Open Web Application Security Project

There’s a hole in my bucket, dear Liza — Examining side channel leaks in web apps.

Benjamin Holland

ben-holland.com

Quick Note Before We Get Started…

- With regard to *some* information in this talk:
  - This material is based on research sponsored by DARPA under agreement number FA8750-15-2-0080. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon.
  - The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA or the U.S. Government.

OWASP
Open Web Application
Security Project

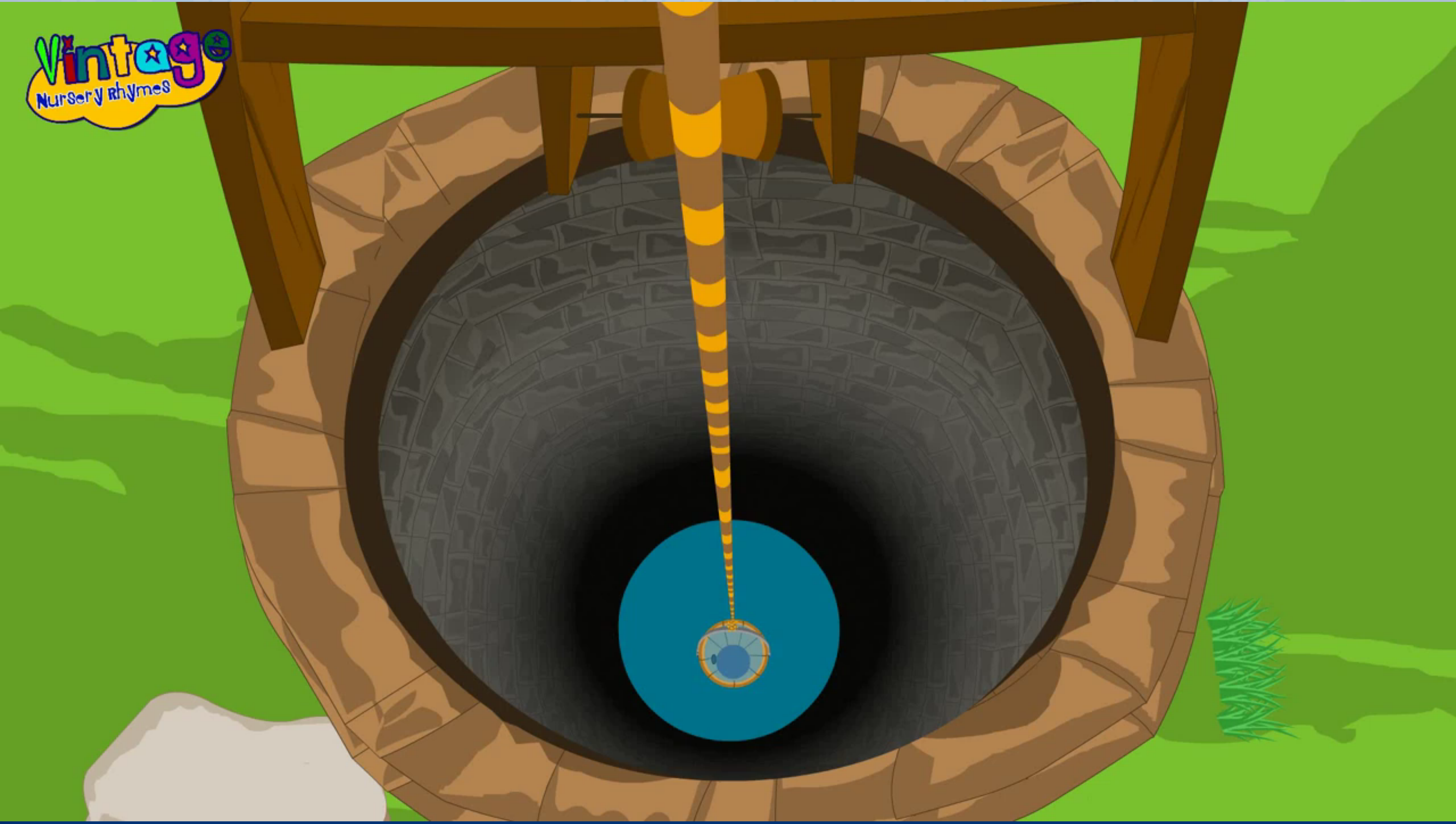
About Me

- B.S. in Computer Engineering (2005 - 2010)
  - Wabtec Railway Electronics, Ames Lab, Rockwell Collins
- B.S. in Computer Science (2010 - 2011)
- M.S. in Computer Engineering and Information Assurance (2010 - 2012)
  - MITRE
- ISU Research Scientist (2012 - 2015)
  - DARPA Automated Program Analysis for Cybersecurity
  - DARPA Space/Time Analysis for Cybersecurity
- PHD in Computer Engineering (2015-????)

Talk Overview

- Establish a common understanding of side channel vulnerabilities
- Provide some example side channel vulnerabilities
  - Physical → Hardware → Software
- Causes of side channels
- Discuss challenges in preventing/detecting software side channels

Setting the Stage



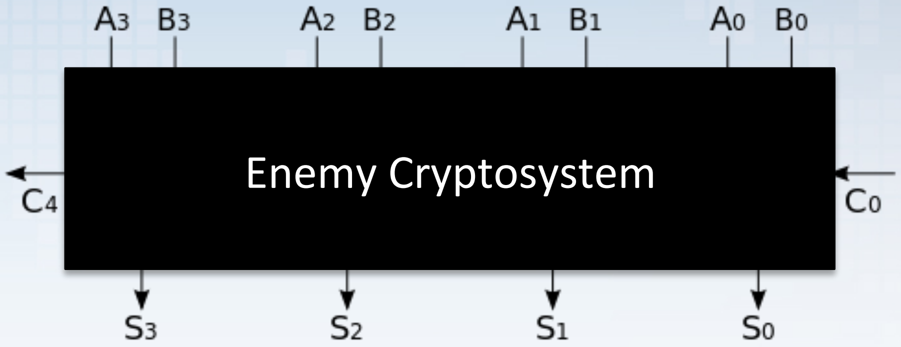
What's a Side Channel?

- How big is Henry's bucket?
  - What information do we have?

Information Leakage (No Pun Intended)

What’s a Side Channel?

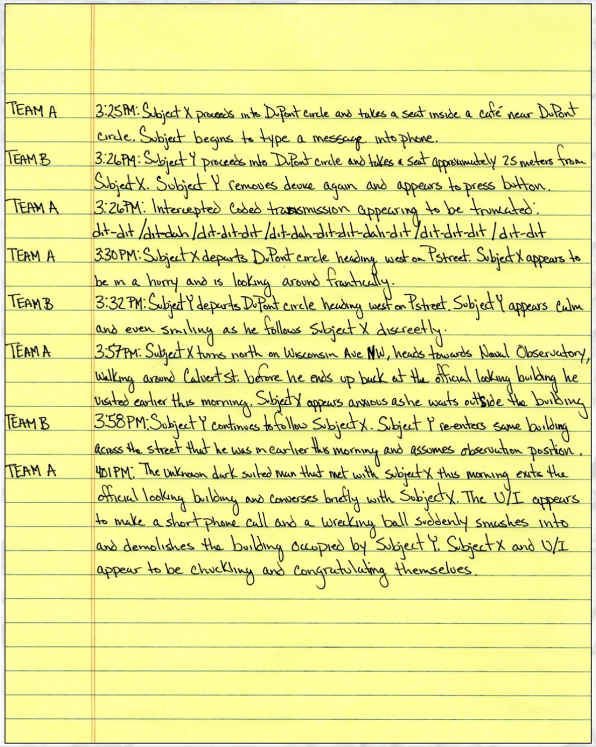
- Historically side channels were used to describe attacks to physical crypto hardware systems
  - Power analysis
  - Timing information
  - Acoustics
  - Faults
  - Electromagnetic radiation
    - Light, heat, IR, etc.
- Some operations require more time, power, etc. to complete than others



A_0	B_0	A_1	B_1	A_2	B_2	A_3	B_3	C_0
0	1	1	1	1	0	0	0	0
-	-	-	-	S_0	S_1	S_2	S_3	C_4
-	-	-	-	1	0	0	1	0

DARPA's Paper Shredder Challenge

- $50,000 prize to unscramble 5 shredded documents
- Puzzles were completely solved on December 2011 by team "All Your Shreds Are Belong To U.S."



TEAM A 3:25PM: Subject X proceeds into DuPont circle and takes a seat inside a café near DuPont circle. Subject begins to type a message into phone.

TEAM B 3:26PM: Subject Y proceeds into DuPont circle and takes a seat approximately 25 meters from Subject X. Subject Y removes device again and appears to press button.

TEAM A 3:26PM: Intercepted Coded transmission appearing to be truncated:
dit-dit /dit-dah /dit-dit-dit /dit-dah-dit-dit-dah-dit /dit-dit-dit / dit-dit

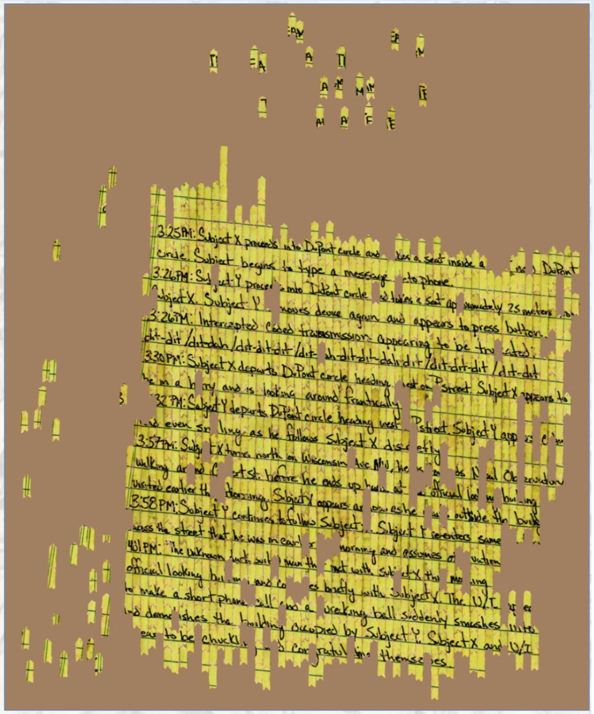
TEAM A 3:30 PM: Subject X departs DuPont circle heading west on P street. Subject X appears to be in a hurry and is looking around frantically.

TEAM B 3:32 PM: Subject Y departs DuPont circle heading west on P street. Subject Y appears calm and even smiling as he follows Subject X discreetly.

TEAM A 3:57PM: Subject X turns north on Wisconsin Ave NW, heads towards Naval Observatory, walking around Calvert St. before he ends up back at the official looking building he visited earlier this morning. Subject X appears anxious as he waits outside the building

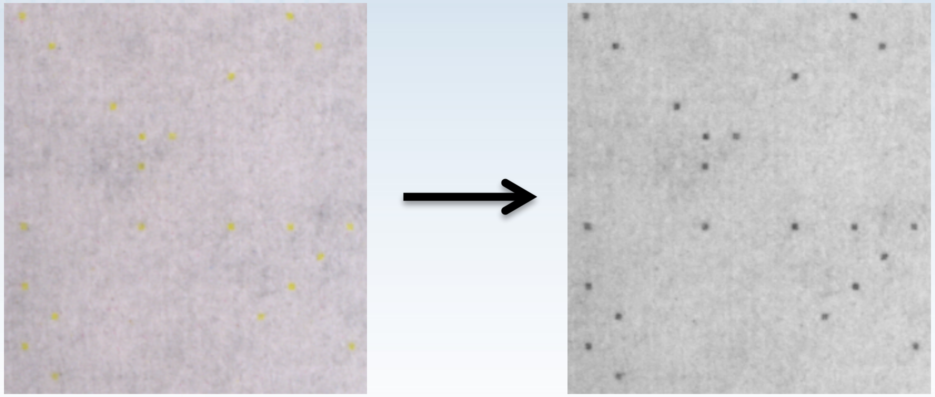
TEAM B 3:58 PM: Subject Y continues to follow Subject X. Subject Y re-enters same building across the street that he was in earlier this morning and assumes observation position.

TEAM A 401 PM: The unknown dark suited man that met with subject X this morning exits the official looking building and converses briefly with Subject X. The U/I appears to make a short phone call and a wrecking ball suddenly smashes into and demolishes the building occupied by Subject Y. Subject X and U/I appear to be chuckling and congratulating themselves.



Paper Shredder Side Channel

- A little of life's irony…
  - ~9000 teams competed, 1 team solved all 5 puzzles
  - Solution used hidden printer dots added by printer manufacturers and U.S. Secret Service



  - Vision recognition software detected dots printed on paper and used dots as a reference guide to identify document fragments
  - Pro-tip: Burn your documents you really want gone…

Don’t Touch Das Blinkenlights

ACHTUNG!

Alles turisten und non-teknischen looken peepers! Das computermaschine ist nicht füer gefingerpoken und mittengraben! Ist easy schnappen der springenwerk, blowenfusen und poppencorken mit spitzensparken.

Ist nicht füer gewerken bei dummkopfen. Das rubbernecken sightseeren keepen das cotton-picken hans in das pockets muss; Zo relaxen und watschen der blinkenlichten.

Das Blinkenlight

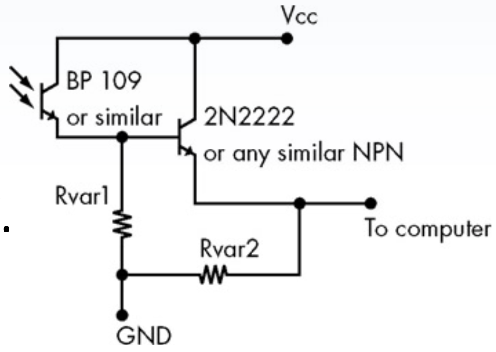
Historically, the blinking lights indicated important things like the state of the system, but as computers became faster and more reliable the lights were either removed or left as diagnostic indicators (example: networking hardware).

OWASP
Open Web Application
Security Project

Blinkenlights Problem

- LEDs on/off time is very fast (almost instant)
  - LEDs are usually used to control fiber optics
- LEDs were wired directly into the serial data line
  - Each blink is a 1 on your network, LED off is a 0
  - Too fast for a human eye
  - Not too fast for a circuit…and a telephoto lens!

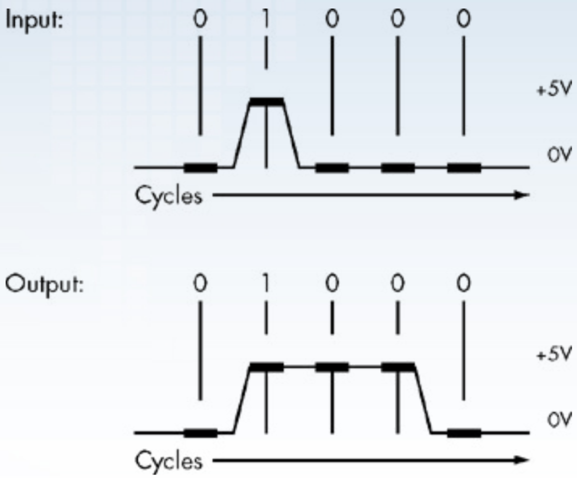
Paper: Joe Loughry and David A. Umphress. 2002. *Information leakage from optical emanations.* ACM Trans. Inf. Syst. Secur. 5, 3 (August 2002), 262-289.

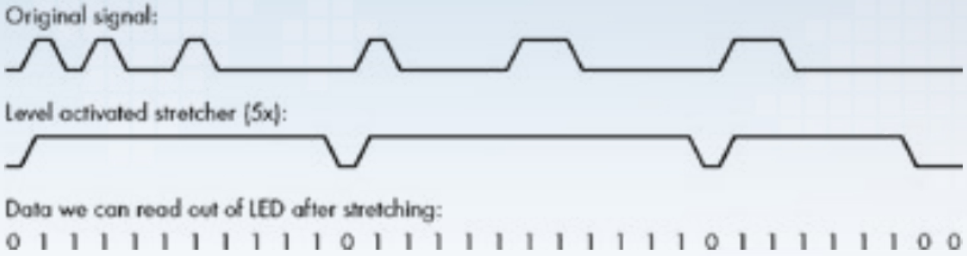


OWASP
Open Web Application Security Project

Blinkenlights Solutions

- Duct tape over the LEDs works, but we still want our blinkenlights!
  – Pulse Stretching ☹





Still possible to recover 99.999988% of bits.
Error correction codes can help us guess the rest.

  – Better approach → Low frequency sampling with a latch till the next sample

Origins of Side Channels

- Short story: optimizations
  - Reducing cost: power, heat, etc.
  - Increasing speed/efficiency
- Consider synchronous vs. asynchronous digital logic circuits
  - Synchronous circuits operate on a fixed clock, all operations take the same time, so the best case and worst case times are the same
    - Every case is the worst case
  - Asynchronous circuits operate without a clock independent of other modules, so there are distinct best, worst, and average cases.
    - Average case costs less than the worst case

Side Channels in Software

- Leakage primarily through
  - Timing information
  - Memory space usage
    - Content, order, size

Space/Time usage are related problems

- Optimizations everywhere…
  - Software algorithms
    - Branching, short-circuiting logic, looping, etc.
  - Compiler optimizations
  - Cache hits
  - Process scheduling
  - Branch prediction…and so on…

```
if(secret){
   doShortA();
} else {
   doLongAction();
}
```

Demasking Google Users

1. Select Google users to target
2. Create a Google drive document and invite targets (uncheck option to send notification)
3. Using HTML/JavaScript create a spear-phishing site that identifies and customizes itself for the target
   - <img src=https://docs.google.com/document/....../edit /> takes longer to call onerror if visitor is a target
   - Google has declined to fix this issue

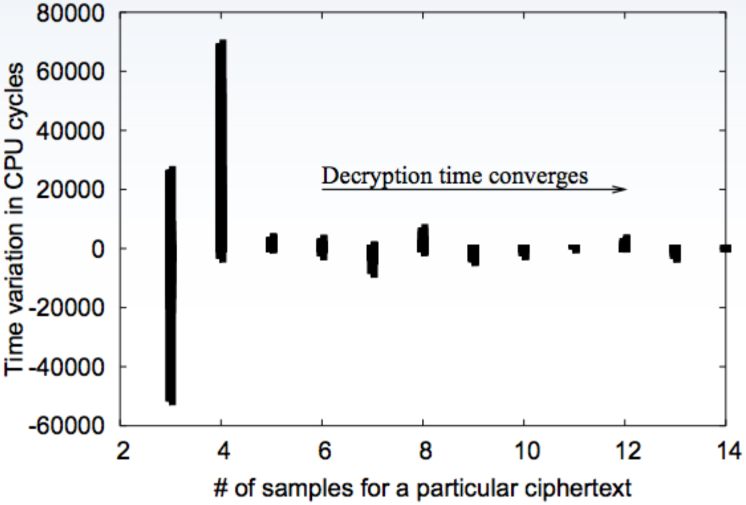
OpenSSL Timing Attack

- Timing attack against OpenSSL server to recover SSL private RSA key
  - RSA decryption: $m = c^*d \bmod N$, where $N = pq$
  - If you know the factorization of N, then $d = e^{-1} \bmod (p - 1)(q - 1)$
- Issue: Algorithm processing time was dependent on ciphertext and private key
  - Extra reductions in a Montgomery reduction (fast mod operation) when ciphertext (c) approaches a multiple of q ($c < q$ should be slower than decryption of $c > q$)
  - OpenSSL uses two different multiplication routines: when $c < q$ fast Karatsuba multiplication is used, otherwise the slower normal multiplication is used since $c > q$ is likely smaller when computing $c \bmod q$
- Performed over a network (dealing with network delays)

OpenSSL Timing Attack (Continued)

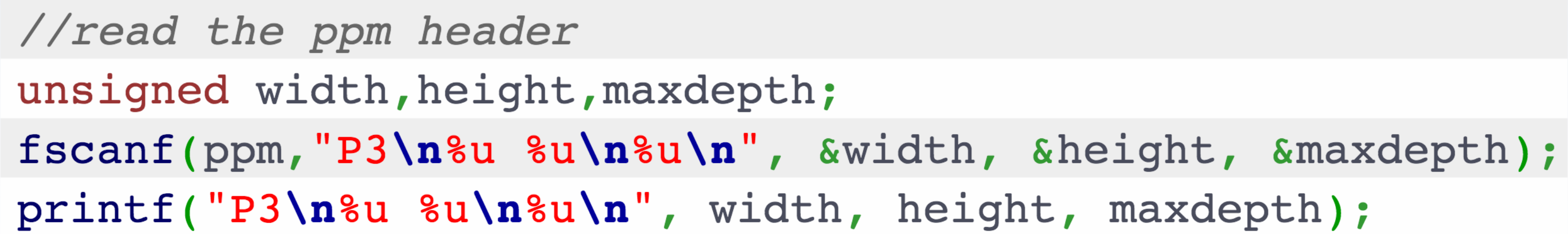
- Use of repeated requests to recover modulus *N* of the public key
  - Binary search of most significant bits
  - After half the bits are recovered factorization is completed with Coppersmith's algorithm

Note that not all of the secret was leaked! Just enough of the secret was leaked to make brute force search feasible.



Underhanded C Contest

- Good example of a malicious side channel
  - Source has to be capable of passing a peer review
  - Execution has to appear to perform the task correctly
  - Challenge: Censor regions on a JPEG image, but somehow leak the redacted information
- Snippet of winning solution:



```
//read the ppm header
unsigned width,height,maxdepth;
fscanf(ppm,"P3\n%u %u\n%u\n", &width, &height, &maxdepth);
printf("P3\n%u %u\n%u\n", width, height, maxdepth);
```

  - Writes the magnitude of the red, green, and blue component for each pixel in order

Underhanded C Contest (Continued)

- When we censor with a black rectangle we write 0's for the RGB pixel (a black pixel)
- PPM file format is flexible and implementation leaks how many digits each value originally when it processes the file character by character

	234	2	0	83	255	255	2	43	255
Implementation:	000	0	0	00	000	000	0	00	000
Should write:	0	0	0	0	0	0	0	0	0

- Perfect reconstruction for black/white images, otherwise partial reconstruction of blacked out region

Mitigations

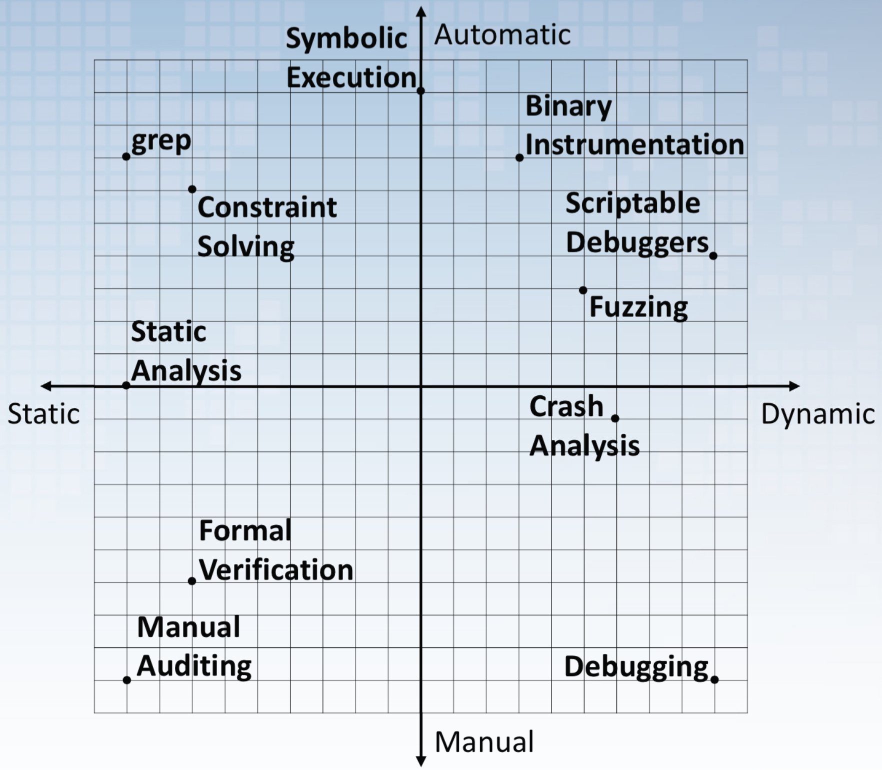
- Unfortunately it is difficult to make software run in fixed time.
  - All *sensitive* program paths need to take the same time
  - All *sensitive* program outputs need to be the same
    - Outputs include memory, file, network, etc.

- Idea: Injecting random time delays
  - Random delays just increase the number of samples needed to perform the attack
  - Statistics eventually win and attacker can begin to discard the noise

OWASP
Open Web Application
Security Project

Ongoing Research

- DARPA STAC Program
- *Human-in-the-loop* based program analysis approach to detect Space/Time side channel attacks
- Tool for amplifying human's program comprehension
- Challenges:
  - Analyze Java bytecode binaries
  - Identification of secrets
  - Loop complexity
  - Exponential paths
  - Large frameworks/libraries
  - Mixed code environments (C/C++)

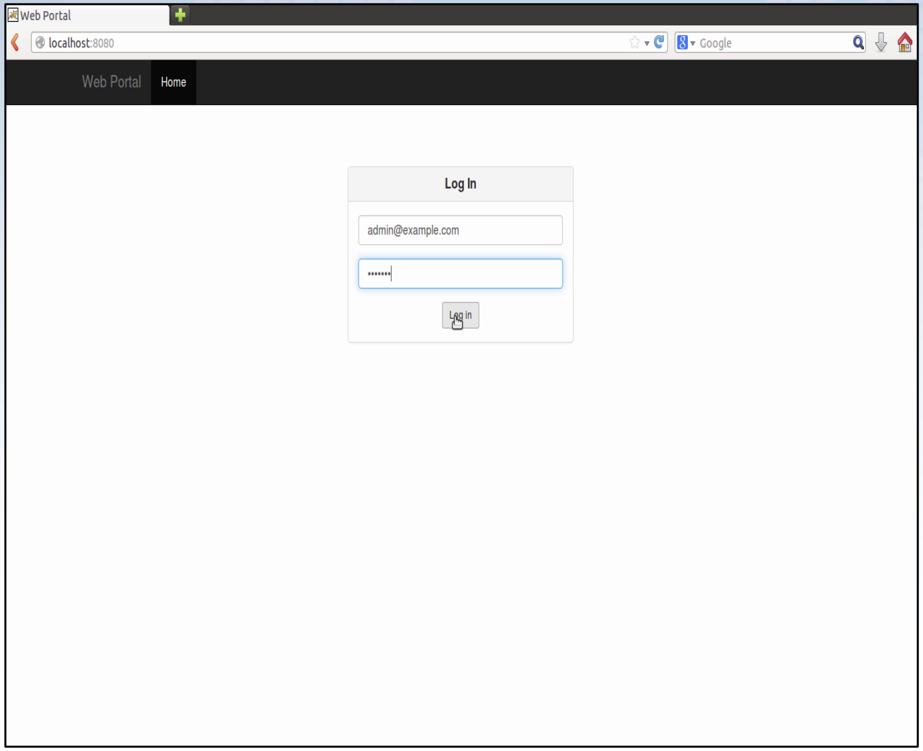


Source: Contemporary Automatic Program Analysis, Julian Cohen, Blackhat 2014

Demonstration Webapp

Java Server Page (JSP) web app with MySQL database backend

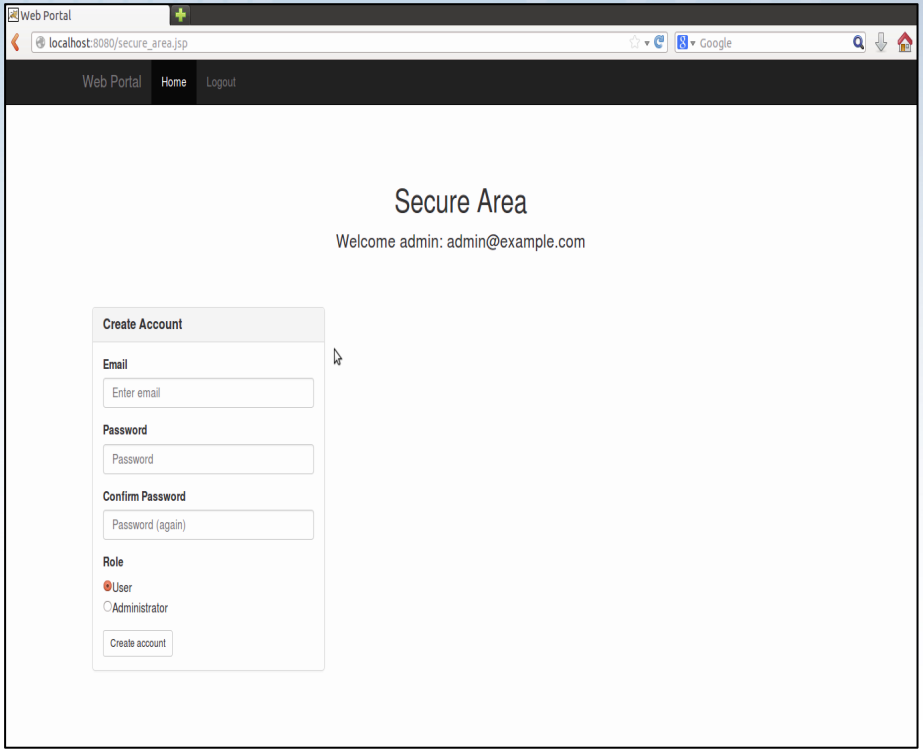
Source: https://github.com/benjholla/LoginSideChannels



Log In

admin@example.com

•••••••



Secure Area

Welcome admin: admin@example.com

Create Account

Email

Enter email

Password

Password

Confirm Password

Password (again)

Role

● User

○ Administrator

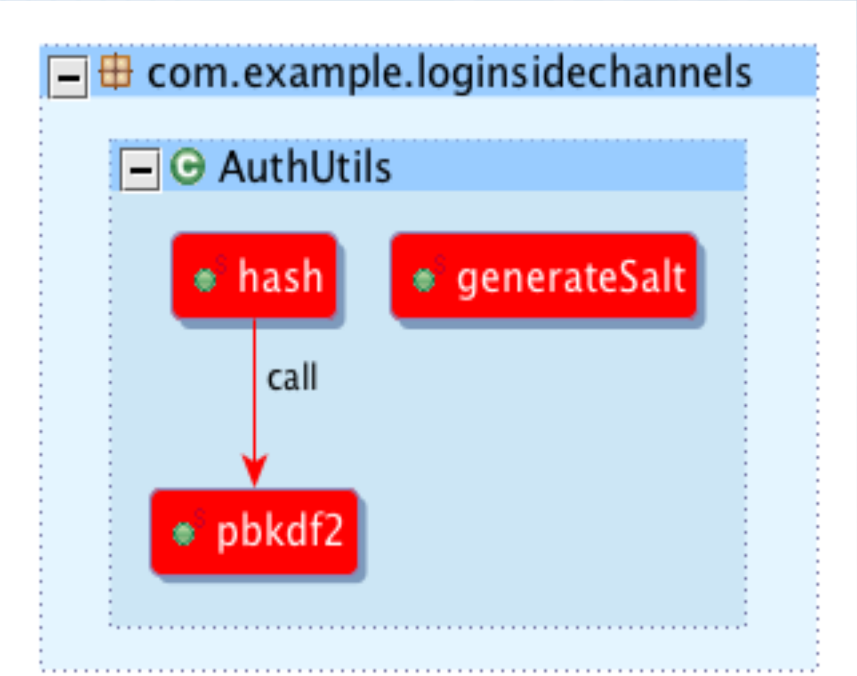
OWASP
Open Web Application
Security Project

LoginSideChannels Vulnerability

- The existence of users can be inferred through timing differentials.
- More time is required to validate a password of a valid user than an invalid user.
- Attacker does not need to know any valid passwords and only has to guess at valid users.

Loop Analysis

- **Approximation:** Loops are expensive and nested loops are more expensive than non-nested loops
- **Loop Call Graph:** Recovers loops, induces call edges, highlights calls of loops called within loops.
- **Note:** Hashes are computed in a feedback loop of N rounds for improved resistance to brute force attacks.



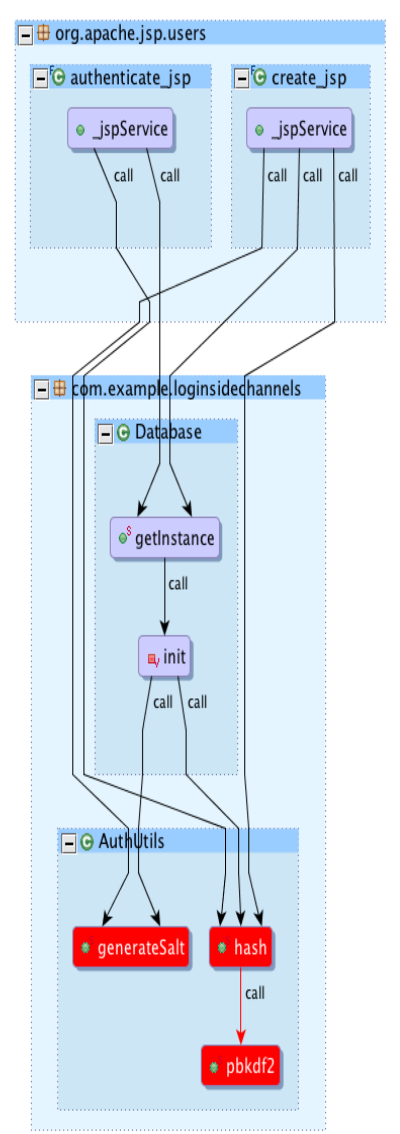
OWASP
Open Web Application Security Project

Loop Context

Question: Where are these loops used and why?

Analysis: Inspect call graph to get some context

Answer: Primarily used by two services: authenticate user and create user.



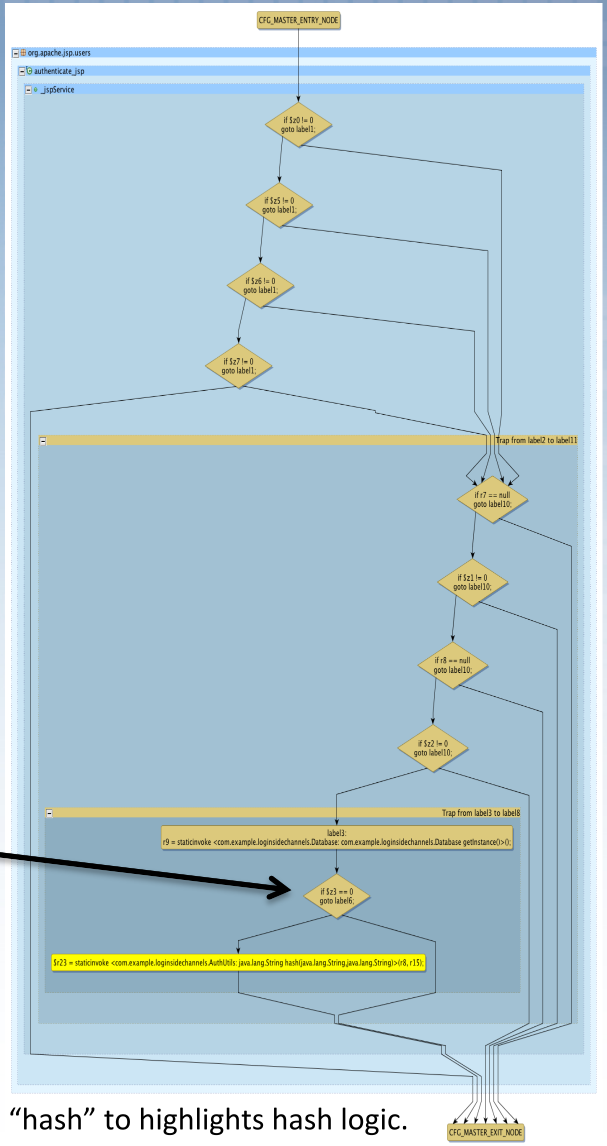
OWASP Open Web Application Security Project

Find guard conditions

Question: Is the expensive logic used conditionally?

Analysis: Compute an Event Flow Graph (EFG, a compact graph containing only relevant conditions). Inspect “authenticate_jsp” method in a EFG.

Answer: EFG reveals a conditional guard on the hash. Analyst clicks to view code. Condition depends on result of SQL query.


CFG_MASTER_ENTRY_NODE
org.apache.jsp.users
authenticate_jsp
_jspService
if $z0 != 0 goto label1;
if $z5 != 0 goto label1;
if $z6 != 0 goto label1;
if $z7 != 0 goto label1;
Trap from label2 to label11
if r7 == null goto label10;
if $z1 != 0 goto label10;
if r8 == null goto label10;
if $z2 != 0 goto label10;
Trap from label3 to label8
label3:
r9 = staticinvoke <com.example.loginsidechannels.Database: com.example.loginsidechannels.Database getInstance()>();
if $z3 == 0 goto label6;
$r23 = staticinvoke <com.example.loginsidechannels.AuthUtils: java.lang.String hash(java.lang.String,java.lang.String)>(r8, r15);
CFG_MASTER_EXIT_NODE

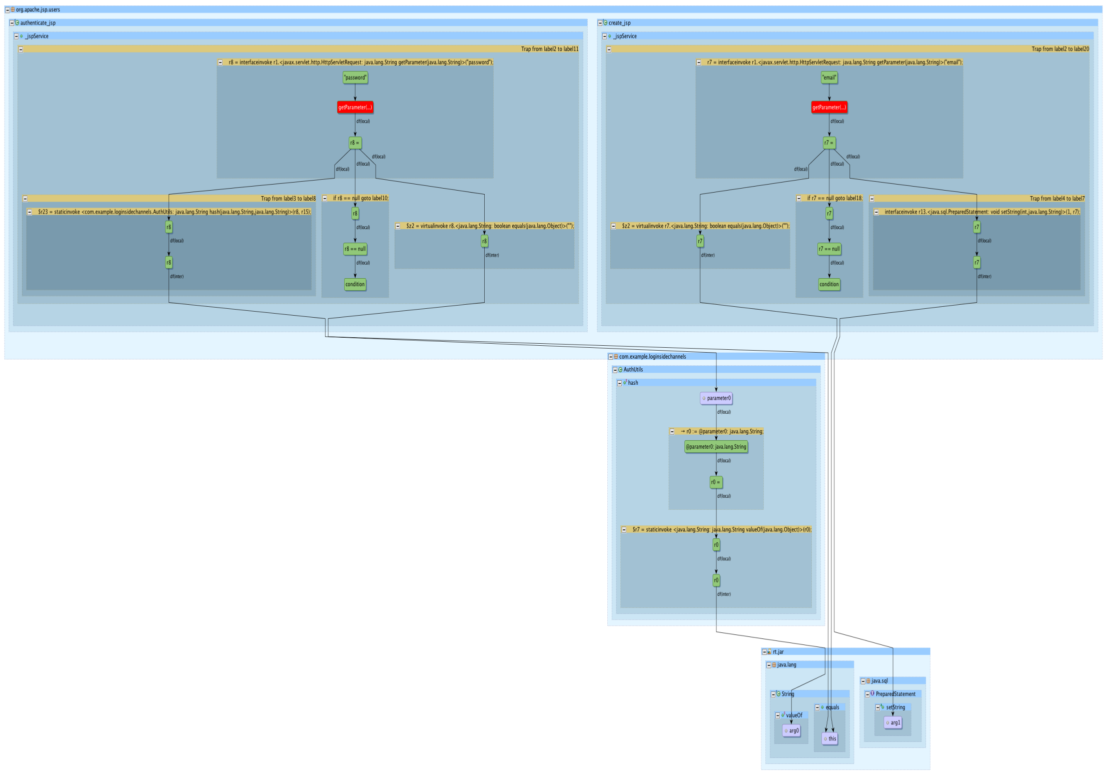
Control-F: Find “hash” to highlights hash logic.

Check secret confidentiality

Question: Can a secret be deduced by this potential timing difference?

Analysis: Follow data flow forward from secrets (email, password) to conditionals.

Observation: Password flows to hash; email flows into SQL.



```
r8 = interfaceinvoke r1.<javax.servlet.http.HttpServletRequest: java.lang.String getParameter(java.lang.String)>("password");
$r23 = staticinvoke <com.example.loginsidechannels.AuthUtils: java.lang.String hash(java.lang.String,java.lang.String)>(r8, r15);
if r8 == null goto label10;
$z2 = virtualinvoke r8.<java.lang.String: boolean equals(java.lang.Object)>("");
r7 = interfaceinvoke r1.<javax.servlet.http.HttpServletRequest: java.lang.String getParameter(java.lang.String)>("email");
$z2 = virtualinvoke r7.<java.lang.String: boolean equals(java.lang.Object)>("");
if r7 == null goto label18;
interfaceinvoke r13.<java.sql.PreparedStatement: void setString(int,java.lang.String)>(1, r7);
→ r0 := @parameter0: java.lang.String;
$r7 = staticinvoke <java.lang.String: java.lang.String valueOf(java.lang.Object)>(r0);
```

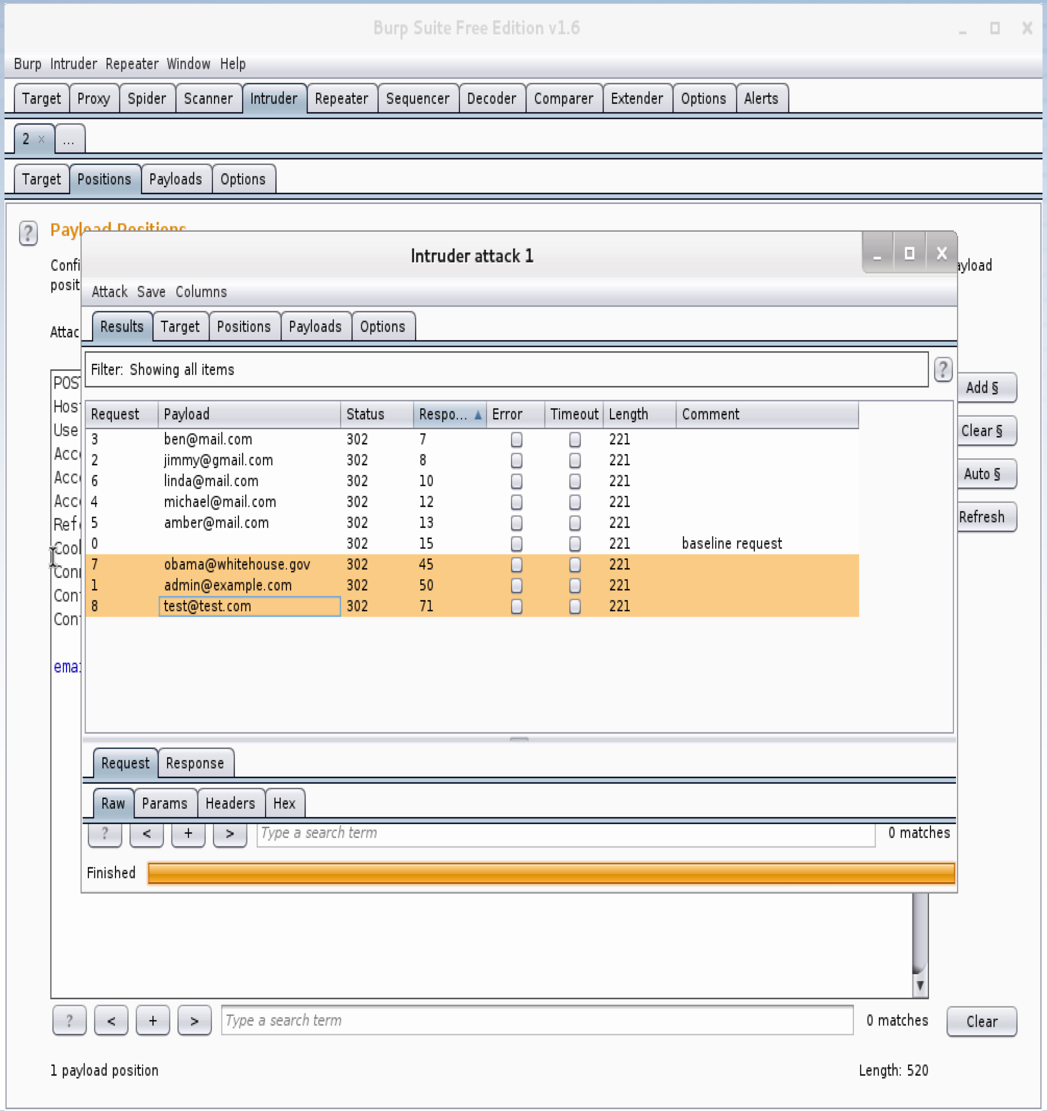
Check secret confidentiality

```
r9 = staticinvoke <Database: Database getInstance()>();
r10 = virtualinvoke r9.<Database: Connection getConnection()>();
r11 = interfaceinvoke r10.<Connection: PreparedStatement prepareStatement(String)>(
        "SELECT * FROM webdb.users where Email=? LIMIT 1");
interfaceinvoke r11.<PreparedStatement: void setString(int,String)>(1, r7);
r12 = interfaceinvoke r11.<PreparedStatement: ResultSet executeQuery()>();
$z3 = interfaceinvoke r12.<ResultSet: boolean next()>();
if $z3 == 0 goto label06;
...
$r23 = staticinvoke <AuthUtils: String hash(String,String)>(r8, r15);
```

Observation: The SQL query controls the condition of interest.

Answer: Relatively expensive logic (hash) is invoked only if email exists in the database.

Attack Demonstration

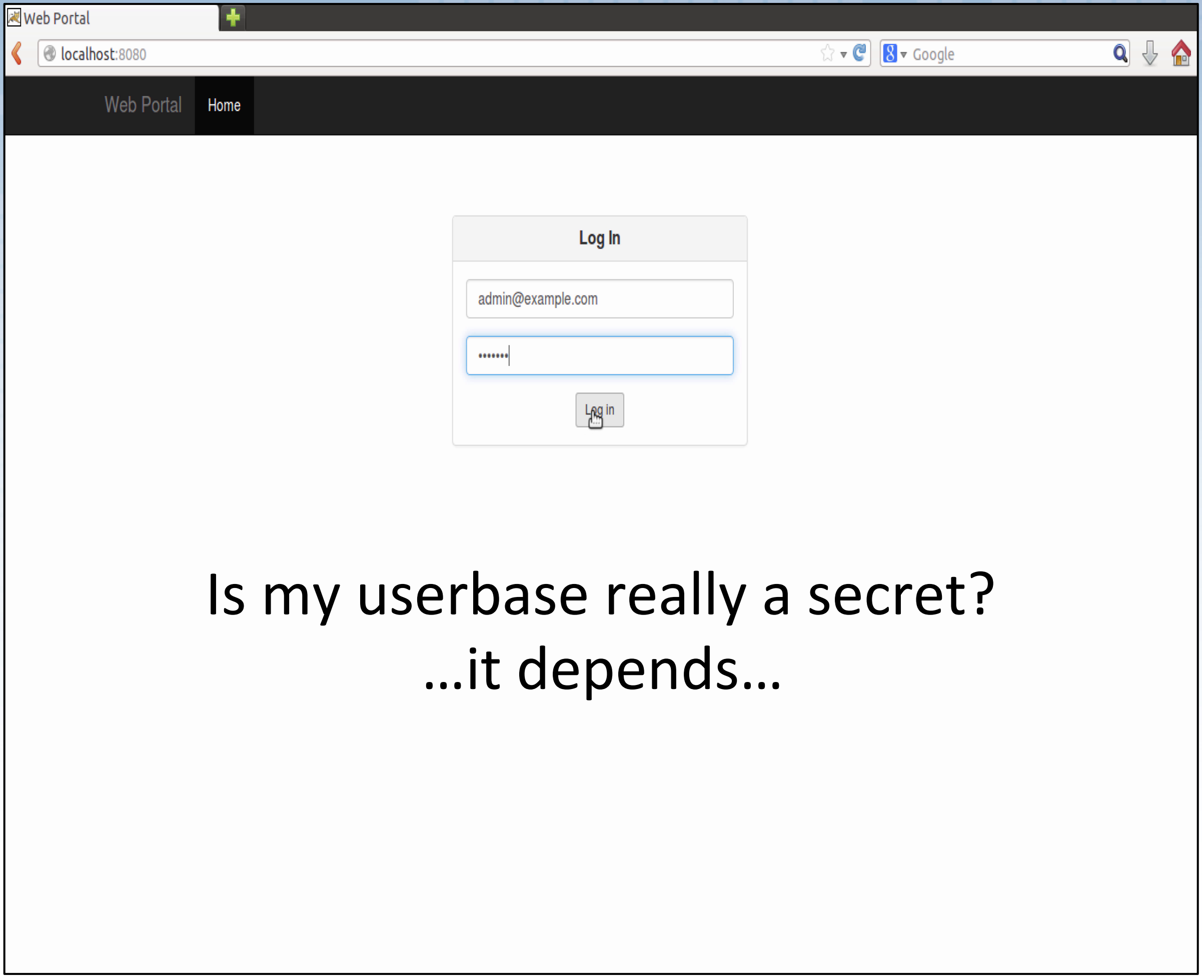


Filter: Showing all items

Request	Payload	Status	Respo... ▲	Error	Timeout	Length	Comment
3	ben@mail.com	302	7	☐	☐	221	
2	jimmy@gmail.com	302	8	☐	☐	221	
6	linda@mail.com	302	10	☐	☐	221	
4	michael@mail.com	302	12	☐	☐	221	
5	amber@mail.com	302	13	☐	☐	221	
0		302	15	☐	☐	221	baseline request
7	obama@whitehouse.gov	302	45	☐	☐	221	
1	admin@example.com	302	50	☐	☐	221	
8	test@test.com	302	71	☐	☐	221	

OWASP
Open Web Application
Security Project

Side Channel Impact



Log In

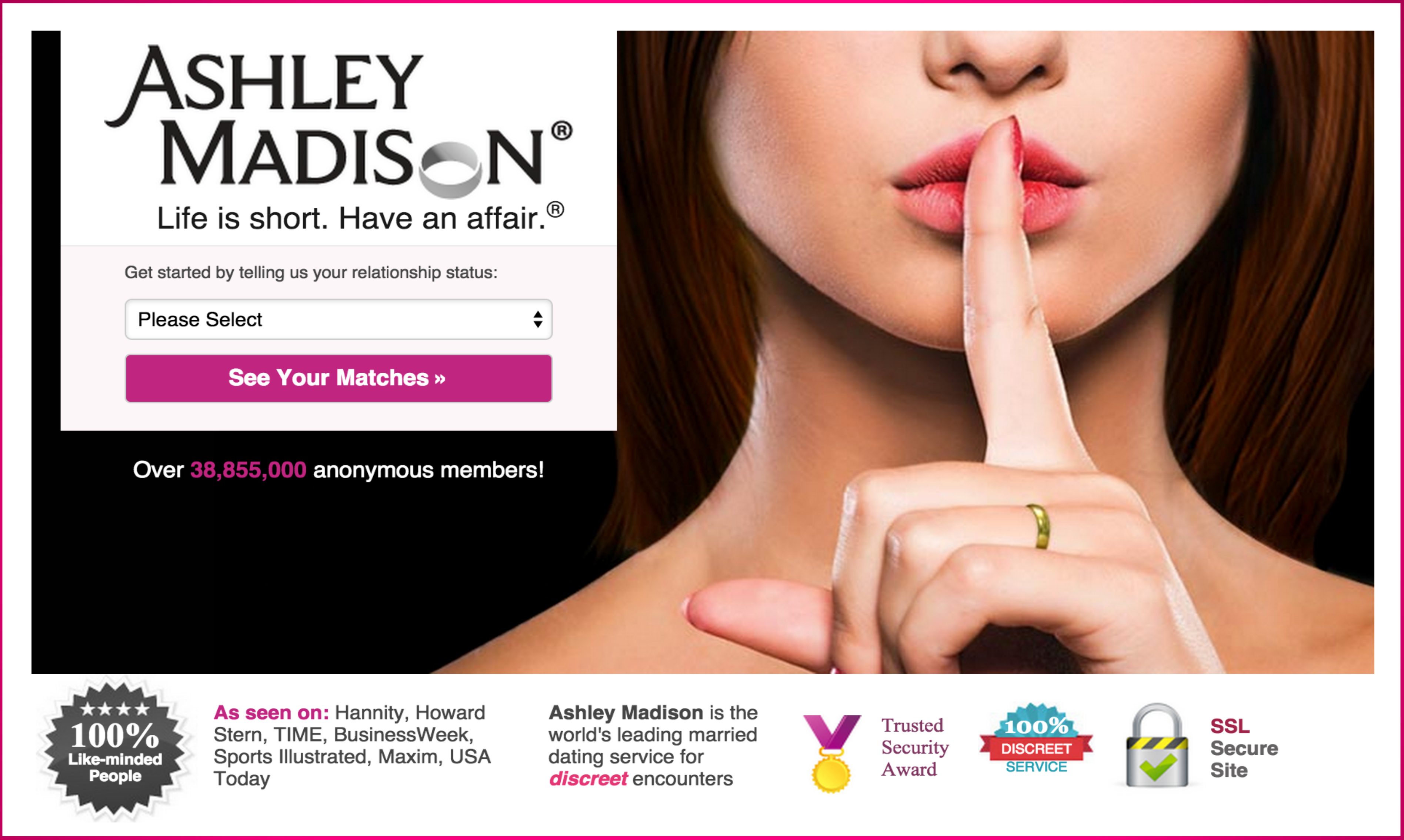
admin@example.com

•••••••

Is my userbase really a secret?

…it depends…

Side Channel Impact



ASHLEY MADISON®

Life is short. Have an affair.®

Get started by telling us your relationship status:

Please Select

Over 38,855,000 anonymous members!

As seen on: Hannity, Howard Stern, TIME, BusinessWeek, Sports Illustrated, Maxim, USA Today

Ashley Madison is the world's leading married dating service for ***discreet*** encounters

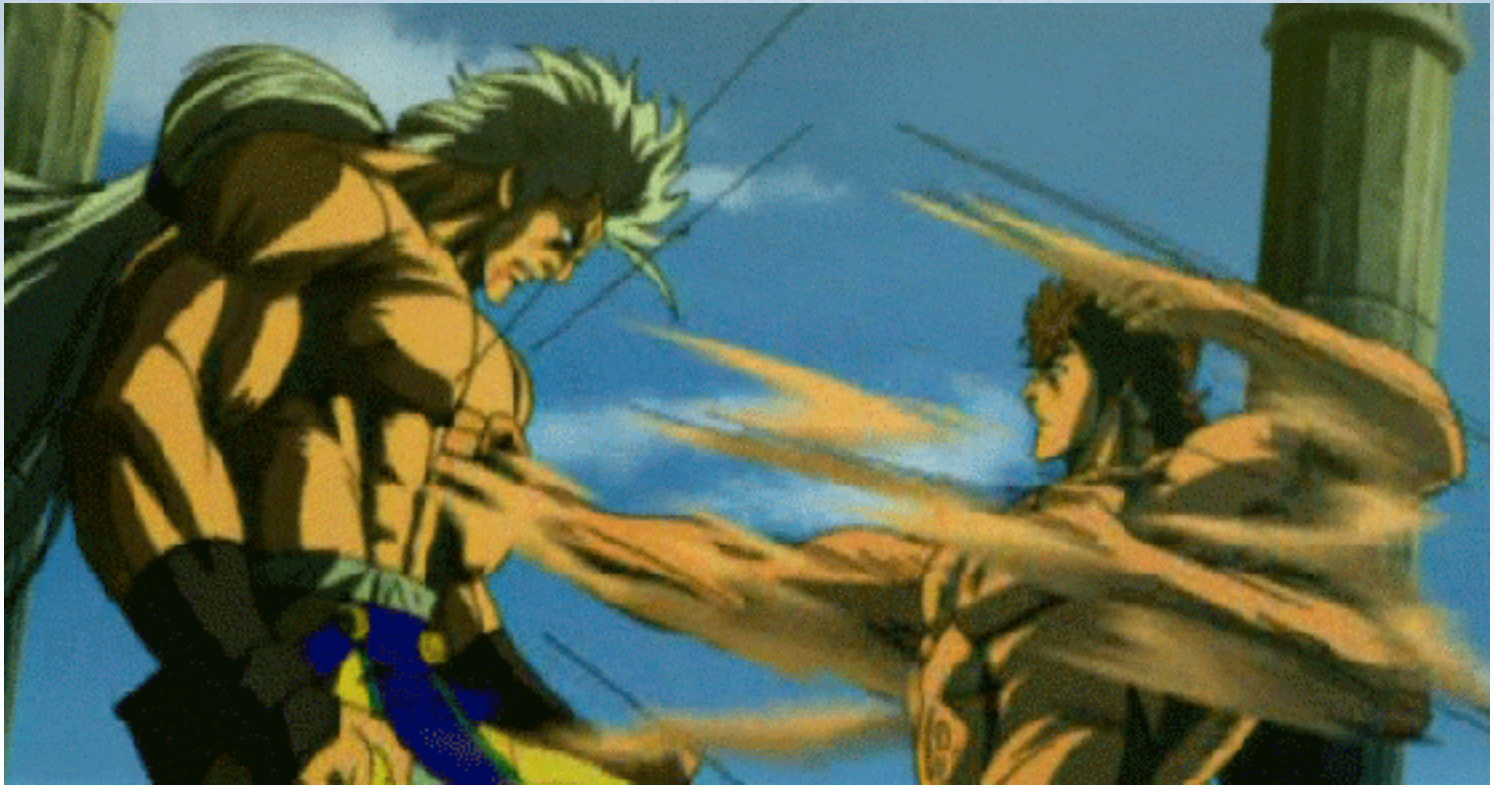
Future Prediction

Currently side channel exploits are like this…

You are vulnerable, you just don’t know it yet.

Future Prediction

In the future side channel exploits will be like this…



Some things to check…

- Timing/response of REST operations
- Ordering/content of
  - HTTP Headers, HTTP Parameters, Cookies
- Error messages
- …
- **Advice:** Start by considering your secrets and an attacker’s operational budget

In Closing

- If there is a hole in your bucket, dear Henry…

Then mend it, dear Henry.

THEN MEND IT, DEAR HENRY, DEAR HENRY, DEAR HENRY

References

[1] Children's Rhymes Video – https://www.youtube.com/watch?v=xzm9urjQbWU

[2] Ripple Carry Adder – https://en.wikipedia.org/wiki/Adder_(electronics)

[3] DARPA Paper Shredder Challenge – http://archive.darpa.mil/shredderchallenge

[4] U.S. Secret Service Printer Program – http://seeingyellow.com

[5] Blinkenlights (Chapter 5) – Michal Zalewski. 2005. *Silence on the Wire: A Field Guide to Passive Reconnaissance and Indirect Attacks.* No Starch Press, San Francisco, CA, USA.

[6] Demasking Google Users with a timing attack. Andrew Cantino. 2014.

[7] OpenSSL Timing Attack – Brumley, David, and Dan Boneh. Remote timing attacks are practical. Computer Networks 48.5 (2005): 701-716.

[8] Underhanded C Contest – http://notanumber.net/...the-leaky-redaction

Recommended Reading

[1] Eliminating Timing Side-Channels. A Tutorial. Peter Schawabe. ShmooCon 2015.

[2] Side Channel Vulnerabilities on the Web - Detection and Prevention. Sebastian Schinzel. OWASP Germany Conference 2010.

[3] Remote timing attacks are practical. Brumley, David, and Dan Boneh. Computer Networks 48.5 (2005): 701-716.

[4] Side Channel Attacks. John Franco. University of Cincinnati Network Security course lecture.

[5] Silence on the Wire: A Field Guide to Passive Reconnaissance and Indirect Attacks. No Starch Press, San Francisco, CA, USA. Michal Zalewski. 2005.

[6] WebGoat Blind String SQL Injection Challenge. https://www.owasp.org/index.php/Category:OWASP_WebGoat_Project

Questions?

Thank you.

Slides: ben-holland.com