TECHWORLD

The security nightmare of REST web services

The web protocol is winning the battle of the APIs

By Chris Comerford and Pete Soderling | CSO | Published: 15:10 GMT, 25 February 10

Clearly, REST (Representational State Transfer) is winning the web service protocol debate. What was once a grassroots movement has proliferated inside the enterprise and out. As of January 2010, 1,100 out of the 1,600 APIs listed on Programmable Web are REST-based. Some of our best-known cloud services utilize REST, including APIs from Amazon, SalesForce and Google.

As API engineers, we're all for it. The architectural style described by REST is simpler, more open, more scalable and more consistent with other Internet protocols (remember that REST is basically HTTP) than the alternative, the litany of web service protocols referred to as WS-*.

But as security experts, we're aghast at some of the REST-based API practices we're seeing. In general, the security of APIs is much lower than in web applications. There are two key reasons for this:

1) REST does not have predefined security methods so developers define their own, and

2) Often, developers in a hurry to just get their web services deployed don't treat them with the same level of diligence as they treat web applications.

These conditions lead to web services with serious vulnerabilities. For instance, most APIs handle authentication using a key but no secret, essentially requiring a user name but no password. Another problem is using HTTP basic authentication (with no SSL) and letting the user name and password cross the wire with no encryption.

REST APIs typically have the same attack vectors as standard web applications, including injection attacks, cross-site scripting (XSS), broken authentication and cross-site request forgery (CSRF).

They also have some unique vulnerabilities and attack vectors, including mashup related issues in which the mashup requires end users to place too much trust in the mashup provider. For example, a mashup that pulls data from multiple APIs might require user names and passwords. It's up to the mashup provider to authenticate end users' access credentials. End users must trust the mashup provider not to steal (or inadvertently reveal) their credentials, and the API providers must trust that the mashup provider has authenticated the valid user of this account, not a hacker or malicious user. Other vulnerabilities stem from immature grassroots protocols such as OAuth 1.0, which is vulnerable to a session fixation attack and could result in an attacker stealing the identity of an API enduser.

So, what can you do to secure your RESTful APIs? There are many rules to follow, but we'll call out a few:

Do employ the same security mechanisms for your APIs as any web application your organisation deploys. For example, if you are filtering for XSS on the web frontend, you must do it for your APIs, preferably with the same tools.
Don't roll your own security. Use a framework or existing library that has been peer reviewed and tested. Developers not familiar with designing secure systems often produce flawed security implementations when they try to do it

themselves, and they leave their APIs vulnerable to attack.
Unless your API is a free, read-only public API, don't use single key-based authentication. It's not enough. Add a password requirement.
Don't pass unencrypted static keys. If you're using HTTP Basic and sending it across the wire, encrypt it.
Ideally, use hash-based message authentication code (HMAC) because it's the most secure. (Use SHA-2 and up, avoid SHA & MD5 because of vulnerabilities.)

We'll be getting into greater technical depth on REST web service security issues and methods at the RSA Conference 2010, session ID AND-203. Hope to see you there!

---

http://features.techworld.com/security/3213655/the-security-nightmare-of-rest-web-services/