Speaking with Cryptographic Oracles

Daniel “unicornFurnace” Crowley
Application Security Consultant, Trustwave - Spiderlabs
The Speaker and the Presentation

A quick introduction and a few distinctions
The Speaker

• Daniel Crowley
• Web application security d00d
• IANAC (I am not a cryptographer)

dcrowley@trustwave.com
@dan_crowley
The Presentation Topic

- **Finding and exploiting:**
  - Encryption Oracles
  - Decryption Oracles
  - Padding Oracles

- **With little to no cryptographic knowledge**
  - More crypto knowledge, more useful attacks
NOT the Presentation Topic

• The Oracle
  • We are not being harvested for energy by robot overlords
    • Maybe

• ORACLE
  • If you Google “<any crypto word> oracle” it’s all you find

• Google, the Internet Oracle
  • While awesome, not what we’re talking about
NOT the Presentation Topic

- Crypto g00r00s like Adi Shamir
  - While also awesome and totally related, not the topic
- New attacks on old crypto
  - Mistakes are easy enough to make in implementation
- How Padding Oracle attacks work
  - Too much time to explain
  - Too many good resources
For the people playing drinking games

- **APT iPad**
  - APT China, cyber-war
- **Cloud mobile botnet**
  - Cloud cloud Twilight APT
  - Sun Tzu
    - RSA HBGary botnet cloud
    - APT
- **Cyber-war?**
- **LulzSec???

APT China cyberwar weebaboo, cloud mobile LulzSec.
Primer on Cryptographic Terms

And some basic mistakes
Very basic terms

• **Cipher**
  • A system for scrambling and unscrambling data to protect it

• **Key**
  • A variable used to permute the cipher

• **Initialization Vector**
  • A second variable used to randomize the cipher

• **Plaintext**
  • The data in readable form

• **Ciphertext**
  • The data in unreadable form

• **Encryption**
  • Turning something you can read into something you can’t

• **Decryption**
  • Turning something you can’t read into something you can
Stream and Block ciphers

Stream
- Encrypt one character at a time
- Key is used to generate pseudo-random numbers
- Those numbers are used to transform plaintext to ciphertext

Block
- Encrypt X characters at a time
  - X is the block size
- Key is used to directly transform plaintext to ciphertext
Very basic mistakes

- **Using a keyless cipher**
  - Completely insecure if cipher is ever discovered
- **Reusing keys and/or IVs**
  - Makes Oracle attacks far more dangerous
  - IV reuse can seriously weaken stream ciphers
    - Think WEP
- **Leaking data from crypto operations**
  - Foundation for Oracle attacks
What is an Oracle?

A system which takes queries and provides answers

- Queries might be
  - Plaintext
  - Ciphertext

- Answers might be
  - Corresponding plaintext
  - Corresponding ciphertext
  - Info about operation
  - Sample from PRNG
Seek the Oracle

How to identify cryptographic Oracles
From a black-box perspective
Decryption Oracles: Identify input

- **Identify where encrypted input occurs**
  - Identify all points of user input
    - For Web apps: GET, POST, URL, Cookie, headers
  - Identify those which may be encrypted
    - Encrypted data is generally encoded
      - Base64
      - ASCII hex
      - URL encoding
    - Decoded data is likely encrypted if seemingly random
    - Modification of values may result in decryption-related errors
Decryption Oracles: Find decrypted output

- May be reflected
  - Normal output
  - Error
- May be given in later response
- May be inferred from modified output
- May be stored and not shown
  - Additional vulnerabilities may reveal output
Decryption Oracles: An example

**Scenario**

- **Consider “GetPage.php?file=<encrypted_stuff>”**
  - Opens a file to be included based on encrypted input
    - Allows for quick page additions
    - Prevents file inclusion attacks...
    - Assumes properly encrypted input is sanitary
  - Errors are verbose

**Usage**

- **Feed the script some ciphertext**
  - Record the “file” the error tells you wasn’t found
Encryption Oracles: Find encrypted data

- Often found in
  - Cookies
  - Hidden variables
  - Databases
  - File resident data

Flickr Creative Commons – Gideon van der Stelt
Encryption Oracles: Determine point of entry

- **Frequently encrypted data**
  - Client-side state variables
  - Passwords
  - Financial data
  - Anything sufficiently sensitive

- **Being encrypted is not enough**
  - We need to be able to manipulate it
  - And see the ciphertext
Encryption Oracles: An example

Scenario

- Consider “auth” cookie, encrypted
  - Username + “:” + password_hash + “:” + timestamp
- Assume usernames can’t contain “:” character
  - No delimiter injection 😞
- Timestamp to control expiration

Usage

- Register with any username, log in
- Copy cookie value and replace any encrypted input with it
  - Can’t use colons or control suffix
    - Might not matter
Padding Oracles

- Input must be encrypted
- Must be a padded block cipher
- Valid vs invalid padding is distinguishable

- Padding Oracles are essentially decryption oracles
  - Using the CBC-R technique they are also encryption Oracles
    - May be limited in that the first block will be garbled
Exploiting Cryptographic Oracles

Against bad crypto and bad crypto usage
Attack 0: Crypto recon examples

- **Check for static key, IV, and deterministic cipher**
  - Encrypt the same plaintext twice
  - Check to see if they are identical

- **Check for stream vs. block ciphers**
  - Encrypt plaintexts of various sizes
  - Compare plaintext size to ciphertext size

- **Check for ECB block cipher mode**
  - Encrypt repeating plaintext blocks
  - Look for repetitive ciphertext
Attack 1: Bad Algorithms

- Occasionally, people try to make their own algorithms
  - And they’re not cryptographers
    - And it doesn’t end well

Real homespun crypto seen in the wild:

- Each character is replaced with a “random” but unique selection of two or three characters
- Characters are separated by the letter “K”

“hello” might become “KqIKefKPPrPKPrPKuJXK”
Attack 1: Bad Algorithms

Is there substitution?

Submit "AAAA" : Get "KLoKLoKLoKLoK"

- There is!
- We can already see patterns, too

Is there transposition?

Submit "AABB" : Get "KLoKLoKaBeKaBeK"

- No transposition
- We can see more patterns
- The “K” seems to be a delimiter
- Substitution doesn’t change on position
  - One replacement per letter
Attack 1: Bad Algorithms

Submit "BABA" : Get "KaBeKLoKaBeKLoK"

• Exactly what we expected

Submit "abcdefghi…XYZ0123456789" : Get entire key!

• We now submit one of every character in sequence

• The Oracle tells us what each maps to
Attack 1 and a half: Revenge of Bad Algorithms

Others use a simple xor operation to encrypt data

\[ P \text{ xor } B = C \]
\[ C \text{ xor } B = P \]
\[ C \text{ xor } P = B \]
Attack 1.75: Bride of Bad Algorithms

For some simple ciphers like xor

Encryption = Decryption

THUS

Encryption Oracle = Decryption Oracle

THUS

Such ciphers are made completely useless by leaking output

THUS

For God’s sake stop using xor
Attack 1: Bad Algorithms

DEMO
Attack 2: Trusted Encrypted Input

- People tend to reuse keys and IVs
  - If we can encrypt arbitrary data in one place
  - It may work in another

- If devs don’t think you can mess with input
  - They probably won’t sanitize it
  - Encrypted inputs with MAC aren’t totally tamper-proof
Attack 2: Trusted Encrypted Input

- Encrypted password with MAC in cookie
  - Checked against database on each request needing auth

- Find encryption Oracle with the same keys & IV
  - Use encryption Oracle to encrypt `or 1=1--`
  - Plug resulting value into cookie
  - Laugh all the way to the bank
Attack 2: Trusted Encrypted Input

DEMO
Attack 3: Let the client have it, it’s encrypted

I. Find a decryption Oracle
II. Find encrypted data
III. Decrypt that sucka
IV. ?????
V. PROFIT!!

This attack also relies on key/IV reuse
Attack 3: Let the client have it, it’s encrypted

DEMO
What encryption?

- If you can find
  - An encryption Oracle
  - A decryption Oracle
- You can encrypt or decrypt any data
  - As long as keys and IVs are reused
    - Algorithm doesn’t matter
    - Padding doesn’t matter
    - Cipher mode doesn’t matter

All encryption which uses the same key and IV is now useless
Questions?

Daniel Crowley
Trustwave – SpiderLabs
@dan_crowley
dcrowley@trustwave.com