NodeJS Security
Still unsafe at most speeds

@DinisCruz

London, 29th Sep 2016
Me

• Developer for 25 years
• AppSec for 13 years
• Day jobs:
  • Leader OWASP O2 Platform project
  • Application Security Training for JBI Training
  • Part of AppSec team of:
    • The Hut Group
    • BBC
• AppSec Consultant and Mentor
  • “I build AppSec teams….”
• https://twitter.com/DinisCruz
• http://blog.diniscruz.com
• http://leanpub.com/u/DinisCruz
Contact

- @Leanpub (get for 0$)
- http://leanpub.com/u/DinisCruz
Recent Presentations (you might find interesting)

- Start with passing Tests (TDD for bugs)
  London, 22nd Sep 2016, v0.5

- Using JIRA to manage Risks and Security Champions activities
  OWASP AppSecEU, Rome, 2016

- APPSEC AND SOFTWARE QUALITY
  FILE: V0.5 (MAY/2016)
  LONDON
  @DINISCRAZ
AppSec and Quality

My thesis is that Application Security can be used to define and measure Software Quality.

MODERN APPLICATION SECURITY
- TDD with Code Coverage
- Threat Models
- Docker and Containers
- Test Automation
- SAST/DAST/IAST/WAF
- Clever Fuzzing
- JIRA Risk workflows
- Kanban for Quality fixes
- Web Services visualisation
- ELK

TECHNICAL DEBT IS A BAD ANALOGY
- The developers are the ones who pays the debt
- Pollution is a much better analogy
- The key is to make the business accept the risk (i.e. the debt)
- Which is done using the JIRA RISK Workflows
Key to AppSec - The AppSec Risk Workflow

RISK Workflow (using JIRA in Cloud)

Start with Passing tests, because:

When creating tests on the ‘Fix’ stage, the focus (and time allocated) is on fixing the bug (not on testing it)

When creating tests on the ‘Issue Creation’ stage, the focus (and time allocated) is on how to test it and what is its root cause

NODEJS SECURITY
Basically....

• Just as good and bad as Java or .NET
• We are still in the same place
• Not many lessons learned
• But at least we are building bigger and faster websites (with more house-power and assets)
What is good 1/3

- native JSON
- super fast
  - V8 Engine executed some javascript code faster than (equivalent) C++
- async pattern
  - one event loop thread
  - highly scalable
- developer friendly
  - fast development
  - REPL (Read, Eval, Print, Loop)
  - enables CI and CD (easy integration with GitHub, Travis, etc…)
- Other languages
  - ECMAScript 6
  - CoffeeScript (my favourite language)
  - Jade (Html template engine)
  - Typescript
What is good 2/3

- community Innovation
  - pure Open Source child (with strong corporate support)
  - equivalent io.js fork should had happened to Java and .NET
  - crazy innovation speed and technologies like JsDOM
  - NodeJS Security Project

- ssl is easy

- enterprise ready
  - used by massive sites with great success
  - amazing live monitoring and instrumentation tools (and SAAS solution)
  - container friendly (i.e. docker)

- promotes Microservices

- great test culture (TDD)

- growing security maturity
  - null checks on file paths

OWASP
Open Web Application Security Project
WWW.OWASP.ORG
What is good 3/3

- WallabyJS
  - real time unit test execution
  - real time code coverage
Just to be clear....

\texttt{nodeJS + CoffeeScript + wallaby}

is my most productive and enjoyable dev environment

where I easily write secure code with 100\% code coverage
• Same old OWASP Top 10
• Have to work hard to write secure apps
  – not out of the box
  – CSRF protection for example
• REST Injection
  – can be as bad as SQL Injection
• Model Binding is alive
What is bad 2/5

• It’s Javascript
  – not strongly typed
    • with crazy type conversions and equals
    • decimal conversion problems
  – ability to overwrite (via prototypes) other API’s methods
  – interpreted code (strings can become code)
    • Eval, file save or ‘dynamic requires’ can lead to RCE

• Strings everywhere (we have to ‘ban strings’)

• Pattern: Proxy to internal Systems (with no data validation checks for more data)
What is bad 3/5

• NPM
  – just as bad and crazy as Maven, NuGet, CocoaPods
  – very little security checks performed in new modules
    • few security eyeballs
    • dependency checks via https://nodesecurity.io/ via nsp
  – just look at what is inside some npm packages
What is bad 4/5

- Unhanded errors will crash server (can be a good thing)
- Server side HTML and Javascript generation
  - source of tons of XSS
- Secure configuration is hard
- Weak code visualisation for
  - Attack surface
  - AST
  - Code Paths
- Limited support for sandboxing code and CAS (Code Access Security)
What is bad 5/5

• Hard to do SAST (Static Analysis)
• NoSQL databases vulnerable to Injection attacks
• Express support for ..%2f in url segments
• … I’m sure there are many more …
OWASP AND NODEJS
OWASP Top 10 (for 2013) is all there

• A1 Injection
• A2 Broken Authentication and Session Management
• A3 Cross-Site Scripting (XSS)
• A4 Insecure Direct Object References
• A5 Security Misconfiguration
• A6 Sensitive Data Exposure
• A7 Missing Function Level Access Control
• A8 Cross-Site Request Forgery (CSRF)
• A9 Using Components with Known Vulnerabilities
• A10 Unvalidated Redirects and Forwards
OWASP Juice Shop Tool Project

I The most trustworthy online shop out there. (dschadow)

OWASP Juice Shop is an intentionally insecure webapp for security trainings written entirely in Javascript which encompasses the entire OWASP Top Ten and other severe security flaws.

Description

Juice Shop is written in Node.js, Express and AngularJS. It was the first application written entirely in JavaScript listed in the OWASP VWA Directory.

The application contains more than 30 challenges of varying difficulty where the user is supposed to exploit the underlying vulnerabilities. The hacking progress is tracked on a score board. Finding this score board is actually one of the (easy) challenges!

Apart from the hacker and awareness training use case, pentesting proxies or security scanners can use Juice Shop as a "guinea pig"-application to check how well their tools cope with Javascript-heavy application frontends and REST APIs.

I Translating "dump" or "useless outfit" into German yields "Saftladen" which can be reverse-translated word by word into "juice shop". Hence the project name.
OWASP NodeGoat Project

OWASP NodeGoat project provides an environment to learn how OWASP Top 10 security risks apply to web applications developed using Node.js and how to effectively address them.

Introduction

Being lightweight and efficient, Node.js is rapidly becoming a platform of choice for building fast, scalable, data-intensive, modern web applications. However, developing stable and resilient web applications on this platform is very dependent on programmers due to its minimal default configuration and architecture choices. The goal of this project is to act as a learning resource demonstrating how OWASP Top 10 security risks apply to web applications developed using Node.js and how to effectively address them. It includes a vulnerable web application and accompanied tutorial guide.

Description

- Demo app: http://nodegoat.herokuapp.com/
- Project source code: https://github.com/OWASP/NodeGoat
- Gitter chat: https://gitter.im/OWASP/NodeGoat

Project Leader

Chetan Karande

Quick Download

- Clone project Github repository at https://github.com/OWASP/NodeGoat

Classifications

Builders
Defenders
CC BY-SA
NodeJS Security Book

https://secureyournodejs.com
KNOW THE RISK OF YOUR APPLICATION
View security issues as features

• You need to have them mapped and accept the risk

• Here are the risks currently accepted for the OWASP/Maturity-Models project (NodeJS app)

  – https://github.com/OWASP/Maturity-Models
...using GitHub Labels to create Risk Workflow

- A1 - Injection
  - risk - accepted
  - security
- A2 - Broken Authentication
- A6 - Sensitive Data Exposure
- A11 - DoS
  - bug
  - ci
  - duplicate
  - hack
  - help wanted
  - invalid
- new feature
  - P0
  - P1
  - P2
  - P3

- quality
- question
- refactor
- research
- risk - accepted
- risk - fixed
- risk - high
- risk - low
- risk - medium
- risk - to accept
- risk - to fix
- security
- test needed
<table>
<thead>
<tr>
<th>Issue</th>
<th>Title</th>
<th>Tags</th>
<th>Closing Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>#8</td>
<td>Add support for SSL</td>
<td>A6 - Sensitive Data Exposure</td>
<td>Jun 8</td>
</tr>
<tr>
<td>#9</td>
<td>Add security tests for lack of SSL</td>
<td>risk - accepted</td>
<td>Jun 8</td>
</tr>
<tr>
<td>#16</td>
<td>There is no Authentication and Authorization</td>
<td>A2 - Broken Authentication</td>
<td>Jun 3</td>
</tr>
<tr>
<td>#17</td>
<td>There is no data classification of assets used</td>
<td>A6 - Sensitive Data Exposure</td>
<td>Jun 8</td>
</tr>
<tr>
<td>#18</td>
<td>Api-Controller - filename is a string and it is not validated</td>
<td>risk - accepted</td>
<td>Jun 8</td>
</tr>
<tr>
<td>#22</td>
<td>Write regression test to prove that Data-Files.find method is not</td>
<td>A1 - Injection</td>
<td>Jun 8</td>
</tr>
<tr>
<td>#26</td>
<td>Data_Files.set_File_Data - DoS via file_Contents</td>
<td>A1 - Injection</td>
<td>Jun 3</td>
</tr>
<tr>
<td>#30</td>
<td>All server logs are exposed via API</td>
<td>A6 - Sensitive Data Exposure</td>
<td>Jun 8</td>
</tr>
</tbody>
</table>
- Server web root (i.e. path) is exposed by API
  - A6 - Sensitive Data Exposure
  - Risk: accepted
  - Risk: low
  - Test needed

- All data can be modified by web users
  - A2 - Broken Authentication
  - Risk: accepted
  - Risk: medium
  - Test needed

- Data is not saved automatically on local and QA server
  - P1
  - Risk: accepted
  - Risk: medium
  - Test needed

- Duplicate team names are allowed and file list is not able to handle them
  - Bug
  - Risk: accepted

- Support for coffee file to create dynamic data sets allow RCE
  - A1 - Injection
  - Risk: accepted

- Project list gets data from File System and allows DoS (with large amounts of requests)
  - A11 - DoS
  - Risk: accepted
  - Security

- There is no Threat Model for this application
  - Risk: accepted
  - Risk: medium
  - Security

- DoS on Data-Project technique to map projects and project's teams
  - A11 - DoS
  - Risk: accepted
  - Risk: low
  - Security
- App will have issues if hosted in a multi-process environment
  #122 by DinisCruz was closed on Jul 10

- There is no Attack Detection or 'AppSensor like' capabilities
  #133 by DinisCruz was closed on Jul 11

- Users are able to delete teams
  #137 by DinisCruz was closed on Jul 14

- There is a CSRF vuln on Add and Delete teams
  #138 by DinisCruz was closed on Jul 14

- Application has no ability to set file based permissions for Data repos
  #145 by DinisCruz was closed on Jul 20

- App is vulnerable to "AngularJS Sandbox Bypass Collection"
  #153 by DinisCruz was closed 13 days ago

- set_File_Data does not provide detailed information on why it failed
  #155 by DinisCruz was closed on Aug 11

- Application is able to write to App root
  #156 by DinisCruz was closed on Aug 11
CASE STUDY: WHEN I CREATED A VULNERABILITY
Feature request: Allow data editing on UI

- Here is the code I wrote (at the Data Layer):

```python
56 set_File_Data: (filename, file_Contents) ->
57     if not filename or not file_Contents
58         return null
59     if typeof file_Contents isn't 'string'
60         return null
61     file_Path = @.find filename
62     if file_Path is null or file_Path.file_Not_Exists()
63         file_Path = @.data_Path.path_Combine filename
64     file_Path.file_Write file_Contents
65     return file_Path
```

- This method is designed to be called by the controller (i.e. rest api endpoint):
Current implementation of Data_Files.set_File_Data (here and below) is vulnerable by design to an Path Traversal attack.

This will allow any caller to write into files outside the expected data folder

```javascript
set_File_Data: (filename, file_Contents) =>
  if not filename or not file_Contents
    return null
  if typeof file_Contents isnt 'string'
    return null
  file_Path = @.find filename
  if file_Path is null or file_Path.file_Not_Exists()
    file_Path = @.data_Path.path_Combine filename
  file_Path.file_Write file_Contents
  return file_Path
```
describe '_security | A1 - Injection', ->

# https://github.com/DinisCruz/BSIMM-Graphics/issues/21
it 'Issue 19 - Data_Files.set_File_Data - Path Traversal', ->
    using new Data_Files(), ->
        folder_Name = 'outside-data-root'
        file_Name = 'some-file.txt'
        file_Content = 'some content'
        target_Folder = @.data_Path.path_Combine('../' + folder_Name)  # Create target
            .folder_Create()
            .assert_Folder.Exists()
        # Confirm it exists.

        target_Folder.path_Combine(file_Name)
            .file_Write(file_Content)
            .assert_File.Exists()  # Confirm it exists.
        payload = "../#{folder_Name}/#{file_Name}"
        new_Content = 'new - content'

        @.data_Path.path_Combine(payload)
            .file_Contents().assert_Is file_Content  # Confirm original.

        @.set_File_Data payload, new_Content

        @.data_Path.path_Combine(payload)
            .file_Contents().assert_Is_Not file_Content  # Confirm original.
            .assert_Is new_Content  # Confirm that.

        target_Folder.folder_Delete_Recursive().assert_Is_True()  # Delete temp for.
As seen in #19 the set_File_Data: (filename, file_Content) method does not check the size (and contents) of the filename and file_Content variables.

The problem is that they are strings, which means that they can be huge:

- http://appsandsecurity.blogspot.co.uk/2013/05/should-string-be-abstract-class.html

And since those values are used to on the name and contents of files written on disk, in addition to possible probs in the Node Heap, this function can be used to fill up the disk

Here is the test for this issue which proves that we can create large files and also detects some weird behaviours on the file name size (which is different in wallaby, mocha and travis)
it.only 'Issue 20 - Data Files.set File Data - DoS via filename and file Contents', ->
  using new Data_Files(), ->
  create_File = (file_Size, content_Size, should_Work) =>
    file_Name        = file_Size      .random_String()
    file_Contents    = content_Size   .random_String()
    file_Path        = @.data_Path     .path_Combine(file_Name)

    file_Path.assert_File_Not_Exists()  # confirm file doesn't exist

    @.set_File_Data file_Name, file_Contents  # PAYLOAD: create file

    if should_Work
      file_Path.assert_File_Exists()  # if it should work
      file_Path.file_Delete().assert_Is_True()  # delete temp file
      file_Path.assert_File_Not_Exists()  # if not
    else
      file_Path.assert_File_Not_Exists()  # confirm creation failed

  # testing multiple file sizes
  create_File 10,10, true
  create_File 100,10, true
  create_File 156,10, true
  #create_File 157,10, false  # interesting in wallaby, and
  #create_File 208,10, false  # in mocha, it's
  create_File 512,10, false  # in travis they

  # testing multiple file contents
  create_File 10,10, true  # 10 bytes
  create_File 10,100, true  # 100 bytes
  create_File 10,10000, true  # 10 Kb
  create_File 10,100000, true  # 1 Mb
  create_File 10,1000000, true  # 10 Mb - will work and take
  create_File 10,10000000, true  # 100 Mb - will work and take

  done();
Related to #19 and #20, at the moment there is no limitations on the type of files that can be saved.

According with the current design, the only file paths that should be supported are .json files.

Here is the test that proves the issue.

```javascript
it 'Issue 23 - Data_Files.set_File_Data - allows creation of files with any extension', ->
  using new Data_Files(), -->
  create_File = (extension) =>
    file_Name = 10.random_String() + extension
    file_Contents = 10.random_String()
    file_Path = @.data_Path.path_Combine(file_Name)

    @.set_File_Data file_Name, file_Contents
    # PAYLOAD: create file

    file_Path.assert_File.Exists()
    .file_Delete().assert_Is_TRUE()
    # confirm file exists
    # delete temp file

    create_File '.json'
    create_File '.json5'
    create_File '.coffee'
    create_File '.js'
    create_File '.exe'
    create_File '.html'
    create_File '.css'
    create_File '...'  
```
Related to [#23](#23) it will be possible to do RCE on the server by editing one of the existing data coffee-scripts files (for example the one used to create random data).

Here is the code from Data-Files that creates the security issue, note how the file is updated and the code is executed:

```javascript
using new Data_Files(), ->
    # PREPARE
    new_File_Contents = 'module.exports = ()-> 40+2'
    file_Name = 'coffee-data'
    file_Path = @.find_File file_Name
    file_Contents = file_Path.file_Contents()
    @.get_File_Data(file_Name).user.assert_Is 'in coffee'
        # confirm original data

    # TEST
    @.set_File_Data file_Name, new_File_Contents
    file_Path.file_Contents().assert_Is new_File_Contents
    delete require.cache[file_Path]
    @.get_File_Data(file_Name).assert_Is '42'
        # PAYLOAD make change
        # confirm it was changed
        # clean the node cache
        # it should be 42 now

    # CLEAN
    @.set_File_Data file_Name, file_Contents
    file_Path.file_Contents().assert_Is file_Contents
    delete require.cache[file_Path]
    @.get_File_Data(file_Name).user.assert_Is 'in coffee'
        # restore file contents
        # confirm it was reset
        # clear the cache again
        # confirm original data
```

Labels:
- risk - fixed
- risk - high
- security

Milestone:
No milestone

Assignees:
No one—assign yourself

1 participant

Notifications:
You're receiving notifications because you modified the open/close state.

Lock conversation
Fix for Path transversal

This has now been fixed.

Here is the updated version of this method that doesn't have the path traversal issue

```python
set_File_Data: (filename, file_Contents) ->

    if not filename or not file_Contents # check if both values are set
        return null

    if typeof file_Contents isnt 'string' # check if file_Contents is a string
        return null

    file_Path = @.find_File filename # resolve file path based on file

    if file_Path is null or file_Path.file_Not_Exists() # check if was able to resolve it
        return null

    file_Path.file_Write file_Contents
```
Regression test

For reference here is the regression test that confirms that it is not possible to write to files outside the data folder:

```r
describe '_regression | A1 - Injection', ->

# https://github.com/DinisCruz/BSIMM-Graphs/issues/21
it 'Issue 19 - Data_Files.set_File_Data - Path Traversal', ->
  using new Data_Files(), ->
    folder_Name   = 'outside-data-root'
    file_Name     = 'some-file.txt'
    file_Content  = 'some content'
    target_Folder = @.data_Path.path_Combine('../' + folder_Name)
                   # Create target folder
    .folder_Create()
    .assert_Folder.Exists()                        # Confirm it exists
  
    target_Folder.path_Combine(file_Name)
                   # Create target file
    .file_Write(file_Content)
    .assert_File.Exists()                         # Confirm it exists

    payload   = "../{folder_Name}/#{file_Name}"
    new_Content = 'new - content'

    @.data_Path.path_Combine(payload)
    .file_Contents().assert_Is file_Content
                   # Confirm original content

    assert_Is_Null @.set_File_Data payload, new_Content
                   # PAYLOAD: Create new file

    @.data_Path.path_Combine(payload)
    .file_Contents().assert_Is file_Content
                   # Confirm original content

    target_Folder.folder_DeleteRecursive().assert_Is_True()
                   # Delete temp folder
```

OWASP
Open Web Application Security Project
WWW.OWASP.ORG
LET’S SEE HOW IT LOOKED IN THE CODE
...before the vuln is created

```javascript
#set_File_Data: fileName

list: () =>
  @.files().file_Names()

files: =>
  values = []
  for file in @.data_Path.files_Recursive()
    if file.file_Extension() in ['.json', '.json5', '.coffee']
      values.push file.remove(@.data_Path)
  values
```
...when the vuln is created

```python
56   set_File_Data: (filename, file_Contents) ->
57       if not filename or not file_Contents
58           return null
59       if typeof file_Contents isnt 'string'
60           return null
61       file_Path = @.find filename
62       if file_Path is null or file_Path.file_Not_Exists()
63           file_Path = @.data_Path.path_Combine filename
64       file_Path.file_Write file_Contents
65       return file_Path
```
set_File_Data: (filename, file_Contents) ->
    if not filename or not file_Contents
        return null
    if typeof file_Contents isnt 'string'
        return null
    file_Path = @.find filename
    if file_Path is null or file_Path.file_Not_Exists()
        file_Path = @.data_Path.path.Combine filename
    file_Path.file_Write file_Contents
    return file_Path

# todo: add security issue: that this method will allow the writing
# of any file (not just the files in the data
# folder, which are the ones that should be edited)

# todo: add security issue: filename is not validated

# todo: add security issue: directory transvesal
# todo: add security issue: no authorization, will write outside d
...after issues are created

```python
# Issue 19 - Data_Files.set_File_Data - Path Traversal
# Issue 20 - Data_Files.set_File_Data - DoS via filename and file_Contents
# Issue 23 - Data_Files.set_File_Data - allows creation of files with any extension

set_File_Data: (filename, file_Contents) ->
  if not filename or not file_Contents
    return null

  if typeof file_Contents isnt 'string'
    return null

  file_Path = @.find filename
  if file_Path is null or file_Path.file_Not_Exists()
    file_Path = @.data_Path.path_Combine filename

  file_Path.file_Write file_Contents
  return file_Path
```
...improving comments

# Issue 19 - Data_Files.set_File_Data - Path Traversal
# Issue 20 - Data_Files.set_File_Data - DoS via filename and file_Contents
# Issue 23 - Data_Files.set_File_Data - allows creation of files with any extension
set_File_Data: (filename, file_Contents) ->

if not filename or not file_Contents  # check if both values are set
  return null

if typeof file_Contents isnt 'string'  # check if file_Contents is a string
  return null

file_Path = @.find_File filename  # resolve file path based on file name

if file_Path is null or file_Path.file_Not_Exists()  # check if was able to resolve it
  return null

file_Path.file_Write file_Contents
# Issue 24 - Data.Files.set.File.Data - allows editing of coffee-script files (RCE)

```javascript
set.File.Data: (filename, fileContents) ->
    if not filename or not fileContents
        return null
    
    if typeof fileContents isnt 'string'
        return null
    
    filePath = @.find.File filename
    if filePath is null or filePath.file_Not_Exists()
        return null
    
    filePath.file_Write fileContents
```

...updating issues after 1st fix
# Issue 26 - Data_Files.set_File_Data - DoS via file_Contents

```
set_File_Data_Json: (filename, json_Data) ->

    if not filename or not json_Data                         # check if both values are set
        return null

    if typeof json_Data isnt 'string'                          # check if file_Contents is a string
        return null

    try
        JSON.parse json_Data                                  # confirm that json_Data parses OK into JSON
    catch
        return null

    file_Path = @.find_File filename                           # resolve file path based on file name

    if file_Path is null or file_Path.file_Not_Exists()      # check if was able to resolve it
        return null

    if file_Path.file_Extension() isnt '.json'                # check that the file is .json
        return null

    file_Path.file_Write json_Data                             # after all checks save file

    return file_Path.file_Contents() is json_Data             # confirm file was saved ok
```
... more issues where found later

# Issue 26 - Data_Files.set_File_Data - DoS via fileContents
# Issue 121 - Race condition on set_File_Data_Json method
# RISK-5: set_File_Data does not provide detailed information on why it failed - https://maturity-models.atl

```javascript
set_Team_Data_Json: (project, team, json_Data) ->
  if not team or not json_Data
    return null
  # check if both values are set

  if typeof json_Data isnt 'string'
    return null
  # check if json_Data is a string

  try
    JSON.parse json_Data
  catch
    return null
  # confirm that json_Data parses OK into JSON

  file_Path = @.team_Path project, team
  # resolve team path based on team name

  if file_Path is null or file_Path.file_Not_Exists() # check if was able to resolve it
    return null

  if file_Path.file_Extension() isnt '.json'
    return null
  # check that the team_Path file extension is .json
```
Thanks, any questions

@diniscruz
dinis.cruz@owasp.org