#FrontCon

JavaScript Security: A Retrospective

Research

@ Kirils Solovjovs

Security

↳ very important

Frontend is all about colours and stuff, not about security, right?

making sure the user is happy with usability etc., but what about security?

CIA triad

- Confidentiality
- Integrity
- Availability

↳ security fundamentals

all 3 aspects are important!

- C — not made available / disclosed to unauthorized people
- I — data can't be modified without authorization
- A — system available when needed

data or access loss should be prevented (for your users)

JScript 96

* 1996
  ↳ clone of JavaScript
  ↳ Netscape pushed for standardization
* 1997 — 97
  ↳ ES1
* 1998 — 98
  ↳ ES2

JavaScript 95

never 20 years old

* 1995 @ NetScape
  ↳ created in 10 days
  ↳ C-like / Java-like syntax
  ↳ BOM + DOM contained
  ↳ same-origin policy (DOM)

ECMAScript

* 1999 — 99
  ↳ ES3
  ↳ string func
  ...
* 2002 — 02
  ↳ DOM2
* 2004 — 04
  ↳ DOM3

ECMAScript 09

* 2009
  ↳ 10 years gap without new JS versions
  ↳ lots of hacker things happened
  ↳ ES 5
    ↳ "use strict"
    ↳ JSON.stringify/parse
    ↳ array methods
    ↳ func.bind()

today + future 11

* 2011 : WebSockets
* 2015 : new ECMAScript YYYY version every year — 15

attacks

- Content type misinterpretation
  (X-Content-Type-Options: nosniff)
- Clickjacking
  (X-Frame-Options: deny)
  → transparent elements to hijack mouse clicks
- cross-site scripting
  ↳ reflected
  ↳ stored
  (X-XSS-Protection: 1) — included in (Content-Security-Policy: script-src 'self')
- referrer attacks
  (referrer-policy: no-referrer)
  (referrer-policy: strict-origin)

X means non-standard header

do pentests
↳ automatically
↳ manually

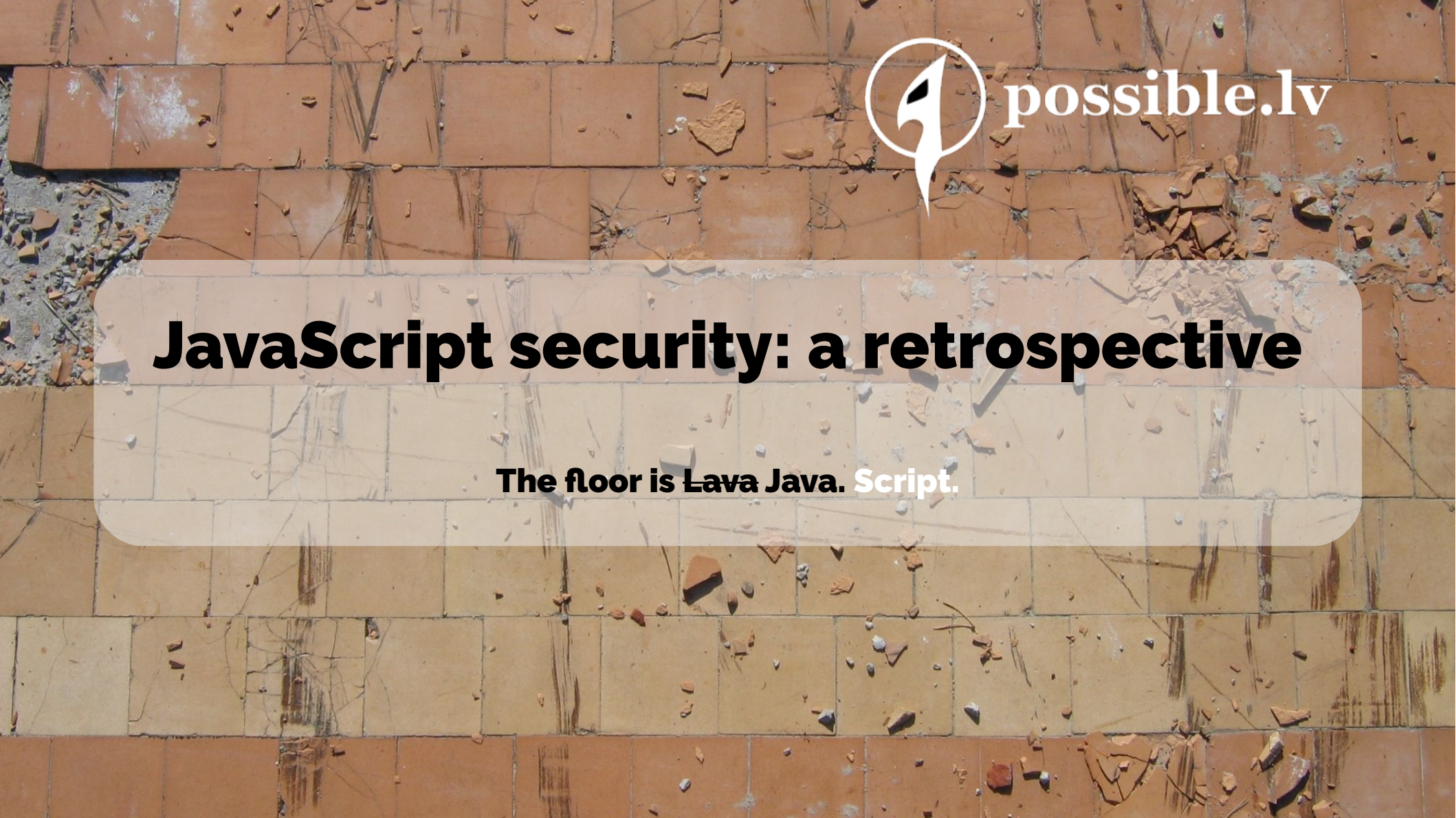
conclusions

↳ new features : increasing vulnerabilities surface
  ↳ browser exploits
  ↳ lack of explicit protection

✓ Browser manufacturers try to mitigate risk by adding additional protections

the race will continue

Katjasays.com



possible.lv

JavaScript security: a retrospective

The floor is ~~Lava~~ Java. Script.

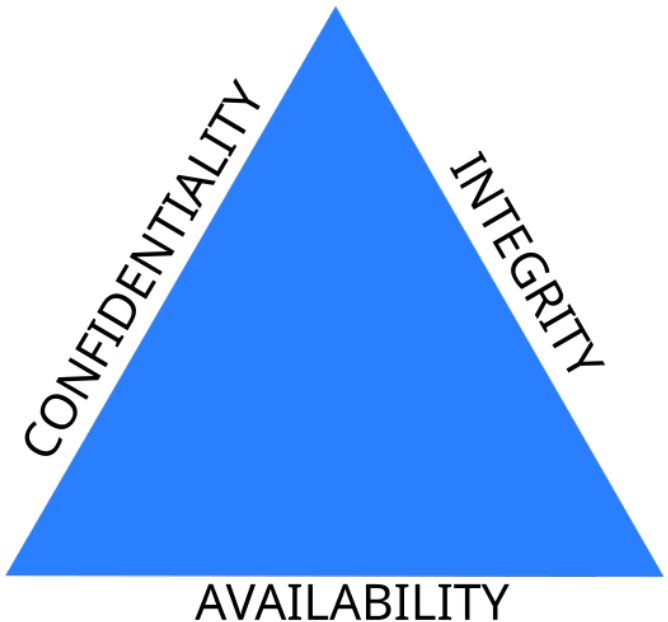
About me

- IT security expert, > 10 years
  - Mg.sc.comp, CEH, CySA+
- Owner and Lead Researcher at Possible Security
- Hacking and breaking things
  - http://kirils.org/
  - http://possiblesecurity.com/news/

Contents

- Security fundamentals
- Birth of JavaScript
- JavaScript feature set & attacks
- Conclusions

Security fundamentals – CIA triad



Security fundamentals – Confidentiality

- Confidentiality is the property, that information is not made available or disclosed to unauthorized individuals, entities, or processes.

Security fundamentals – Integrity

- Integrity means that data cannot be modified in an unauthorized or undetected manner.

Security fundamentals – Availability

- Availability is the property of the information system to be available when it is needed.

JavaScript

- Just a tad over 20 years old
- 1995 @Netscape
  - Scheme or Java?
  - scripting or static?
  - JavaScript!
    - C-like / Java-like syntax
    - Objects: BOM + DOM
    - same-origin policy (for DOM)
      - same protocol, host, and port

JS

JScript

- 1996
  - Microsoft creates a clone of JavaScript
  - Netscape pushes for standardization
    - ECMA-262 (ECMAScript)
- 1997
  - ES1 is published
- 1998
  - ES2 (formal spec changes) + DOM1

ECMAScript

- 1999
  - ES3 is born
  - string functions, regexps
  - do-white
  - try-catch
  - etc.
- 2000
  - DOM2
- 2004
  - DOM3

DOM 1 => DOM2

OMG IDL
The DOM Level 2 specifications are now using Corba 2.3.1 instead of Corba 2.2.

Type `DOMString`
The definition of `DOMString` in IDL is now a `valuetype`.

A.1.1: Changes to DOM Level 1 Core interfaces and exceptions

Interface `Attr`
The `Attr` interface has one new attribute: `ownerElement`.

Interface `Document`
The `Document` interface has five new methods: `importNode`, `createElementNS`, `createAttributeNS`, `getElementsByTagNameNS` and `getElementById`.

Interface `NamedNodeMap`
The `NamedNodeMap` interface has three new methods: `getNamedItemNS`, `setNamedItemNS`, `removeNamedItemNS`.

Interface `Node`
The `Node` interface has two new methods: `isSupported` and `hasAttributes`.
`normalize`, previously in the `Element` interface, has been moved in the `Node` interface.
The `Node` interface has three new attributes: `namespaceURI`, `prefix` and `localName`.
The `ownerDocument` attribute was specified to be `null` when the node is a `Document`. It now is also `null` when the node is a `DocumentType` which is not used with any `Document` yet.

Interface `DocumentType`
The `DocumentType` interface has three attributes: `publicId`, `systemId` and `internalSubset`.

Interface `DOMImplementation`
The `DOMImplementation` interface has two new methods: `createDocumentType` and `createDocument`.

Interface `Element`
The `Element` interface has eight new methods: `getAttributeNS`, `setAttributeNS`, `removeAttributeNS`, `getAttributeNodeNS`, `setAttributeNodeNS`, `getElementsByTagNameNS`, `hasAttribute` and `hasAttributeNS`.
The method `normalize` is now inherited from the `Node` interface where it was moved.

Exception `DOMException`
The `DOMException` has five new exception codes: `INVALID_STATE_ERR`, `SYNTAX_ERR`, `INVALID_MODIFICATION_ERR`, `NAMESPACE_ERR` and `INVALID_ACCESS_ERR`.

A.1.2: New features

A.1.2.1: New types

`DOMTimeStamp`
The `DOMTimeStamp` type was added to the Core module.

DOM2 => DOM3

Interface `Entity`
The `Entity` interface has three new attributes: `Entity.inputEncoding`, `Entity.xmlEncoding`, and `Entity.xmlVersion`.

Interface `Element`
The `Element` interface has one new attribute, `Element.schemaTypeInfo`, and three new methods: `Element.setIdAttribute(name, isId)`, `Element.setIdAttributeNS(namespaceURI, localName, isId)`, and `Element.se`

Interface `Node`
The `Node` interface has two new attributes, `Node.baseURI` and `Node.textContent`. It has nine new methods: `Node.compareDocumentPosition(other)`, `Node.isSameNode(other)`, `Node.lookupPrefix(namespaceURI)`, `Node`
`Node.setUserData(key, data, handler)`, `Node.getUserData(key)`. It introduced 6 new constants: `Node.DOCUMENT_POSITION_DISCONNECTED`, `Node.DOCUMENT_POSITION_PRECEDING`, `Node.DOCUMENT_POSITION_FOLLOWING`, `Node.DOC`
`Node.insertBefore(newChild, refChild)`, `Node.replaceChild(newChild, oldChild)` and `Node.removeChild(oldChild)` have been modified.

Interface `Text`
The `Text` interface has two new attributes, `Text.wholeText` and `Text.isElementContentWhitespace`, and one new method, `Text.replaceWholeText(content)`.

A.3 New DOM features

"XMLVersion"
The "XMLVersion" DOM feature was introduced to represent if an implementation is able to support [*XML 1.0*] or [*XML 1.1*]. See `Document.xmlVersion`.

A.4 New types

`DOMUserData`
The `DOMUserData` type was added to the Core module.

`DOMObject`
The `DOMObject` type was added to the Core module.

A.5 New interfaces

`DOMStringList`
The `DOMStringList` interface has one attribute, `DOMStringList.length`, and one method, `DOMStringList.item(index)`.

`NameList`
The `NameList` interface has one attribute, `NameList.length`, and two methods, `NameList.getName(index)` and `NameList.getNamespaceURI(index)`.

`DOMImplementationList`
The `DOMImplementationList` interface has one attribute, `DOMImplementationList.length`, and one method, `DOMImplementationList.item(index)`.

`DOMImplementationSource`
The `DOMImplementationSource` interface has two methods, `DOMImplementationSource.getDOMImplementation(features)`, and `DOMImplementationSource.getDOMImplementationList(features)`.

`TypeInfo`
The `TypeInfo` interface has two attributes, `TypeInfo.typeName`, and `TypeInfo.typeNamespace`.

`UserDataHandler`
The `UserDataHandler` interface has one method, `UserDataHandler.handle(operation, key, data, src, dst)`, and four constants: `UserDataHandler.NODE_CLONED`, `UserDataHandler.NODE_IMPORTED`, `UserDataHandler.NOD`

`DOMError`
The `DOMError` interface has six attributes: `DOMError.severity`, `DOMError.message`, `DOMError.type`, `DOMError.relatedException`, `DOMError.relatedData`, and `DOMError.location`. It has four constants: `DOMError.SEVERITY`

`DOMErrorHandler`
The `DOMErrorHandler` interface has one method: `DOMErrorHandler.handleError(error)`.

ECMAScript

- Fast forward ten years 1999 => 2009
- ES 5
  - “use strict”
  - JSON.stringify() / JSON.parse()
  - array methods
    - .indexOf(), .map(), etc.
  - func.bind()

Today + future

- 2011 – WebSockets
- 2015...
  - new ECMAScript YYYY version every year

Attacks

Content type misinterpretation

- Allows forcing browser (MSIE) to misinterpret the content type

  [2008, IE only]
- X-Content-Type-Options: nosniff

Clickjacking

- Using transparent elements to hijack mouse clicks

  [2010, RFC in 2013]
- X-Frame-Options: deny
  - prevents content to be loaded as a frame source

Cross-site scripting

- Reflected
  - hxxp://site.com/file.php?data=hello<script>alert(1);</script>
- Stored
  - STORE → hxxp://site.com/store.php?data=hello<script>alert(1);</script>
  - RETRIEVE ← hxxp://site.com/read.php

Solution – X-XSS-Protection

[2010, IE only at first]

- X-XSS-Protection: 1
  - built-in blacklist filter
  - NOT A FULL PROTECTION

Solution – Content-Security-Policy

[2015]

- Content-Security-Policy: script-src 'self'
- Defines where can different resources be loaded from. Disables inline JavaScript.
- X-XSS-Protection now a part of CSP
- QUITE EFFECTIVE

Referrer attacks

- Could be used for tracking, locating private and local systems,

  [2017]
- Referrer-Policy: no-referrer
- Referrer-Policy: strict-origin
- Defines what kind of referrer information to send in what cases.

Conclusions

- New features in ECMAScript + DOM Levels provide for ever increasing vulnerability surface
  - due to browser exploits (implementation bugs)
  - due to lack of explicit protection
- Browser manufacturers try to mitigate this increased risk by adding additional protections
- The race will continue!

Thank you!

JavaScript security: a retrospective

The floor is ~~Lava~~ Java. Script.

possible.lv