBC-R can produce many different ciphertexts that decrypted to the same plaintext block chain $P_{n-1}, \ldots, P_1$. 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
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?
Creating malicious JSF view states

- Which view states to create?
- How to solve the garbled block problem?
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

- 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

- 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

- 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.
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.
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.
- The javascript will be released in a few days.
Demo
Distributed cross-site PO attacks

- Cracking CAPTCHA using Javascript running locally.
- The javascript will be released in a few days.
Demo

Distributed cross-site PO attacks

- Cracking CAPTCHA using Javascript running locally.
- The javascript will be released in a few days.
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.

CBC-R turns a decryption oracle into an encryption oracle, and allow us to create malicious JSF view states.

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

CBC-R turns a decryption oracle into an encryption oracle, and allow us to create malicious JSF view states.

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

- 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.

- CBC-R turns a decryption oracle into an encryption oracle, and allow us to create malicious JSF view states.

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

CBC-R turns a decryption oracle into an encryption oracle, and allow us to create malicious JSF view states.

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