Lecture Notes:
Towards a Formal Foundation of Web Security

Chris Griggs

April 6th

Introduction

The fast paced evolution of the internet brings a great deal of new features, however, these features may introduce new vulnerabilities as the specifications for the features are complex, they make assumptions about the interactions with the rest of the internet that may be false.

Central concepts being formalized include browsers, servers, scripts, HTTP, and DNS as well as the way they interact

Three distinct threat models

1. Web attacker
    a. Operates a malicious web site
2. Active network attacker
    a. Ability to eavesdrop, block, and forge network messages + web attacker + use browser APIs
3. Gadget attacker
    a. Able to inject limited content into an honest website

Security Goals

1. New mechanisms should not violate any of the invariants that web site commonly rely on for security
2. “Session integrity”, an attacker should not be able to cause an honest server to undertake potentially sensitive actions

General Model

Primary idea is to focus on how an attacker can abuse web functionality that exists by design not such things as phishing or drive by downloads

Web Concepts

The central concepts that are common to every web security mechanism

1. Non-linear time - how to model time
    a. They don't delve into branching choices i.e. if a user has an option to click two links their model assumes both are clicked no matter what
    b. The only temporal order they have is for events that have to occur in sequence i.e. an HTTP request before an HTTP response
2. Browser - how abstract do they make the browser
    a. Script Context
        i. Represents all the scripts running in the browser on behalf of a single web origin
        ii. No isolation between scripts
    b. Security UI
        i. Browser guarantees some security properties for elements i.e. the location bar accurately displays the URL
    c. State Storage
        i. Assume the browser has some persistent storage (cookie store or pw database) and that this information can be read script context on behalf on that origin
        ii. Assume state is append only
3. Servers
    a. Each web server is owned by a single principle that controls how that server responds to network messages
    b. Many-to-many relation to DNS names essential for DNS Rebinding
        i. In this attack, a malicious web page causes visitors to run a client-side script that attacks machines elsewhere on the network.
4. Network
    a. Model the communication between servers and browsers

Threat Models

1. Web Attacker
    a. Web Server
        i. Controls at least one web server and how that servers responds to HTTP requests
        ii. Can purchase domains from trusted CA to host malicious content at something like attacker.com/. From their main site of attacker.com
    b. Network
        i. Can only respond to requests
        ii. Can send requests to honest servers without regard for http protocol

    c. Browser
        i. Once the user visits the attacker's website the attacker can use any browser APIs that are available to other websites
2. Network Attacker
    a. All the abilities of a web attacker
    b. Ability to read, control, and block the contents of all unencrypted network
        i. Cannot corrupt HTTPS traffic between honest principals because CA won't issue them a certificate for an honest site
3. Gadget Attacker
    a. All the abilities of the network attacker
    b. The ability to to inject limited content into honest web sites
        i. What content depends on the web application
4. User Behavior
    a. The user may visit any website
    b. The user does not confuse the attacker's website with an honest website
        i. Rules out phishing attacks

Security Goals

1. Security Invariants
    a. There are a large number of assumptions made by existing web applications about web security
        i. Only deal with those relating to mechanisms at hand
            1. I'm not sure how they identified these unless it was experience
2. Session Integrity
    a. The desire to make sure a request was generated by a trusted principal not an attacker

Case Studies

1. Origin Header
    a. Propose that browser identify the origin of HTTP requests by including an origin header and websites use this to defend against cross-site request forgery
    b. Vulnerability
        i. If an honest server sends a request to an attacker's server, then the attacker can redirect the request back to the honest server. The

            honest server will accept the request which violates session integrity

   c. Solution
      i. Naming all of the origins involved in in the redirect chain
2. Cross-Origin Resource Sharing
   a. Let's web sites opt out of some of the browsers security protections. In particular shore contents of a response with a particular origin, or let an origin request otherwise forbidden task.
      i. Requires a complex request that requires pre-flight request that asks the servers permision
   b. Vulnerability
      i. A legacy server might redirect the pre-flight request to the attacker's server
   c. Solution
      i. Ignore all pre-flight requests
3. Referer Validation
   a. To defend against Cross-site request forgery and cross site scripting the web site should reject the request unless the referer header is from the site's origin or the request is directed at a gateway page that is vetted for these attacks
   b. Vulnerability
      i. If the attacker is able to insert a hyperlink on the honest web page that links to their server they can redirect that request back to the honest server
   c. Solution
      i. Suppress all outgoing referer headers
4. HTML5 forms
   a. Adds functionality to the FormElement API to generate API requests with PUT and Delete methods. Requires the new methods to be sent to the same origin as the form
   b. Vulnerability
      i. An attacker can generate a PUT request to their own website and then redirect that request to an honest webserver, causing that server to recieve an unexpected PUT request
         1. Not sure what this does
   c. Solution
      i. Refuse to follow redirects of PUT or DELETE requests generated by HTML forms
5. WebAuth

a. Vulnerability
    i. Can send a link that will login the attacker on the user’s system without revealing the attacker’s credentials
        1. Not sure why this matters