Fun with Padding Oracles

Justin Clarke
OWASP London Chapter

Introduction

- Circa 2002
- Juliano Rizzo/Thai Duong (Black Hat Europe 2010) repopularised with POET tool
- ASP.NET vectors for exploitation

Understanding a Padding Oracle

- Cryptographic padding (PKCS#5 in this case)
  - N blocks of "N"

	BLOCK 1							
	1	2	3	4	5	6	7	8
Plaintext	A	P	P	L	E			
Padded Plaintext	A	P	P	L	E	0x03	0x03	0x03

Understanding a Padding Oracle

- Cryptographic padding (PKCS#5 in this case)
  - N blocks of "N"

	BLOCK 1							
	1	2	3	4	5	6	7	8
Plaintext	A	V	O	C	A	D	O	
Padded Plaintext	A	V	O	C	A	D	O	0x01

Understanding a Padding Oracle

- Cryptographic padding (PKCS#5 in this case)
  - N blocks of "N"

	BLOCK 1								BLOCK 2							
	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8
Plaintext	P	L	A	N	T	A	I	N								
Padded Plaintext	P	L	A	N	T	A	I	N	0x08	0x08	0x08	0x08	0x08	0x08	0x08	0x08

Understanding a Padding Oracle

- Cryptographic padding (PKCS#5 in this case)
  - N blocks of "N"

	BLOCK 1								BLOCK 2							
	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8
Plaintext	P	A	S	S	I	O	N	F	R	U	I	T				
Padded Plaintext	P	A	S	S	I	O	N	F	R	U	I	T	0x04	0x04	0x04	0x04

The Basic Padding Oracle Attack

- Scenario
  - Some encrypted content (say, a user token containing the username and roles for a user) is passed in
  - Encrypted CBC mode with a unique IV (prepended to ciphertext)
  - Application responds in three ways
    - Valid ciphertext and valid plaintext received (e.g. 200 OK)
    - Invalid ciphertext, throws error (e.g. 500)
    - Valid ciphertext and invalid plaintext, throws error (e.g. 200 OK)

http://sampleapp/home.jsp?
UID=7B216A634951170FF851D6CC68FC9537858795A28ED4AAC6

The Basic Padding Oracle Attack

- Cookie contains "BRIAN;12;1;"

	INITIALISATION VECTOR							
	1	2	3	4	5	6	7	8
Plaintext	-	-	-	-	-	-	-	-
Padded Plaintext	-	-	-	-	-	-	-	-
Ciphertext (HEX)	0x07B	0x021	0x6A	0x63	0x49	0x51	0x17	0x0F

	BLOCK 1							
	1	2	3	4	5	6	7	8
Plaintext	B	R	I	A	N	;	1	2
Padded Plaintext	B	R	I	A	N	;	1	2
Ciphertext (HEX)	0xF8	0x51	0xD6	0xCC	0x68	0xFC	0x95	0x37

	BLOCK 2							
	1	2	3	4	5	6	7	8
Plaintext	;	1	;					
Padded Plaintext	;	2	;	0x05	0x05	0x05	0x05	0x05
Ciphertext (HEX)	0x85	0x87	0x95	0xA2	0x8E	0xD4	0xAA	0xC6

Encryption

	BLOCK 1 of 2								BLOCK 2 of 2							
	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8
Initialization Vector	0x7B	0x21	0x6A	0x63	0x49	0x51	0x17	0x0F	0xF8	0x51	0xD6	0xCC	0x68	0xFC	0x95	0x37
	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕
Plain-Text (Padded)	B	R	I	A	N	;	1	2	;	1	;	0x05	0x05	0x05	0x05	0x05
	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
Intermediary Value (HEX)	0x39	0x73	0x23	0x22	0x07	0x6A	0x26	0x3D	0xC3	0x60	0xED	0xC9	0x6D	0xF9	0x90	0x32
	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
	TRIPLE DES								TRIPLE DES							
	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
Encrypted Output (HEX)	0xF8	0x51	0xD6	0xCC	0x68	0xFC	0x95	0x37	0x85	0x87	0x95	0xA2	0x8E	0xD4	0xAA	0xC6

Decryption

	BLOCK 1 of 2								BLOCK 2 of 2							
	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8
Encrypted Input (HEX)	0xF8	0x51	0xD6	0xCC	0x68	0xFC	0x95	0x37	0x85	0x87	0x95	0xA2	0x8E	0xD4	0xAA	0xC6
	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
	TRIPLE DES								TRIPLE DES							
	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
Intermediary Value (HEX)	0x39	0x73	0x23	0x22	0x07	0x6a	0x26	0x3D	0xC3	0x60	0xED	0xC9	0x6D	0xF9	0x90	0x32
	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕
Initialization Vector	0x7B	0x21	0x6A	0x63	0x49	0x51	0x17	0x0F	0xF8	0x51	0xD6	0xCC	0x68	0xFC	0x95	0x37
	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
Plain-Text (Padded)	B	R	I	A	N	;	1	2	;	1	;	0x05	0x05	0x05	0x05	0x05

VALID PADDING

The Attack

- Isolate the first block by sending in a value with a NULL IV
  - Request: http://sampleapp/home.jsp?UID=00000000000000F851D6CC68FC9537
  - Response: 500 - Internal Server Error

The Attack (cont)

	BLOCK 1 of 1							
	1	2	3	4	5	6	7	8
Encrypted Input	0xF8	0x51	0xD6	0xCC	0x68	0xFC	0x95	0x37
	↓	↓	↓	↓	↓	↓	↓	↓
	TRIPLE DES							
	↓	↓	↓	↓	↓	↓	↓	↓
Intermediary Value	0x39	0x73	0x23	0x22	0x07	0x6a	0x26	0x3D
	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕
Initialization Vector	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00
	↓	↓	↓	↓	↓	↓	↓	↓
Decrypted Value	0x39	0x73	0x23	0x22	0x07	0x6a	0x26	0x3D

INVALID PADDING

The Attack (cont)

- Increment the last byte of the IV by 1
  - Request: http://sampleapp/home.jsp?UID=00000000000000001F851D6CC68FC9537
  - Response: 500 - Internal Server Error

The Attack (cont)

	BLOCK 1 of 1							
	1	2	3	4	5	6	7	8
Encrypted Input	0xF8	0x51	0xD6	0xCC	0x68	0xFC	0x95	0x37
	↓	↓	↓	↓	↓	↓	↓	↓
	TRIPLE DES							
	↓	↓	↓	↓	↓	↓	↓	↓
Intermediary Value	0x39	0x73	0x23	0x22	0x07	0x6a	0x26	0x3D
	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕
Initialization Vector	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x01
	↓	↓	↓	↓	↓	↓	↓	↓
Decrypted Value	0x39	0x73	0x23	0x22	0x07	0x6a	0x26	0x3C

INVALID PADDING

The Attack (cont)

- Iterate incrementing the last byte of the IV by 1 until we get something different
  - Request: http://sampleapp/home.jsp?UID=00000000000003CF851D6CC68FC9537
  - Response: 200 - OK

The Attack (cont)

	Block 1 of 1							
	1	2	3	4	5	6	7	8
Encrypted Input	0xF8	0x51	0xD6	0xCC	0x68	0xFC	0x95	0x37
	↓	↓	↓	↓	↓	↓	↓	↓
	TRIPLE DES							
	↓	↓	↓	↓	↓	↓	↓	↓
Intermediary Value	0x39	0x73	0x23	0x22	0x07	0x6a	0x26	0x3D
	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕
Initialization Vector	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x3C
	↓	↓	↓	↓	↓	↓	↓	↓
Decrypted Value	0x39	0x73	0x23	0x22	0x07	0x6a	0x26	0x01

VALID PADDING

The Attack (cont)

- Now we know the intermediate (plaintext, prior to XOR) value

If [Intermediary Byte] ^ 0×3C == 0×01,
then [Intermediary Byte] == 0×3C ^ 0×01,
so [Intermediary Byte] == 0×3D

The Attack (cont)

- Since we can now derive what the value of the last byte is, we can go after the next byte
  - Request: http://sampleapp/home.jsp?UID=00000000000003DF851D6CC68FC9537
  - Response: 500 - Internal Server Error

The Attack (cont)

	Block 1 of 1							
	1	2	3	4	5	6	7	8
Encrypted Input	0xF8	0x51	0xD6	0xCC	0x68	0xFC	0x95	0x37
	↓	↓	↓	↓	↓	↓	↓	↓
	TRIPLE DES							
	↓	↓	↓	↓	↓	↓	↓	↓
Intermediary Value	0x39	0x73	0x23	0x22	0x07	0x6a	0x26	0x3D
	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕
Initialization Vector	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x3F
	↓	↓	↓	↓	↓	↓	↓	↓
Decrypted Value	0x39	0x73	0x23	0x22	0x07	0x6a	0x26	0x02

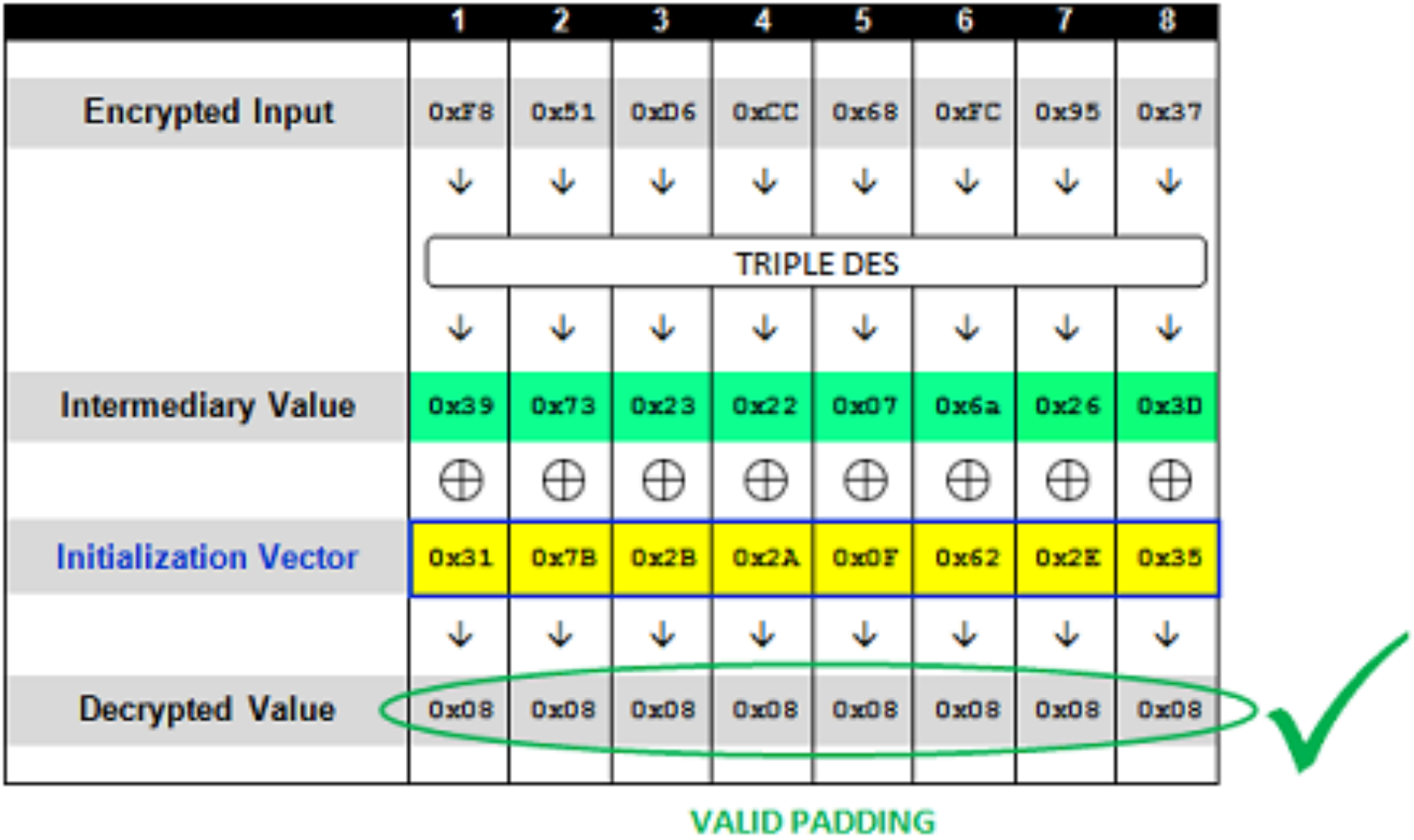
INVALID PADDING

The Attack (cont)

	1	2	3	4	5	6	7	8
Encrypted Input	0xF8	0x51	0xD6	0xCC	0x68	0xFC	0x95	0x37
	↓	↓	↓	↓	↓	↓	↓	↓
	TRIPLE DES							
	↓	↓	↓	↓	↓	↓	↓	↓
Intermediary Value	0x39	0x73	0x23	0x22	0x07	0x6a	0x26	0x3D
	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕
Initialization Vector	0x00	0x00	0x00	0x00	0x00	0x00	0x24	0x3F
	↓	↓	↓	↓	↓	↓	↓	↓
Decrypted Value	0x39	0x73	0x23	0x22	0x07	0x26	0x02	0x02

VALID PADDING

The Attack (cont)



The Attack (cont)

- Since we now know the intermediate values, we can now XOR with the original IV

	1	2	3	4	5	6	7	8
IV	0x07B	0x021	0x6A	0x63	0x49	0x51	0x17	0x0F
	XOR	XOR	XOR	XOR	XOR	XOR	XOR	XOR
Intermediate	0x39	0x73	0x23	0x22	0x07	0x6A	0x26	0x3D
Plaintext	B	R	I	A	N	;	1	2

Demo

- Padbuster – ASP.NET exploit (patch released by Microsoft 28 September 2010)

THANK YOU!