

Introduction to Security

Static and Dynamic Analysis

Ming Chow (mchow@cs.tufts.edu)

Twitter: @0xmchow



Learning Objectives

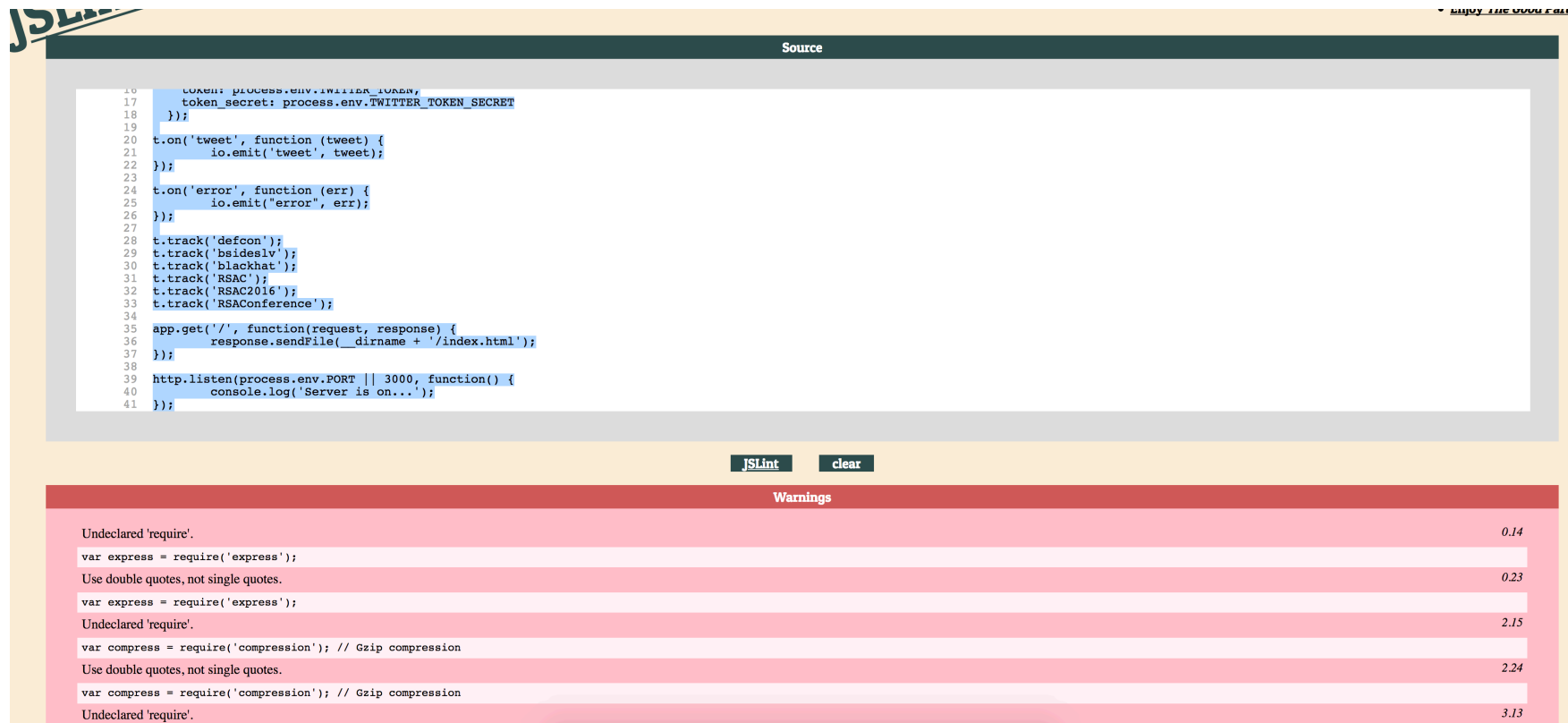
- By the end of this week, you will be able to:
 - Use static analysis software to identify vulnerabilities in a software
 - Understand the difference between static and dynamic analysis

Static Analysis

- Also known as static code analysis
- No execution of program
- Rule based
- *Full code coverage*
- Will catch bugs in source code such as using insecure or unsafe functions
- Binary static analysis: black box, no code
- Code: white box , given source code
- Examples: grep, lint, Coverity (commercial), Fortify (commercial), Veracode (commercial)
- Reference: <https://www.veracode.com/products/static-analysis-sast/static-code-analysis>

Tool: JSLint (Lint for JavaScript)

- <http://www.jshint.com/>



The screenshot shows the JSHint web interface. The top section, titled "Source", contains a text area with JavaScript code. The code includes variable declarations for `token` and `token_secret`, event listeners for `tweet` and `error`, tracking calls for various events, and an Express.js server setup. Below the code area are two buttons: "JSHint" and "clear". The bottom section, titled "Warnings", displays a list of linting errors. The errors include "Undeclared 'require'." and "Use double quotes, not single quotes." with corresponding line numbers.

```
16 token: process.env.TWITTER_TOKEN,  
17 token_secret: process.env.TWITTER_TOKEN_SECRET  
18 });  
19  
20 t.on('tweet', function (tweet) {  
21   io.emit('tweet', tweet);  
22 });  
23  
24 t.on('error', function (err) {  
25   io.emit("error", err);  
26 });  
27  
28 t.track('defcon');  
29 t.track('bsideslv');  
30 t.track('blackhat');  
31 t.track('RSAC');  
32 t.track('RSAC2016');  
33 t.track('RSACConference');  
34  
35 app.get('/', function(request, response) {  
36   response.sendFile(__dirname + '/index.html');  
37 });  
38  
39 http.listen(process.env.PORT || 3000, function() {  
40   console.log('Server is on...');  
41 });
```

Warnings

Undeclared 'require'.	0.14
var express = require('express');	
Use double quotes, not single quotes.	0.23
var express = require('express');	
Undeclared 'require'.	2.15
var compress = require('compression'); // Gzip compression	
Use double quotes, not single quotes.	2.24
var compress = require('compression'); // Gzip compression	
Undeclared 'require'.	3.13

A Glance at Static Analysis Techniques

1. Data flow analysis
 - Collect runtime info about data while in a static state
 - Basic block (the code), control flow, control path
2. Control graph
 - Node => block
 - Edges => jumps / paths
3. Taint Analysis (also Deterministic Finite Automaton)
 - Identify variables that have been tainted
 - Used vulnerable functions known as sink
4. Lexical analysis
 - code => tokens (e.g., /* gets */)

Strengths and Weaknesses of Static Analysis

- Strengths:
 - Find vulnerabilities with high confidence
- Weaknesses:
 - Many false positives or false negatives can be generated
 - Can't find configuration issues
 - Can you prove findings are actual vulnerabilities?

Dynamic Analysis

- System execution; run-time
- Trial and error
- Detect dependencies
- Deal with real runtime variables
- Based on automated tests, user interactions
- No guarantee of full coverage of source
- Example: valgrind – for memory debugging, memory leak detection, and profiling. <http://valgrind.org/>