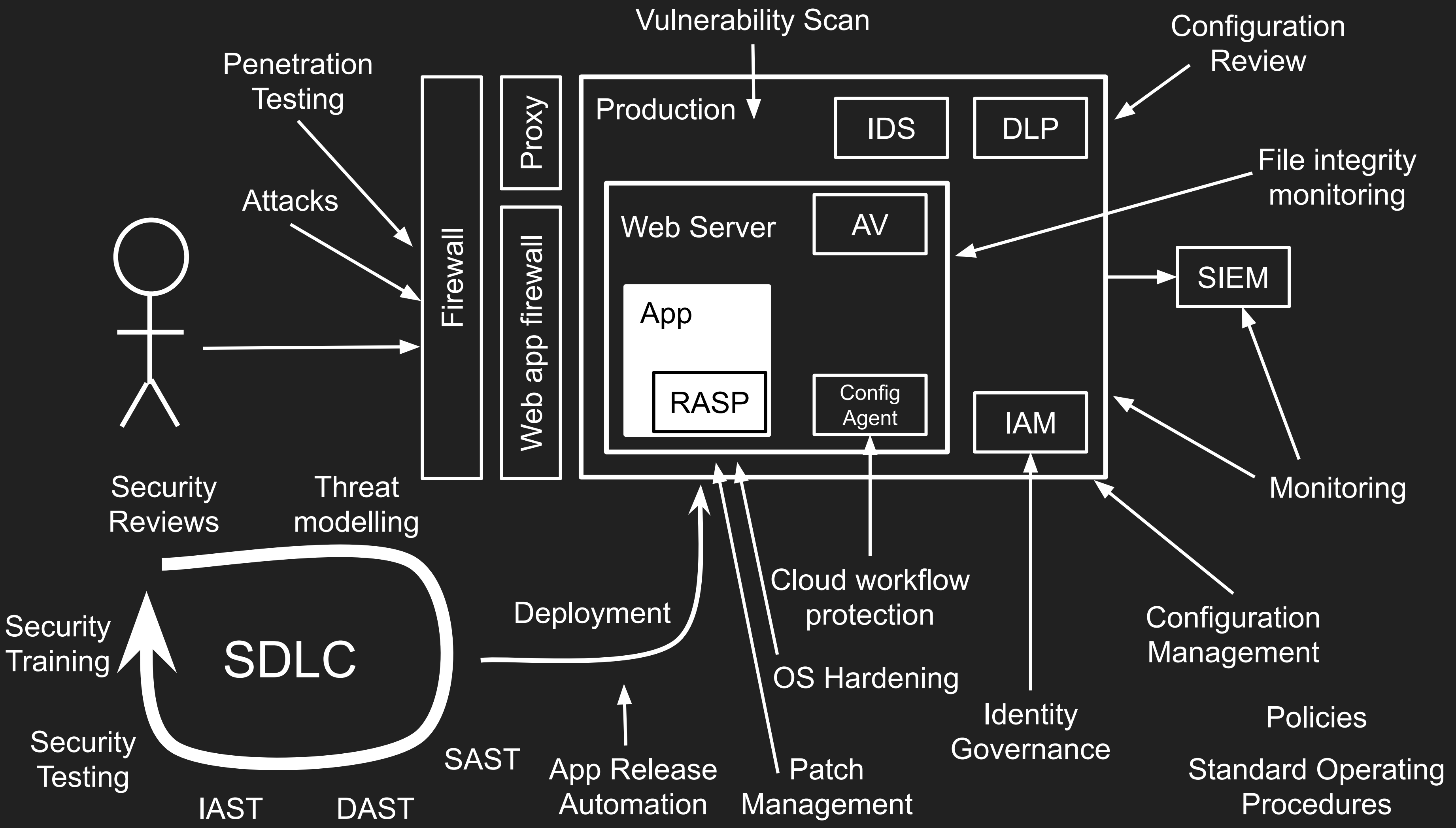
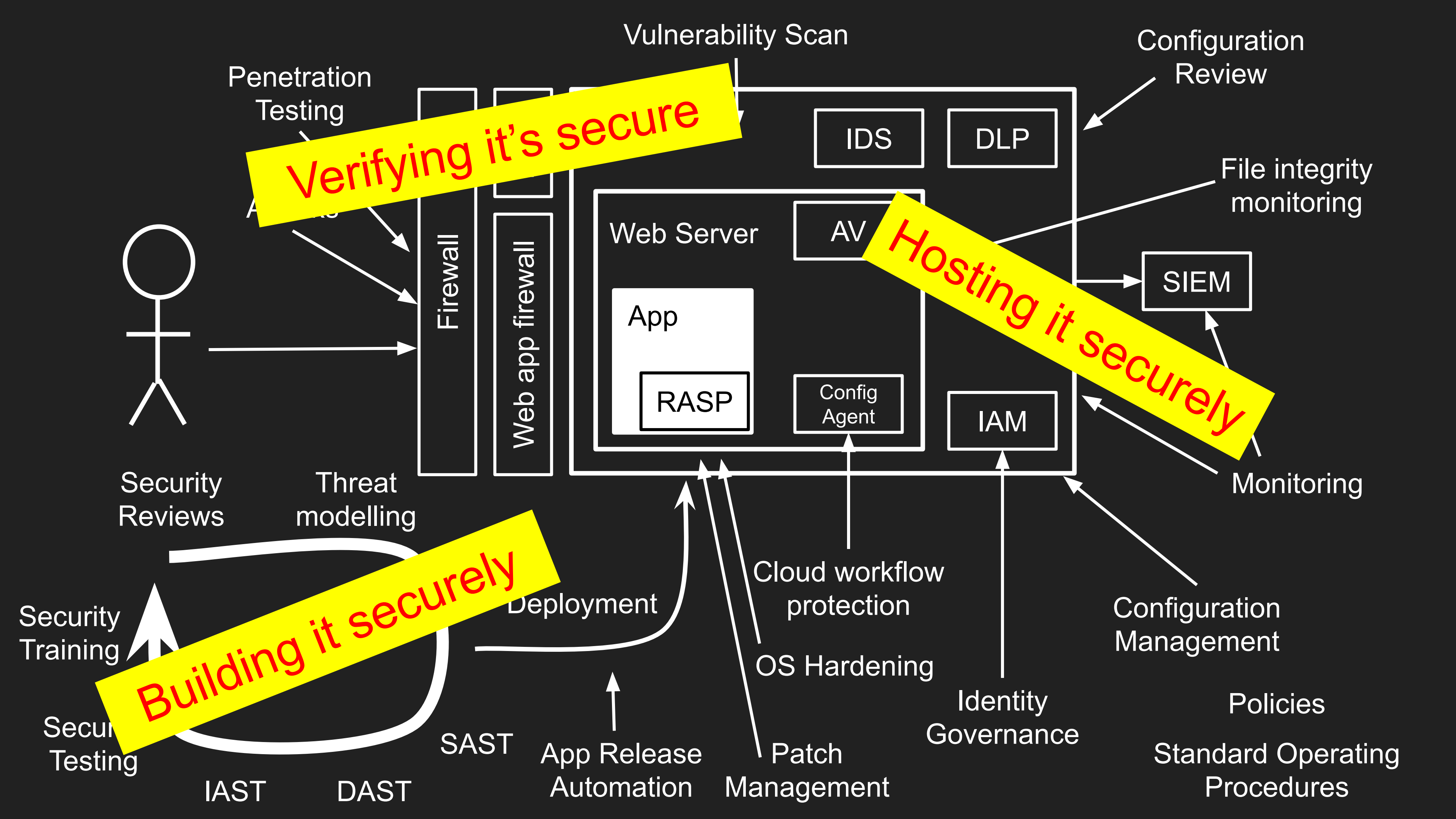
Don’t trust user input

Kirk Jackson, RedShield
security.ac.nz, 25 Aug 2019

Building a secure web app


Vulnerability Scan
Penetration Testing
Configuration Review
Production
IDS
DLP
Proxy
File integrity monitoring
Attacks
Web Server
AV
Firewall
Web app firewall
SIEM
App
RASP
Config Agent
IAM
Security Reviews
Threat modelling
Monitoring
Cloud workflow protection
Deployment
Configuration Management
Security Training
SDLC
OS Hardening
Identity Governance
Policies
Security Testing
SAST
App Release Automation
Patch Management
Standard Operating Procedures
IAST
DAST

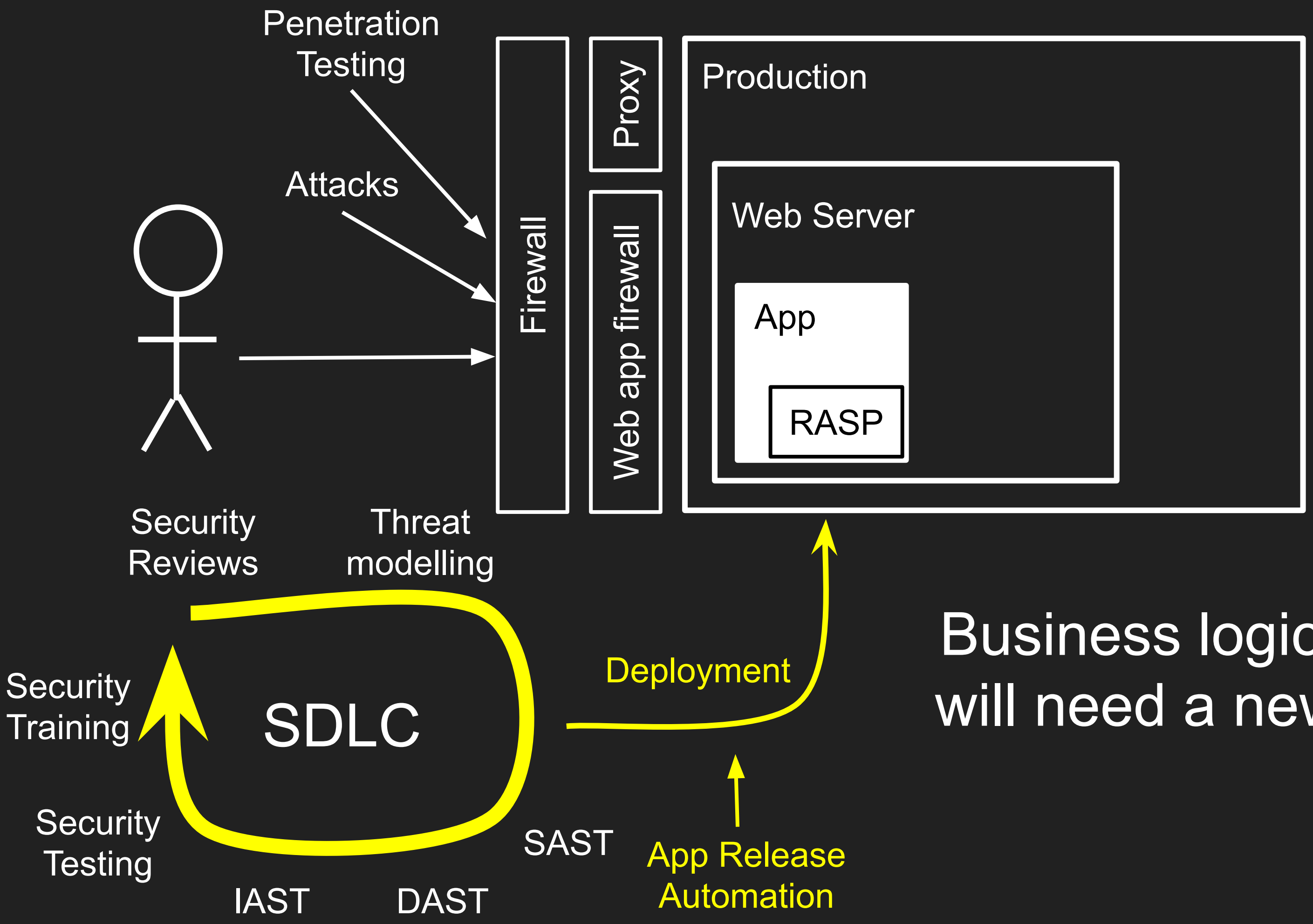

Vulnerability Scan
Penetration Testing
Configuration Review
Verifying it's secure
IDS
DLP
File integrity monitoring
Web Server
AV
Firewall
Web app firewall
App
RASP
Config Agent
Hosting it securely
SIEM
IAM
Monitoring
Security Reviews
Threat modelling
Cloud workflow protection
Configuration Management
Deployment
Building it securely
Security Training
Secu Testing
OS Hardening
Identity Governance
Policies
SAST
App Release Automation
Patch Management
Standard Operating Procedures
IAST
DAST

Approximate cost:
$4.2m

Building a secure web app

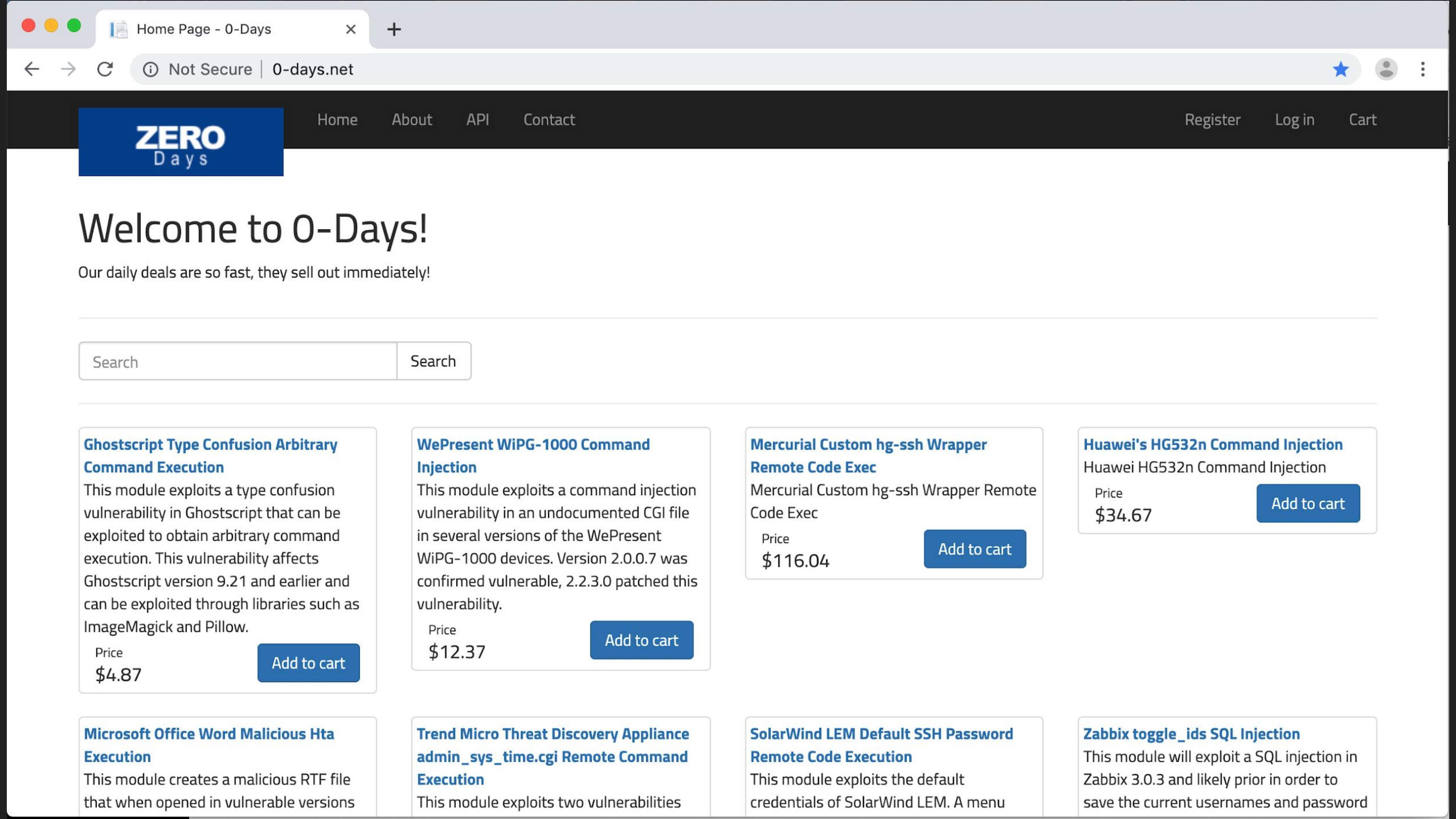
✓

But what if there are bugs?



Business logic weaknesses will need a new app release

XSS



Welcome to 0-Days!

Our daily deals are so fast, they sell out immediately!

Ghostscript Type Confusion Arbitrary Command Execution
This module exploits a type confusion vulnerability in Ghostscript that can be exploited to obtain arbitrary command execution. This vulnerability affects Ghostscript version 9.21 and earlier and can be exploited through libraries such as ImageMagick and Pillow.
Price
$4.87
Add to cart

WePresent WiPG-1000 Command Injection
This module exploits a command injection vulnerability in an undocumented CGI file in several versions of the WePresent WiPG-1000 devices. Version 2.0.0.7 was confirmed vulnerable, 2.2.3.0 patched this vulnerability.
Price
$12.37
Add to cart

Mercurial Custom hg-ssh Wrapper Remote Code Exec
Mercurial Custom hg-ssh Wrapper Remote Code Exec
Price
$116.04
Add to cart

Huawei's HG532n Command Injection
Huawei HG532n Command Injection
Price
$34.67
Add to cart

Microsoft Office Word Malicious Hta Execution
This module creates a malicious RTF file that when opened in vulnerable versions

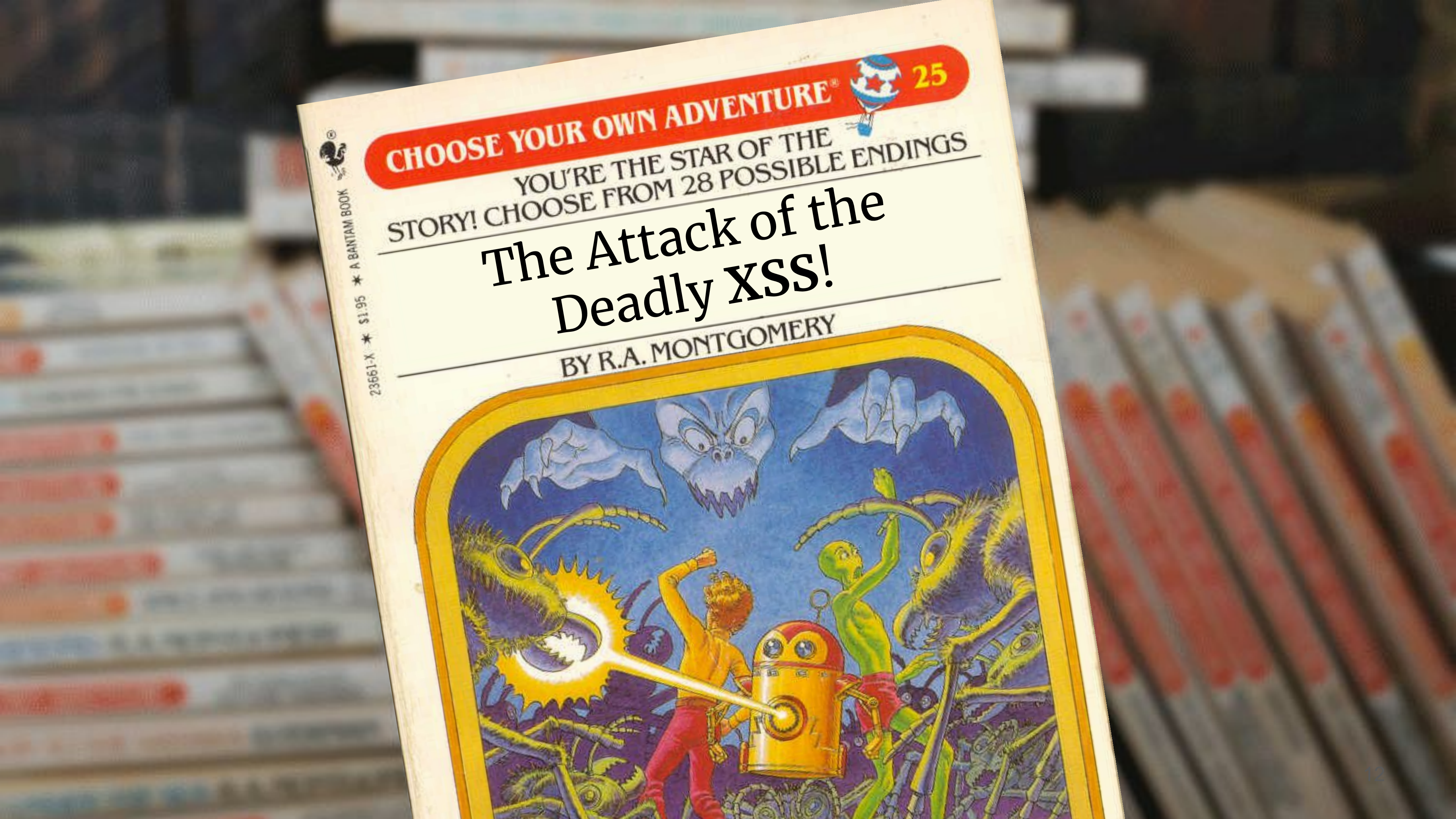
Trend Micro Threat Discovery Appliance admin_sys_time.cgi Remote Command Execution
This module exploits two vulnerabilities

SolarWind LEM Default SSH Password Remote Code Execution
This module exploits the default credentials of SolarWind LEM. A menu

Zabbix toggle_ids SQL Injection
This module will exploit a SQL injection in Zabbix 3.0.3 and likely prior in order to save the current usernames and password

```
GET /Product/Search?SearchTerm=ghost HTTP/1.1
Host: www.0-days.net
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_5)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/76.0.3809.100
Safari/537.36
Accept:
text/html,application/xhtml                            ge/webp,i
mage/apng,*/*;                                        ge;v=b3
Referer: htt
Accept-Encodi             , deflate
Accept-Language: en-GB,en-US;q=0.9,en;q=0.8
Cookie:
.AspNetCore.Antiforgery.9TtSrW0hzOs=CfDJ8JVmqLgybchGooENk8b3J2Arp7
JPwBPHmd6ZFeABp7WkL3Oad7vmVBUmgjuLe7B3p8KApo1sdYkvdxqdkwqN1XS3YjCV
eoOlLfwdrFSH8PltvmwuVnhUJNpl3pF3ys9YA8LISJVZAeSo69A2QYDedxc
```

Don’t trust anything!


CHOOSE YOUR OWN ADVENTURE®
25
YOU'RE THE STAR OF THE STORY! CHOOSE FROM 28 POSSIBLE ENDINGS
The Attack of the Deadly XSS!
BY R.A. MONTGOMERY
A BANTAM BOOK ★ $1.95 ★ 23561-X

Beware and warning!

This book is different from other books.

You and YOU ALONE are in charge of what happens in this story.

There are dangers, choices, adventures and consequences. YOU must use all of your numerous talents and much more of your enormous intelligence.

The wrong choice could end in disaster - even death.

But don’t despair. At anytime, YOU can go back and make another choice, alter the path of your story, and change its result.

Turn to page 14.

The beginning

You are an intrepid web developer named Justice, aiming to protect your application as best as you can.

Armed with your trusty list of websites, you begin your quest to rid the world of XSS!

- Javascript was invented in 1995
- Cross-site scripting was invented shortly after

Do you understand XSS? Go to Page 19.

Want to learn more? Go to Page 15.

XSS is everywhere

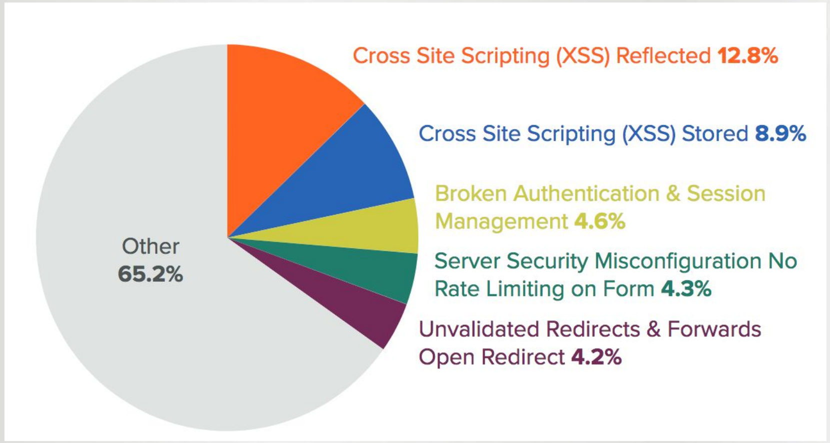
60% of bug payouts by Google

21.7% of bugs found on Bug Crowd are cross-site scripting

Almost all sites have XSS*

Injected javascript can do *anything that your users can do*

(Source: anecdotal)*



Bug Crowd's 2018 State of Bug Bounty report

Go to Page 16.

HTML = code + data

A single HTML page mixes both code and data

It’s all jumbled together

The browser doesn’t know who wrote the HTML:

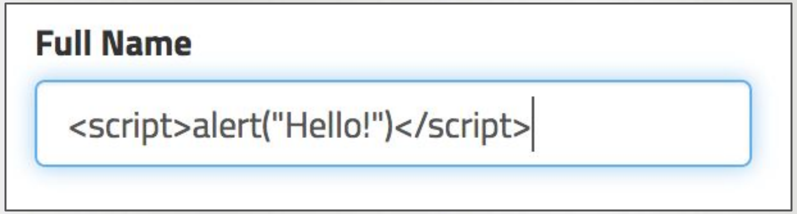
- The site creator?
- The end user?

```
<html>
<body>
<h1>Hello!</h1>
<script>
    var urchin_id = "61143";
</script>
</body>
</html>
```

Go to Page 17.

An attack

User enters their name:



```
<script>alert("Hello!")</script>
```

The page renders the name in an HTML context

```
<html>
<body>
<h1>
    Hello
    <script>
        alert(“Hello!”);
    </script>
</h1>
</body>
</html>
```

Go to Page 18.

An attack

The user's data can be inserted into many different contexts on a page:

- HTML element
- HTML attribute
- URL query parameter
- CSS value
- Javascript value
- ...
- or a combination of the above

```
<h1>Hello <%= name %></h1>

<input value="<%= name %>">

<a href="/?name=<%= name %>">

<style> h1 {color: <%= name %>; }

<script>
    var name='<%= name %>';
</script>
```

Go back to Page 14.

Fixing XSS

Fix the output:

Understand the context that you're outputting data:

- HTML
- URL
- Attribute
- Javascript
- ...

Encode data to make it safe in that context

Or, fix the input:

Restrict the input to your application to only safe characters

- Validation
- Whitelists

If you've fixed all your XSS, goto Page 21.

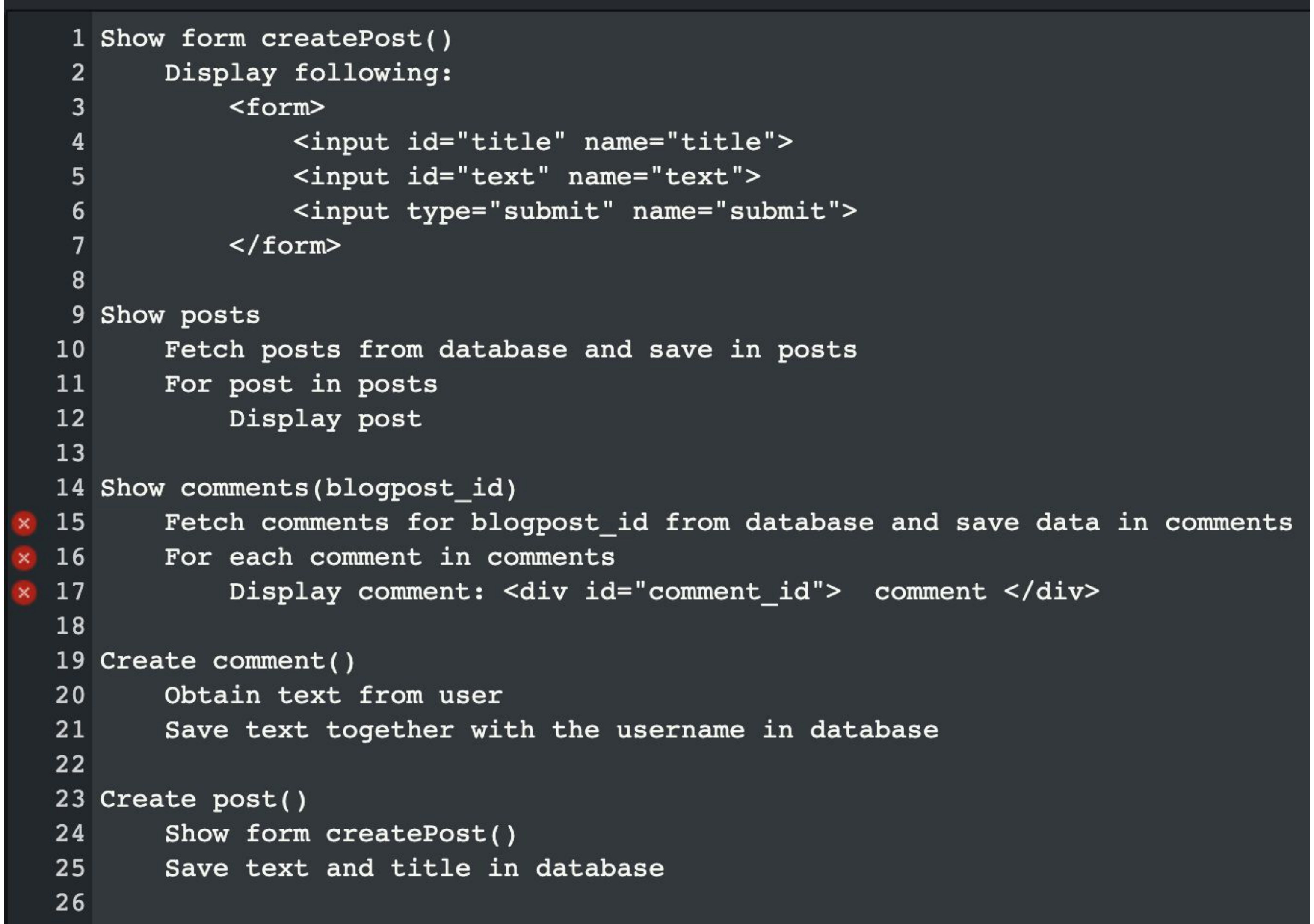
If you've got known XSS to fix, go to Page 20.

CHOOSE YOUR OWN ADVENTURE®
25
YOU'RE THE STAR OF THE STORY! CHOOSE FROM 28 POSSIBLE ENDINGS
A BANTAM BOOK ★ $1.95 ★ 23561-X

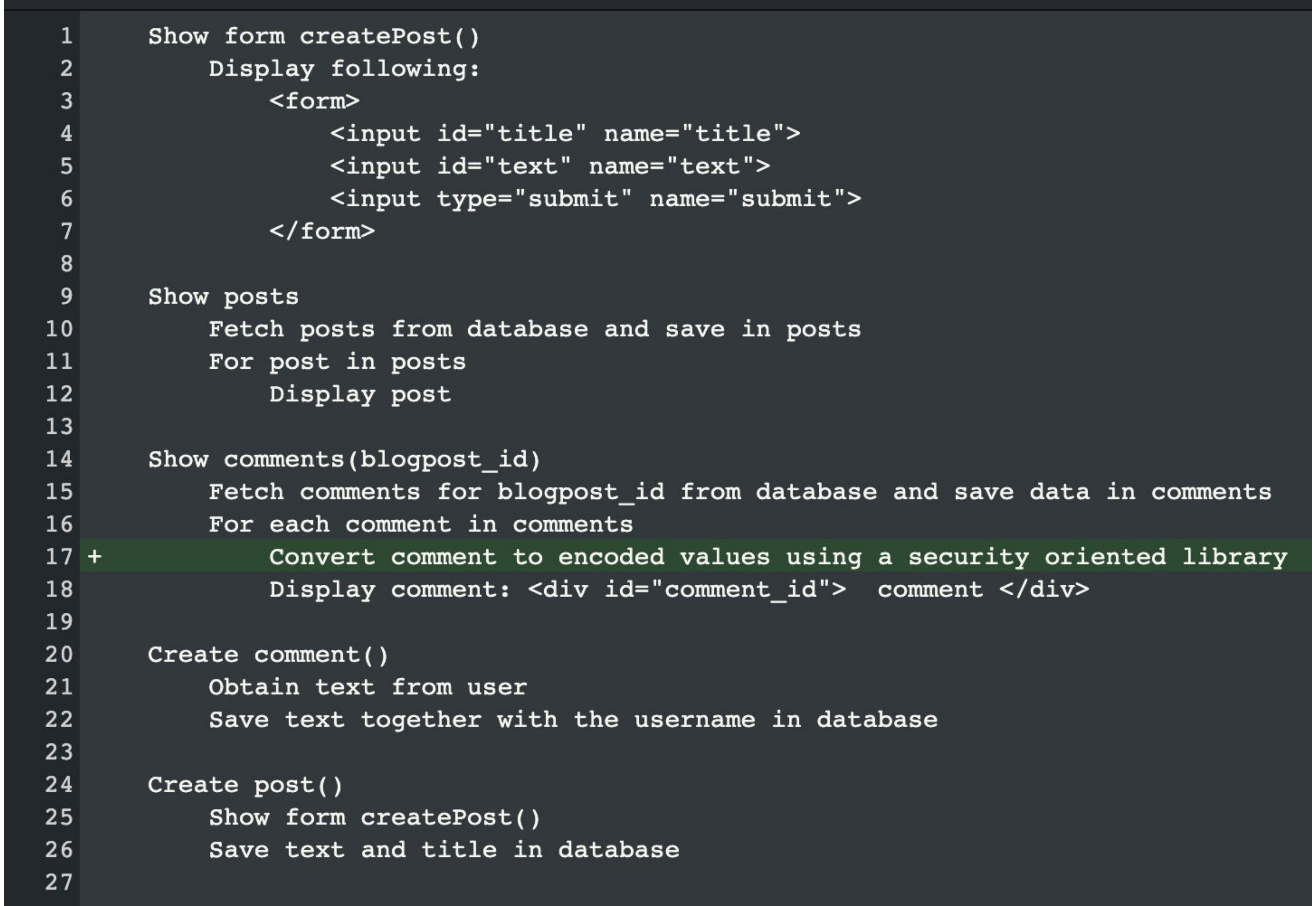
The Attack of the Deadly XSS!

BY R.A. MONTGOMERY

www.owasp.org.nz



```
Show form createPost()
    Display following:
        <form>
            <input id="title" name="title">
            <input id="text" name="text">
            <input type="submit" name="submit">
        </form>

Show posts
    Fetch posts from database and save in posts
    For post in posts
        Display post

Show comments(blogpost_id)
    Fetch comments for blogpost_id from database and save data in comments
    For each comment in comments
        Display comment: <div id="comment_id">  comment </div>

Create comment()
    Obtain text from user
    Save text together with the username in database

Create post()
    Show form createPost()
    Save text and title in database

```



```
  Show form createPost()
      Display following:
          <form>
              <input id="title" name="title">
              <input id="text" name="text">
              <input type="submit" name="submit">
          </form>

  Show posts
      Fetch posts from database and save in posts
      For post in posts
          Display post

  Show comments(blogpost_id)
      Fetch comments for blogpost_id from database and save data in comments
      For each comment in comments
+         Convert comment to encoded values using a security oriented library
          Display comment: <div id="comment_id">  comment </div>

  Create comment()
      Obtain text from user
      Save text together with the username in database

  Create post()
      Show form createPost()
      Save text and title in database

```

```
Name = kirk&<'"
```

```
<h1>Hello <%= name %></h1>
```

HTML Encoding:

```
<h1>Hello kirk&amp;&lt;'"</h1>
```

```
<input value="<%= name %>">
```

Attribute Encoding:

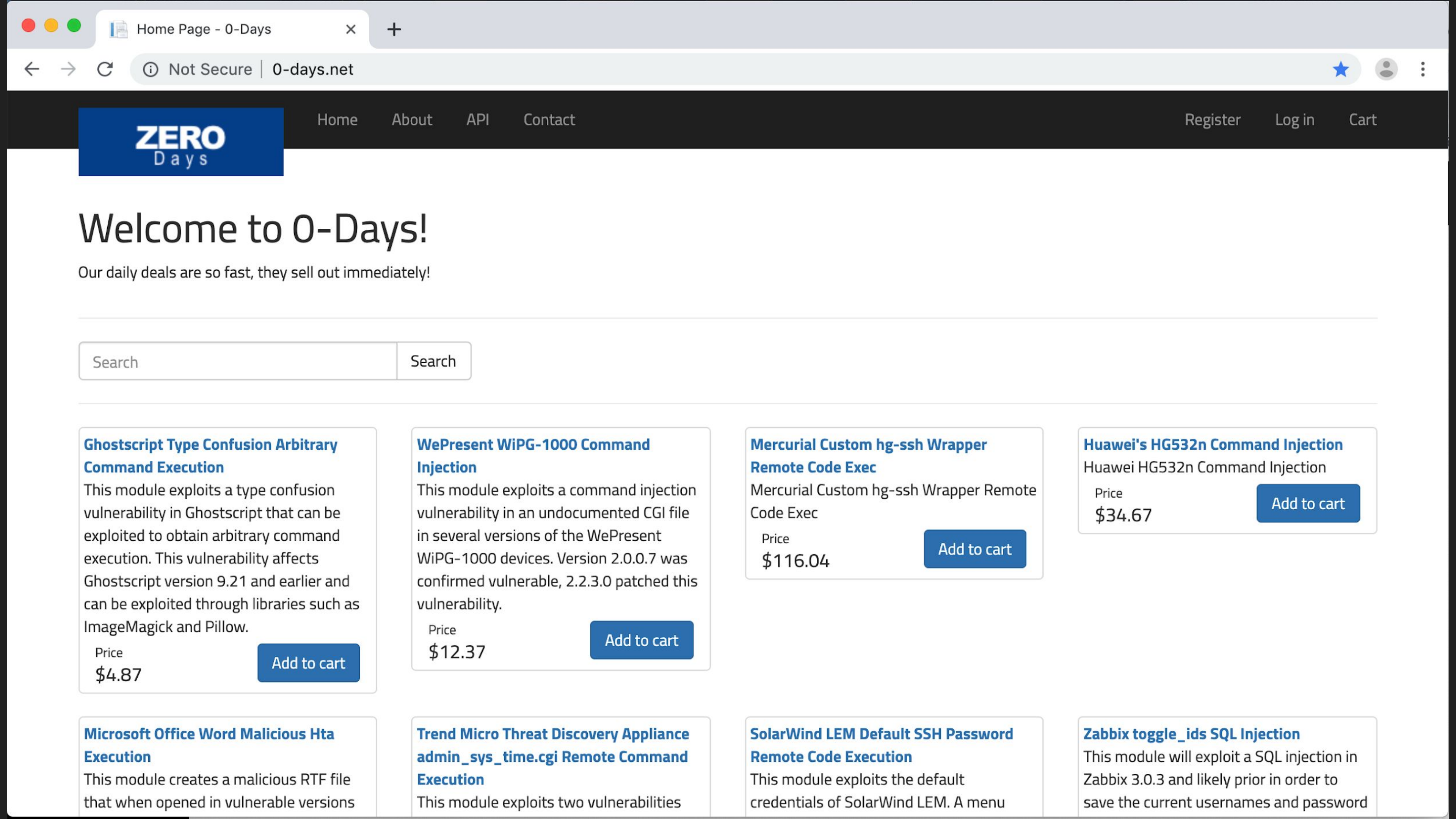
```
<input value="kirk&amp;&lt;'&quot;">
```

```
<a href="/?name=<%= name %>">
```

URL Encoding:

```
<a href="/?name=kirk%26%3c%27%22">
```

SQL Injection



Welcome to 0-Days!

Our daily deals are so fast, they sell out immediately!

Ghostscript Type Confusion Arbitrary Command Execution
This module exploits a type confusion vulnerability in Ghostscript that can be exploited to obtain arbitrary command execution. This vulnerability affects Ghostscript version 9.21 and earlier and can be exploited through libraries such as ImageMagick and Pillow.
Price
$4.87
Add to cart

WePresent WiPG-1000 Command Injection
This module exploits a command injection vulnerability in an undocumented CGI file in several versions of the WePresent WiPG-1000 devices. Version 2.0.0.7 was confirmed vulnerable, 2.2.3.0 patched this vulnerability.
Price
$12.37
Add to cart

Mercurial Custom hg-ssh Wrapper Remote Code Exec
Mercurial Custom hg-ssh Wrapper Remote Code Exec
Price
$116.04
Add to cart

Huawei's HG532n Command Injection
Huawei HG532n Command Injection
Price
$34.67
Add to cart

Microsoft Office Word Malicious Hta Execution
This module creates a malicious RTF file that when opened in vulnerable versions

Trend Micro Threat Discovery Appliance admin_sys_time.cgi Remote Command Execution
This module exploits two vulnerabilities

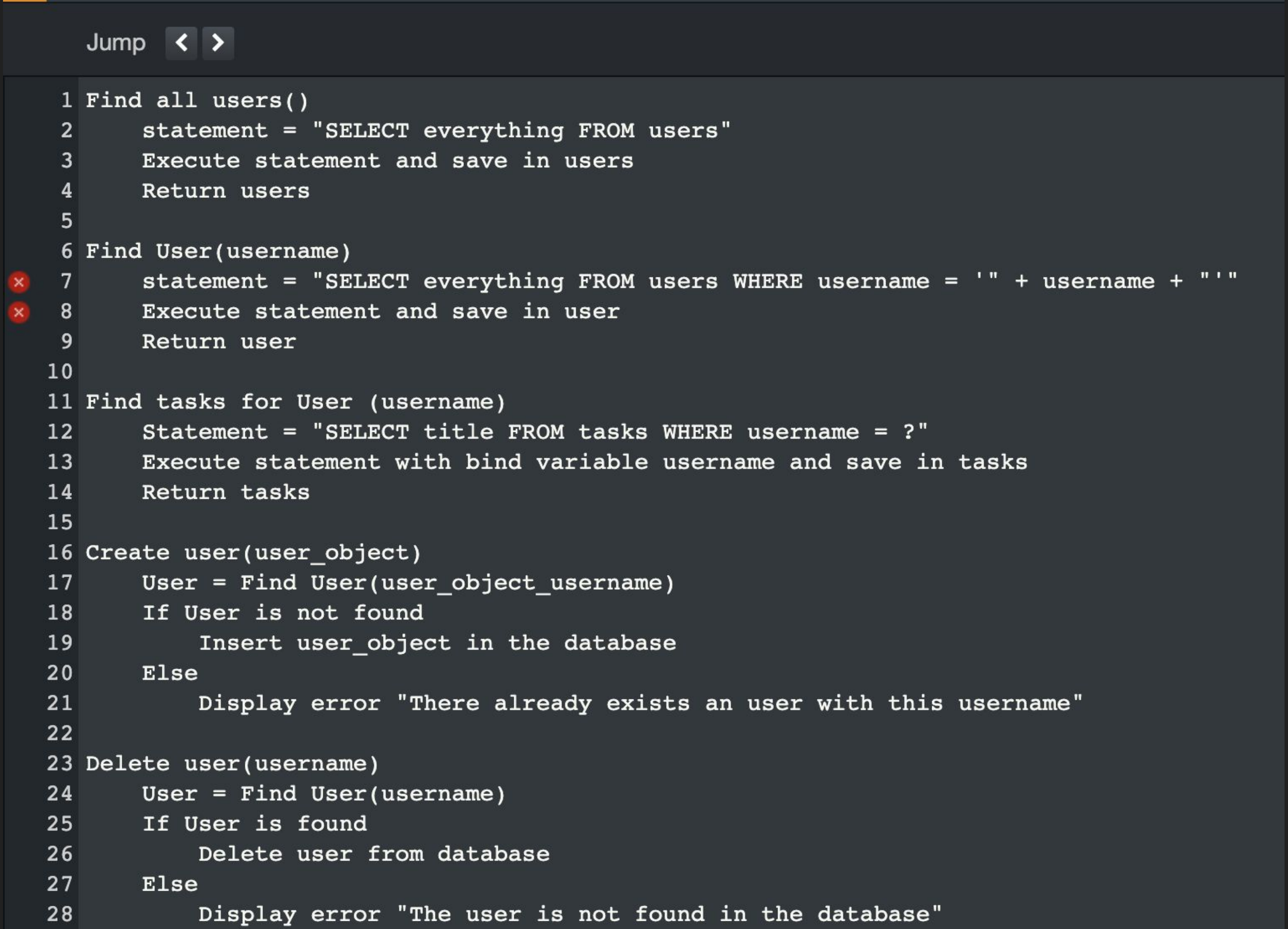
SolarWind LEM Default SSH Password Remote Code Execution
This module exploits the default credentials of SolarWind LEM. A menu

Zabbix toggle_ids SQL Injection
This module will exploit a SQL injection in Zabbix 3.0.3 and likely prior in order to save the current usernames and password

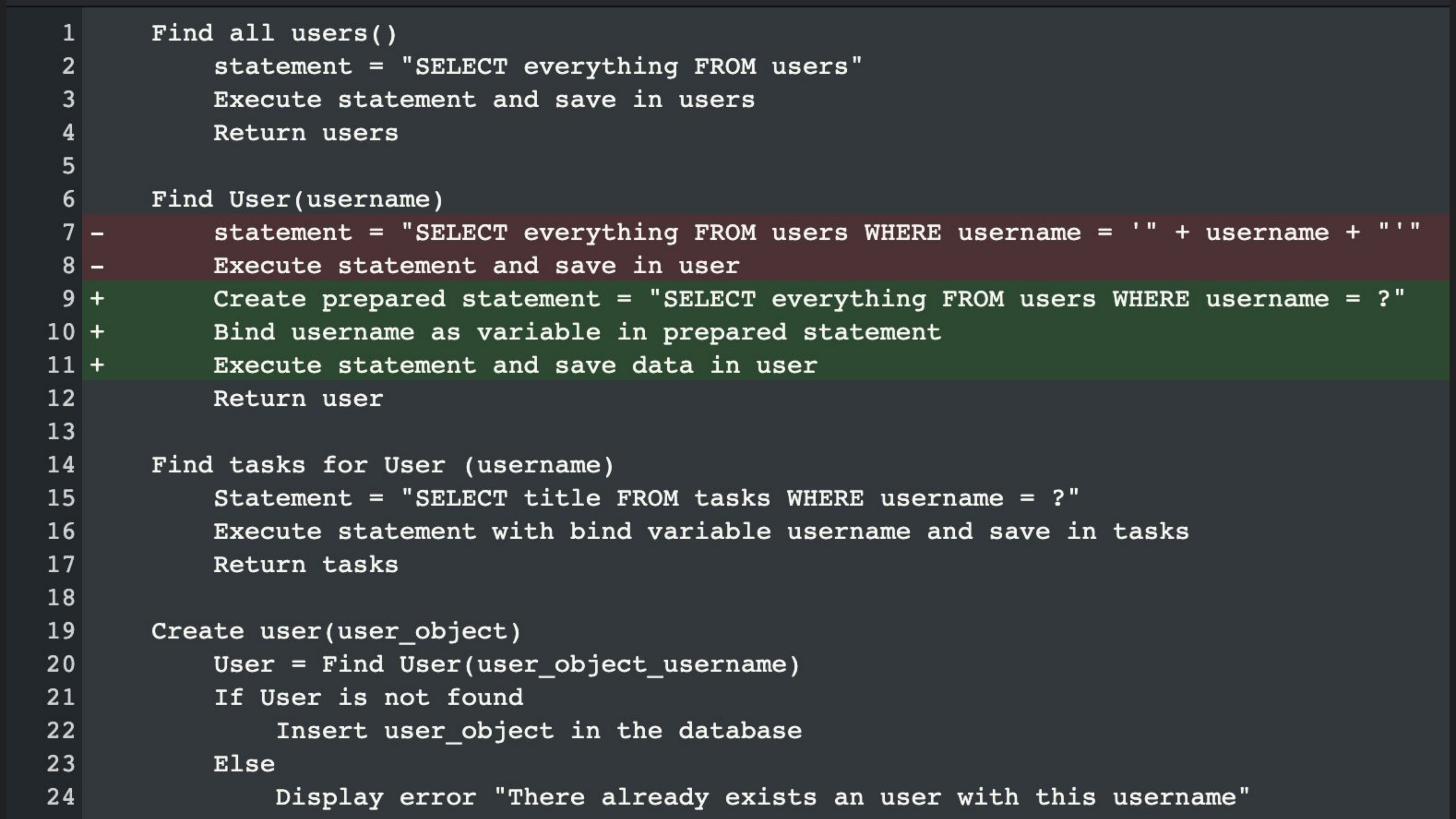
SQL Injection

Video of sqlmap

Files extracted



```
Find all users()
    statement = "SELECT everything FROM users"
    Execute statement and save in users
    Return users

Find User(username)
    statement = "SELECT everything FROM users WHERE username = '" + username + "'"
    Execute statement and save in user
    Return user

Find tasks for User (username)
    Statement = "SELECT title FROM tasks WHERE username = ?"
    Execute statement with bind variable username and save in tasks
    Return tasks

Create user(user_object)
    User = Find User(user_object_username)
    If User is not found
        Insert user_object in the database
    Else
        Display error "There already exists an user with this username"

Delete user(username)
    User = Find User(username)
    If User is found
        Delete user from database
    Else
        Display error "The user is not found in the database"
```



```
    Find all users()
        statement = "SELECT everything FROM users"
        Execute statement and save in users
        Return users

    Find User(username)
-       statement = "SELECT everything FROM users WHERE username = '" + username + "'"
-       Execute statement and save in user
+       Create prepared statement = "SELECT everything FROM users WHERE username = ?"
+       Bind username as variable in prepared statement
+       Execute statement and save data in user
        Return user

    Find tasks for User (username)
        Statement = "SELECT title FROM tasks WHERE username = ?"
        Execute statement with bind variable username and save in tasks
        Return tasks

    Create user(user_object)
        User = Find User(user_object_username)
        If User is not found
            Insert user_object in the database
        Else
            Display error "There already exists an user with this username"
```

```
GET /Product/Search?SearchTerm=ghost HTTP/1.1
Host: www.0-days.net
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_5)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/76.0.3809.100
Safari/537.36
Accept:
text/html,application/xhtml                              ge/webp,i
mage/apng,*/*;                                          ge;v=b3
Referer: htt
Accept-Encodi            , deflate
Accept-Language: en-GB,en-US;q=0.9,en;q=0.8
Cookie:
.AspNetCore.Antiforgery.9TtSrW0hzOs=CfDJ8JVmqLgybchGooENk8b3J2Arp7
JPwBPHmd6ZFeABp7WkL3Oad7vmVBUmgjuLe7B3p8KApo1sdYkvdxqdkwqN1XS3YjCV
eoOlLfwdrFSH8PltvmwuVnhUJNpl3pF3ys9YA8LISJVZAeSo69A2QYDedxc
```

Don’t trust anything!

Don’t trust user input

Kirk Jackson, RedShield

security.ac.nz, 25 Aug 2019