

Automating Almost All Application Security Things with CI/CD

Even Honeypots!

Mick Douglas and Andy Douglas



```
graph LR; A[Development] --> B[Continuous Integration]; B --> C[Continuous Delivery or Deployment];
```

Development

Continuous
Integration

Continuous Delivery
or Deployment

Development

Continuous
Integration

Continuous Delivery
or Deployment

SECURITY ISSUES - 3

- app.js - 2 issues
 - Timing Attack
 - Cross-Site Scripting attack
- server.js - 1 issue
 - Path Transversal Vulnerability**

CODE ISSUES - 1

- standalone.js - 2 issues
 - Using component state to comput ...
 - http (used in require) is an insecure...
- Header.js - 1 issue
 - Unsanitized input flows from the H...
- Code.js - 1 issue
 - Testing a collection size for >= 0 wi...
- contentScript.js - 1 issue
 - Setting targetOrigin to "" in postM...
- standalone.js - 1 issue
 - Signature mismatch: the implement...

```
8 const mime = require('mime');
9
10 function sendFile(filename, response) {
11   response.setHeader('Content-Type', mime.lookup(filename));
12   response.writeHead(200);
13   const fileStream = createReadStream(filename);
14   fileStream.pipe(response);
15   fileStream.on('finish', response.end);
16 }
17
18 function createHTTP2Server(benchmark) {
19   const server = http2Server.createServer({}, {request,
20     const filename = join(
21       __dirname,
22       'benchmarks',
23       benchmark,
24       request.url
25     ).replace(/\/?.*/g, '');
26
27   if (existsSync(filename) && statSync(filename).isFile)
28     sendFile(filename, response);
29   } else {
30     const indexHtmlPath = join(filename, 'index.html')
31
32     if (existsSync(indexHtmlPath)) {
33       sendFile(indexHtmlPath, response);
34     } else {
35       response.writeHead(404);
36       response.end();
37     }
38   }
39 }
```

Code flow

- scripts/bench/server.js line #24
- scripts/bench/server.js line 20
- scripts/bench/server.js line 13

External example fixes

This issue was fixed by 708 projects. Here are 3 example fixes.

georgi/grant

```
}
- function static_file(route, p, req, res) {
-   var uri = url.parse(req.url).pathname;
+ function dump_static_file(route, p, req, res) {
+   var uri = url.parse(this.req.url).pathname;
   var filename = path.join(process.cwd(), 'public', uri);
   fs.exists(filename, function(exists) {
     if(!exists) {
       var fileStream = fs.createReadStream(filename);
-     fileStream.pipe(res);
+     fileStream.pipe(this.res);
     });
  });
```

Share issue Ignore issue Feedback

Development

Continuous
Integration

Continuous Delivery
or Deployment



sonarcloud bot commented 3 minutes ago



SonarCloud Quality Gate failed.

Failed



0 Bugs



0 Vulnerabilities



0 Security Hotspots



1 Code Smell



No Coverage information




0.0% Duplication

Development

Continuous
Integration

Continuous Delivery
or Deployment

```
Windows PowerShell
PS C:\Users\AndyDouglas> docker run --rm -t --network host owasp/zap2docker-stable:2.12.0
zap-baseline.py -t http://localhost:3000 -s_
```



Development

Continuous
Integration

Continuous Delivery
or Deployment

Active Defense

Automate Application Security with CI/CD

Part 1: Dynamic Scans for prevention

Part 2: Active Defense for detection

Goal: Actionable Security Improvement

Hi, my name is Andy Douglas

- Long-time CodeMash attendee, first-time speaking
- Full Stack Software Dev/Engineer > Architect > Engineering Manager
- Security...meh

Hi, my name is Mick Douglas

- First time CodeMash attendee
- Passionate about security
- Infosec Innovations, SANS Principle Instructor, IANS Research Faculty

Part 1: Dynamic Scans for prevention



SAST - quick word

Recommendation:

Integrate SAST w/ IDE + CI

Popular Tools: Snyk, Checkmarx, SonarQube/SonarCloud

Dynamic Application Security Testing (DAST)



```
graph LR; A[Merge into main] --> B[CI build runs (hopefully with SAST)]; B --> C[CD build runs to deploy to env X]; C --> D[Smoke tests and DAST];
```

Merge into main

CI build runs
(hopefully with SAST)

CD build runs
to deploy to env X

Smoke tests
and **DAST**

4 Labs

DAST Lab

DAST Lab: deploy vulnerable web app

DAST Lab: Automate a leading DAST tool

DAST Lab: Learn how to include
DAST in your pipelines

DAST Strengths

DAST Weaknesses

Recommendation:

DAST: CD and/or scheduled

Popular Tools: ZAP, StackHawk, Burp Suite, Astra Pentest, etc.

Part 2: Active Defense for detection



Part 2: Active Defense for ~~detection~~ FTW!!



WAF

Traditional Defense == Passive

Traditional Defense == KNOWN

Active Defense == Passive & Active

Shift focus/effort to higher reward

Attackers have predictable paths

Active Defence at each phase

Recon: Ports

Honey Port

DEMO: Honey Port

Privilege Escalation: session tokens

Honey Tokens

Demo: Honey Tokens

Strategy: Defense Options

Watch and learn?

Random error response?

Firewall

QoS

Demo response options

Security != Hard/Expensive

Security as Functional Requirement

Security + CI/CD is good

LAB IT UP!!