2013 Ruby on Rails Exploits

CS 558
Allan Wirth

Background: Ruby on Rails

- Ruby: Dynamic general purpose scripting language similar to Python
- Ruby on Rails: Popular Web app framework using Ruby
  - Designed for rapid and easy development
  - Used by sites such as Twitter, Github, Basecamp, Scribd, and Groupon
  - Has become very popular in the past few years. About 2% of the top 10,000 websites use Ruby on Rails.
  - Comes with every Mac OS X computer since 10.7 (Mountain Lion)

Background: XML

- XML (Extensible Markup Language): Very popular human-readable data serialization format.
  - Used to transfer data from one program to another in a platform agnostic fashion.
  - Core markup language for many formats, such as:
    - XHTML – Used for Web Pages
    - Open Office XML – Document Format for Microsoft Office 2007+
    - XMPP – Google Chat, Facebook Chat, Jabber
    - Nearly all web API services support XML data (such as the Twitter, Facebook, Tumblr and Reddit APIs)
  - Example:

```
<message>Hello World!</message>
```

Background: JSON / YAML

- JSON: JavaScript Object Notation
  - Designed to be a simplistic human-readable data serialization using familiar JavaScript syntax
  - Example:

```
{
    "first_name": "Allan",
    "last_name": "Wirth",
    "major": "Computer Science"
}
```

- YAML: YAML Ain't Markup Language
  - Designed to be a human readable data serialization language.
  - JSON is supported as a strict subset of YAML.

Exploit: Ruby YAML implementation

- Under normal conditions, the only types returned by YAML parsing are generic types such as integers, lists and dictionaries.
- The Ruby YAML implementation allows encoding of arbitrary object formats. When deserialized, the object will be constructed with the provided parameters.
  - Example:

```
--- !ruby/array:Array
  - apples
  - pears
```

  - Useful for automagic serialization of user defined types
  - Not inherently dangerous, as long as run on trusted input. If an attacker's input is interpreted as YAML, they can instantiate arbitrary objects by name. For example, an attacker could instantiate a class representing an SQL statement to run custom queries against the database.

Exploit: Rails Deserialization

- Rails will automagically deserialize incoming requests that contain JSON or XML parameters. It does not do this for YAML, because it would be unsafe.
- However, the XML parser interprets elements in the form `<foo type="yaml">...</foo>` as YAML documents. This can be used by an attacker to have the YAML interpreter interpret malicious YAML code.
- Furthermore, the JSON parser simply uses the YAML parser as the backend. This can be used to get the YAML interpreter to interpret client-defined YAML documents.

Exploit: Rails Code Execution

- By using one of the aforementioned methods, an attacker can obtain YAML processing and instantiate arbitrary objects on the server.
- By finding an object that will execute code on instantiation, the attacker can run whatever they want.
- The class `ActionDispatch::Routing::RouteSet::NamedRouteCollection` allows for this sort of execution.
- Example:

```
--- !ruby/hash:ActionDispatch::Routing::RouteSet::NamedRouteCollection
 "foo; eval(puts '=== hello there'.inspect);": !ruby/object:OpenStruct
   table:
    :defaults: {}
```

Impact

- Both exploits allowed arbitrary code execution on any website running any Rails release for the past 6 years. The exploit had 100% reliability.
- XML vulnerability released Tuesday, January 8th, 2013
- JSON vulnerability released Monday, January 28th, 2013
- Both vulnerabilities were patched within the same day as release.
- A few large sites that were not patched quickly enough were attacked, such as rubygems.org and Bitcoin exchange Vircurex where an undisclosed sum of Bitcoins were stolen.
- Both vulnerabilities were rolled into Metasploit.

Questions?

References

- Ruby on Rails: http://rubyonrails.org/
- Ruby: http://www.ruby-lang.org/en/
- JSON Parser Vuln Release: https://groups.google.com/forum/?fromgroups=#!topic/rubyonrails-security/1h2DR63ViGo
- CVE-2013-0333 write-up: http://micrmsoft.blogspot.com/2013/01/cve-2013-0333-ruby-on-rails-json-parser.html
- Anatomy of an Exploit: http://rubysource.com/anatomy-of-an-exploit-an-in-depth-look-at-the-rails-yaml-vulnerability/
- XML Vuln Release: http://www.kb.cert.org/vuls/id/380039
- Ruby Exploit Recap: http://trends.builtwith.com/framework/Ruby-on-Rails