selectively disabling JavaScript, and removing form length limits, as well as provides a number of other tools useful for security testing and web application development.

Firebug

- Authors: Joe Hewitt and Rob Campbell
- Location: https://addons.mozilla.org/en-US/firefox/addon/1843

Firebug allows for the inspection, manipulation, and inline editing of the source of a rendered page. This can be used to remove elements, to delete or change blocks of JavaScript, and to just get a feel for the application's structure. Another add-on that usefully extends the functionality of Firebug is "Firecookie," which allows for cookie viewing and editing for the site you're inspecting (see Figure B-9).

Networking Tools

Similar to web or client/server application testing, mobile application security also benefits from networking tools. The following is a list of the network tools that may assist in a mobile security test.

Wireshark

- Authors: Gerald Combs et. al.
- Location: www.wireshark.org

Wireshark is a packet capture and analysis tool widely used in the security, network, and systems administration industries and the software development

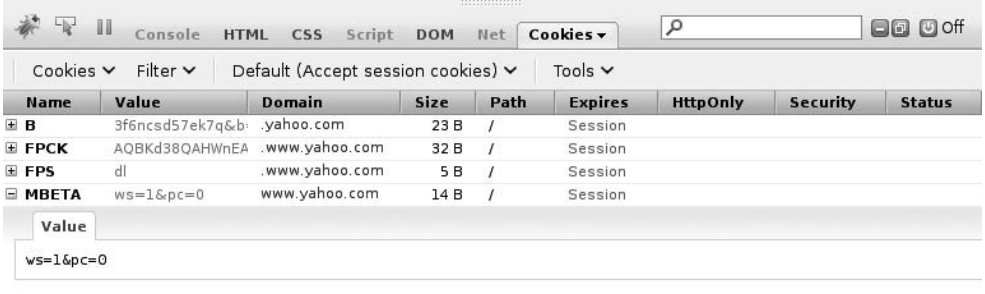

Figure B-9 *Editing cookies within Firecookie*

industry. It can either capture packets live or analyze those stored in standard formats, such as libpcap output files. It analyzes packets from OSI layers 2–7, giving information on Ethernet frames, IP packets, and higher-level protocols such as HTTP. In a mobile development context, tools such as Wireshark are useful for ensuring that clear-text data is not being sent over the network, as well as for debugging when networking code doesn't behave as you expect.

Wireshark knows how to parse a great many protocols, including HTTP, various chat services (such as AIM and XMPP), DNS traffic, and voice data (such as that sent over SIP/RTP), displaying detailed information in a tree-structured interface. It is a valuable tool for anyone working with computers to be familiar with—people attacking your software most certainly will be! Figure B-10 shows an example of the types of traffic captured by Wireshark.

Tcpdump

- Location: www.tcpdump.org

An old standard for network packet analysis, tcpdump allows for the capture of network packets using user-defined filters, only capturing traffic matching specific patterns. Whereas tools such as Wireshark give more vivid insight into the content of network packets, tcpdump is lighter weight, available on many systems, and suitable for performing packet capture on systems where a graphical environment isn't easily available (such as Unix servers). Sometimes, rather than running a tool such as Wireshark on your local network, running tcpdump on the server that you're trying to test against can be an easier way to capture data, without having to deal with any messiness such as ARP spoofing (http://en.wikipedia.org/wiki/ARP_spoofing). Simply run the tool on the server as follows (you will need root access to do this):

```
tcpdump  s0  ni  eth0  w  mycapture.pcap  tcp  and  port  80
tcpdump:  listening  on  ath0,  link type  EN10MB  (Ethernet)
capture  size  65535  bytes
^C136  packets  captured

240  packets  received  by  filter

0  packets  dropped  by  kernel
```

This will capture all packets sent and received by the eth0 interface on TCP port 80 (HTTP), listening until interrupted with CTRL-C. Output will be stored in "pcap" format in mycapture.pcap. When finished, you can copy the resulting pcap file to a local desktop system for further analysis or filtering with Wireshark or another parsing tool.

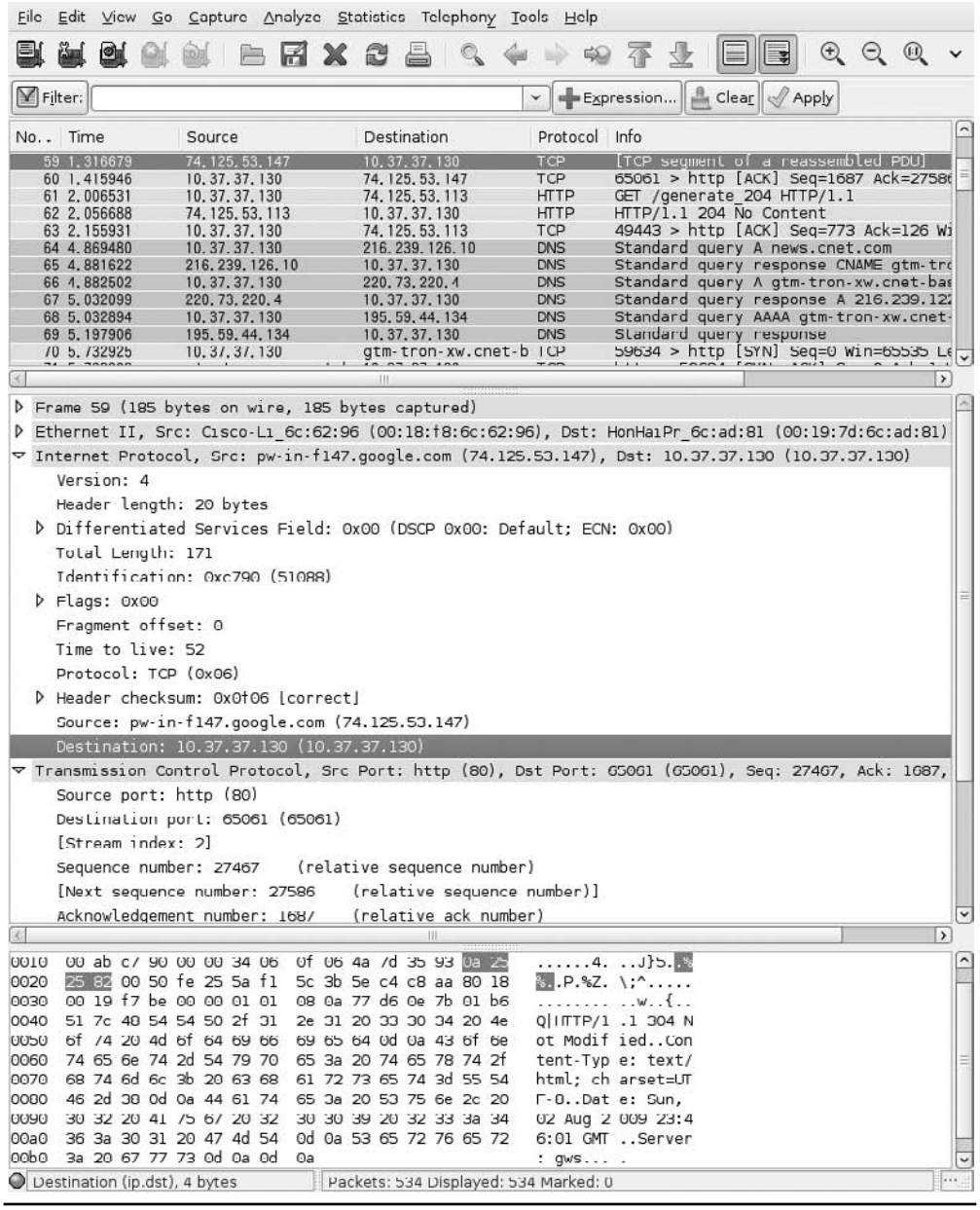

Figure B-10 *Wireshark analyzing live network traffic*

See the main page for tcpdump for more information on its command-line options and filter expressions (http://www.tcpdump.org/tcpdump_man.html).

Scapy

- Author: Philippe Biondi
- Location: http://www.secdev.org/projects/scapy/

Scapy also performs packet capture and analysis, but it can also actively generate traffic, encapsulated and transformed in many ways. It is something of a Swiss army knife of packet manipulation, and can be useful for writing tools to send specifically crafted packets or to watch for specific traffic patterns, responding with particular packet transmissions. Knowledge of Python is necessary to work with Scapy; however, Python is a relatively easy language to understand for most experienced developers.

Web Application Tools

The following tools can help test mobile HTML sites by allowing the developer to modify content after it leaves a browser or local application, but before it is sent to a remote server. This approach is very commonly used in penetration testing and QA, and can be quite convenient to a developer as well.

WebScarab

- Author: Rogan Dawes
- Location: www.owasp.org/index.php/Category:OWASP_WebScarab_Project

WebScarab is a free open-source network proxy maintained by OWASP. It performs interception of HTTP traffic, allowing for changing it in transit, replaying it in different ways, fuzzing, and more. This is useful to mobile developers for testing the results of changing traffic in-flight or for simply seeing what HTTP requests a given application makes, along with the content of server replies. Monitoring and altering traffic at the network level can be far more convenient than changing your code or using debugging output. Additionally, for mobile HTML sites, you can use this approach to perform attacks on the server as well—for instance, inserting malicious script into various parameters and removing validation tokens in transit. Several other proxy tools also

perform similar functions to WebScarab, all with their different strengths and weaknesses. A few of these are Burp, gizmo-proxy, and Paros.

To use WebScarab, simply run it on a desktop machine, configuring it to listen on a network interface accessible to your mobile device, rather than the default of 127.0.0.1. Then, configure your mobile device's proxy settings to use the IP of your desktop machine, port 8008. By default, WebScarab only gathers information—by using the Proxy | Manual Edit | Intercept Requests option, you can edit requests before their transmission on the network (see Figure B-11).

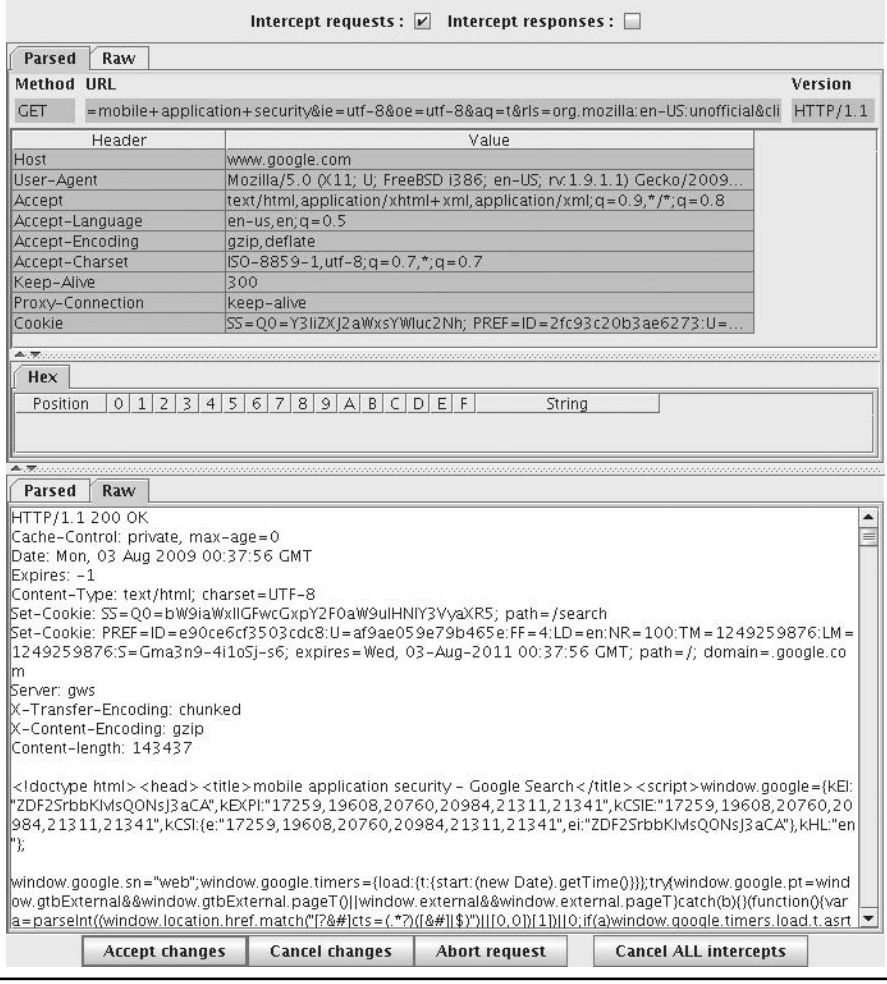

Figure B-11 *WebScarab's traffic interception mode*

One caveat with using such tools is that because a primary function of SSL is to prevent middle-person attacks, an error or warning will (or at least, should) be thrown whenever a client tries to access an SSL-enabled URL through the proxy. To solve this, one option is to create your own SSL Certificate Authority, create a new SSL certificate for the HTTPS server that you want to impersonate, and install the Certification Authority (CA) certificate on the mobile device. Note that on the iPhone, new certificate installation can only be done on the mobile device itself, not the emulator.

In the event that you need to do this for multiple servers, you can automate the individual certificate-signing process using CyberVillainsCA (www.isecpartners.com/cybervillainsca.html).

This will dynamically create a new certificate, signed by your Certificate Authority, for every site you visit.

Gizmo

- Author: Rachel Engel
- Location: code.google.com/p/gizmo-proxy/

If WebScarab is a bit heavy for your taste, Gizmo strips down and simplifies the concept. Gizmo includes CyberVillainsCA—it will generate a unique CA certificate upon first use, which you can then import into your browser or other certificate store. Requests are navigated with basic vi editor keybindings: j/k to move up and down in the request list. “e” edits a request, and “s” sends.

One of the useful features Gizmo offers is the ability to send requests directly to commands, or even to your favorite text editor, before sending the request along. This can be done by specifying your default shell and commands to be executed—the request itself will be loaded into a file referred to by the environment variable “BUF”, which can then be sent to another process. The text output of this process (“stdout”) will be returned into the bottom frame. For example, if using a Unix machine, one can enter the following as the parsing commands:

```
grep  v Cookie: $BUF
```

to remove the cookie header from the request.