THE CUSTOMER SUCCESS PLATFORM
SALES SERVICE MARKETING COMMUNITY ANALYTICS APPS

Protect your data against malicious scripts

Xiaoran Wang
Senior Product Security Engineer
@0x1a0ran

dreamforce

Safe Harbor

Safe harbor statement under the Private Securities Litigation Reform Act of 1995:

This presentation may contain forward-looking statements that involve risks, uncertainties, and assumptions. If any such uncertainties materialize or if any of the assumptions proves incorrect, the results of salesforce.com, inc. could differ materially from the results expressed or implied by the forward-looking statements we make. All statements other than statements of historical fact could be deemed forward-looking, including any projections of product or service availability, subscriber growth, earnings, revenues, or other financial items and any statements regarding strategies or plans of management for future operations, statements of belief, any statements concerning new, planned, or upgraded services or technology developments and customer contracts or use of our services.

The risks and uncertainties referred to above include – but are not limited to – risks associated with developing and delivering new functionality for our service, new products and services, our new business model, our past operating losses, possible fluctuations in our operating results and rate of growth, interruptions or delays in our Web hosting, breach of our security measures, the outcome of any litigation, risks associated with completed and any possible mergers and acquisitions, the immature market in which we operate, our relatively limited operating history, our ability to expand, retain, and motivate our employees and manage our growth, new releases of our service and successful customer deployment, our limited history reselling non-salesforce.com products, and utilization and selling to larger enterprise customers. Further information on potential factors that could affect the financial results of salesforce.com, inc. is included in our annual report on Form 10-K for the most recent fiscal year and in our quarterly report on Form 10-Q for the most recent fiscal quarter. These documents and others containing important disclosures are available on the SEC Filings section of the Investor Information section of our Web site.

Any unreleased services or features referenced in this or other presentations, press releases or public statements are not currently available and may not be delivered on time or at all. Customers who purchase our services should make the purchase decisions based upon features that are currently available. Salesforce.com, inc. assumes no obligation and does not intend to update these forward-looking statements.

Agenda

- Background
- HTML Parsing
- URL Parsing
- JavaScript Parsing
- Parsing Order
- Salesforce protections

Background

Background

XSS 101

- <html> Hello <%=userName> ! </html>
  - When username is “Cookie Monster <script>stealMyCookie();</script>”
  - <html>Hello Cookie Monster ! <script>stealMyCookie();</script> </html>

Background

Introduction

- Browser Parsings are complex
  - Several Parsers
  - Rounds of decodings
- How do we know whether a particularly encoded script is going to execute in certain context?

Background

Will this execute?

Charater entity encoded “javascript” and URL encoded “alert(2)”

Background

Will this execute?

<script>\u0061\u006c\u0065\u0072\u0074(10);</script>

Unicode Escape Sequence encoded alert

<script>alert(10);</script>

Background

Will this execute?

<button onclick="confirm('7');">Button</button>

Character Entity Encoded ‘ (single quote)

<button onclick=“confirm(‘7’);”>Button</button>

Background

Will this execute?

<a
href="javascript:%
5c%75%30%30%3
6%31%5c%75%30
%30%36%63%5c%
75%30%30%36%3
5%5c%75%30%30
%37%32%5c%75%
30%30%37%34(1
5)">

????

HTML Parsing

HTML Parsing

Basics

- HTML parser works as a state machine
- Recognize valid tags as tokens
- “<“ make it go into Tag Open State
- We don’t want arbitrary user data be treated as a token
- Character entity encoding is here to help
  - ‘<‘ → ‘<’
  - ‘>’ → ‘>’
  - etc.

HTML Parsing

States

- Only three parsing states can contain character entities
  - Data state
    - <img src=x onerror=alert(123)>
  - Attribute value state
    -
  - RCDATA state
    - <title><img src=x onerror=alert(123)></title>
- Any '<' decoded from '<' in these states will NOT be treated as token

HTML Parsing

HTML Elements

- Void elements
  - <area>, <base>,
, <img>, <input>, <keygen>, <link>, …
  - Cannot have any contents
- Raw text elements
  - <script>, <style>
  - Text only
- RCDATA elements
  - <textarea>, <title>
  - Text and character references
- Foreign elements
- Normal elements

HTML Parsing

RCDATA state

- When the parser is in <textarea> or <title>
- Character entities will be decoded
  - <textarea><img src=x onerror=alert(123)></textarea>
- Even open tags will NOT be trigger the “Open Tag State”
  - <textarea><img src=x onerror=alert(123)></textarea>

URL Parsing

URL Parsing

Basics

- URL parser works as a state machine
- Decoding using UTF-8
- URL scheme has to be in ASCII characters
  - U+0041-U+005A
  - U+0061-U+007A

URL Parsing

Would it execute?

URL encoded “javascript”

URL Parsing

Would it execute? NO

URL encoded “javascript”

The scheme must not be encoded

JavaScript Parsing

JavaScript Parsing

Basics

- Context free language
  - Context free grammar for parsing
- Error Intolerant
- Located in <script> tags or inline
  - <script> is a RAW element thus not having HTML decoding

JavaScript Parsing

Would it execute?

<script>aler&#116(9)&#59</script>

Character entity encoded alert(9);

<script>alert(9);</script>

JavaScript Parsing

Would it execute?

<script>aler&#116(9)&#59</script>

Character entity encoded alert(9);

<script>alert(9);</script>

NO

JavaScript Parsing

Would it execute?

Character encoded 'j'

JavaScript Parsing

Would it execute?

Character encoded 'j'

YES

JavaScript Parsing

Unicode Escape Sequence

- \uXXXX
- If JavaScript is encoded using \uXXXX, would it execute?
  - It depends on where the encoded content is located
    - In strings
    - In identifier names
    - In control characters

JavaScript Parsing

Strings

- alert(‘my string’);
- String literals are surrounded by single or double quotes
- Unicode escape sequence will not break out of the string context
  - alert(‘my string\u0027’); (\u0027 is single quote)
- ECMA-262 edition5.1 Rev 6, Clause 6
  - *“a Unicode escape sequence occurring within a string literal in an ECMAScript program always contributes a Unicode character to the literal and is never interpreted as a line terminator or as a quote mark that might terminate the string literal.”*

JavaScript Parsing

Identifier names

- alert(‘my string’);
- Unicode escape sequence will be decoded in identifier names
  - \u0061lert(‘my string’); (\u0061 is ‘a’)
- ECMA-262 edition5.1 Rev 6, Clause 6
  - *"Unicode escape sequences are also permitted in an IdentifierName, where they contribute a single character to the IdentifierName, as computed by the CV of the UnicodeEscapeSequence (see 7.8.4). The \ preceding the UnicodeEscapeSequence does not contribute a character to the IdentifierName. A UnicodeEscapeSequence cannot be used to put a character into an IdentifierName that would otherwise be illegal."*

JavaScript Parsing

Control characters

- alert(‘my string’);
- Unicode escape sequence will not be decoded as control characters
  - alert\u0028‘my string’); (\u0028 is ‘(’)

JavaScript Parsing

Would it execute?

<script>\u0061\u006c\u0065\u0072\u0074(\u0031\u0032)</script>

Unicode escape sequence encoded ‘alert(12)’

<script>alert(12)</script>

JavaScript Parsing

Would it execute? NO

<script>\u0061\u006c\u0065\u0072\u0074(\u0031\u0032)</script>

Unicode escape sequence encoded ‘alert(12)’

<script>alert(12)</script>

\u0031\u0032 is not part of an identifier name or in a string literal

Parsing Order

Parsing Order

Basics

- All the parsers work together in an order
- The order depends on the location of the content to parse
- Each round of parsing will decode the content once
- The content needs to be encoded in the reverse order

Parsing Order

Example: Two rounds of encoding

-
  - HTML parser first decode the whole document (UserInput is HTML entity decoded)
  - HTML parser sees the href attributes
  - HTML parser ask URL parser to decode the URL
  - URL parser decode the href (UserInput is URL decoded)
- In order to make the UserInput safe from XSS
  - URL encode the UserInput first
  - Then HTML entity encode the UserInput
  -

Parsing Order

Example: Three rounds of encoding

-
  - HTML parser first decode the whole document (UserInput is HTML entity decoded)
  - HTML parser sees the onclick attributes
  - HTML parser asks the JavaScript parser to handle the JavaScript
  - JavaScript parser decode the string in onclick (UserInput is JavaScript decoded)
  - JavaScript parser sees the window.open function, expecting a URL in the string
  - JavaScript parser ask URL parser to decode the URL
  - URL parser decode the string (UserInput is URL decoded)
- In order to make the UserInput safe from XSS
  -

Parsing Order

Example: more rounds of encoding

- <a href="javascript:window.open('/foo?name=UserInput')">
- Advice: Don’t do this… because you have to encode it four times
  -

Salesforce Protections

Salesforce Protections

Visualforce

- All {!MergeFields} are html encoded automatically
- Encoding functions available for complex contexts
  - HTMLENCODE
  - JSENCODE
  - URLENCODE
  - JSINHTMLENCODE
- Use them in the correct order to properly escape your user input

Salesforce Protections

Visualforce – Encoding Functions

- HTMLENCODE
  - Encode characters in HTML context
  - <apex:outputText value="{!HTMLENCODE(userInput)}" escape="false"/>
  - This is anti-pattern, use escape automatically
  - <apex:outputText value="{!userInput}" />

Salesforce Protections

Visualforce – Encoding Functions

- JSENCODE
  - Encode characters in JavaScript string literal context
  - <script>var string = “{!JSENCODE(userInput)}”;</script>

Salesforce Protections

Visualforce – Encoding Functions

- URLENCODE
  - Encode characters in URL context
  -
  - Developers need to make sure userInput has a expected scheme
    - userInput.startsWith('https')
    - Make sure it is not a javascript: scheme

Salesforce Protections

Visualforce – Encoding Functions

- JSINHTMLENCODE
  - Encode characters in HTML tags in JavaScript
  - node.innerHTML = “<div id=‘” + {!JSINHTMLENCODE(userInput)} + “’>My Div ”;

Salesforce Protections

Visualforce – Encoding Example

-
  - UNSAFE
-
  - SAFE

Other Salesforce security controls

- Data protections
  - CRUD
  - FLS
  - Sharing
  - Encrypted custom fields
  - Etc.
- Operation protections
  - Login IP Range
  - Login Hours
  - Two Factor Authentications
  - Audit Trails
  - Etc.

Questions?

Secure Development Sessions

Secure Coding: Field-level Security, CRUD, and Sharing
Monday, October 13 @ 11:00 a.m. - 11:40 a.m.

Secure Coding: Storing Secrets in Your Salesforce Instance
Monday, October 13 @ 2:00 p.m. - 2:40 p.m.

Building Secure Mobile Apps
Monday, October 13 @ 5:00 p.m. - 5:40 p.m.

Protect Your Data Against Malicious Scripts
Tuesday, October 14 @ 11:00 a.m. - 11:40 a.m.

Secure Coding: External App Integration
Wednesday, October 15 @ 9:00 a.m. - 9:40 a.m.

Secure Coding: SSL, SOAP, and REST
Thursday, October 16 @ 10:30 a.m. - 11:10 a.m.

dreamforce

Announcements:

Force.com Code Scanner now supports Salesforce1 and JavaScript! Try it here: http://bit.ly/SF1Scanner

Chimera Web App Scanner alpha nominations are open. Partners apply at: http://bit.ly/SFChimera

Live security office hours are available in the partner zone.

salesforce

Thank You