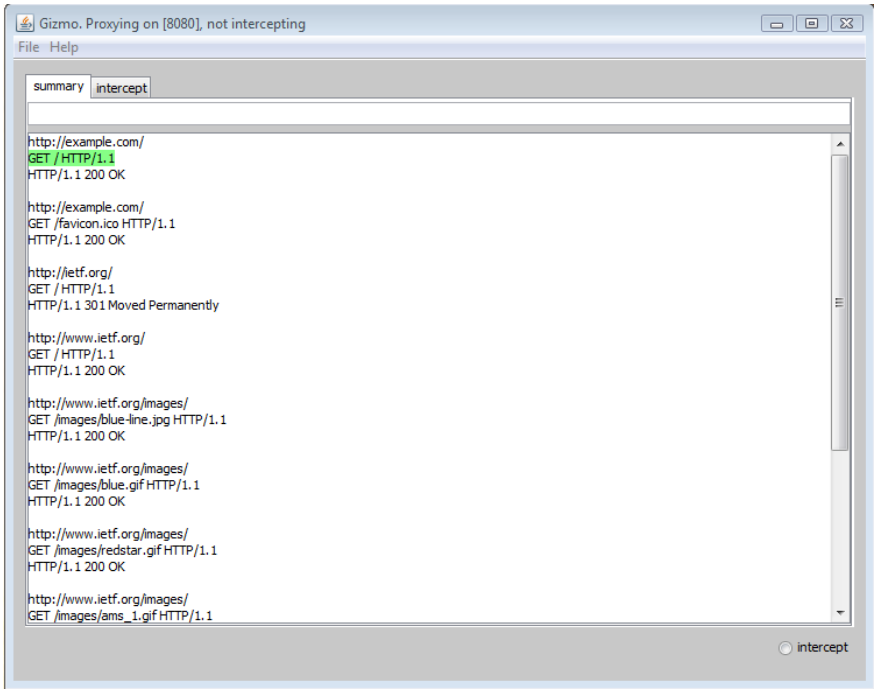
Gizmo, a lightweight open source web proxy

Rachel Engel : June, 2009

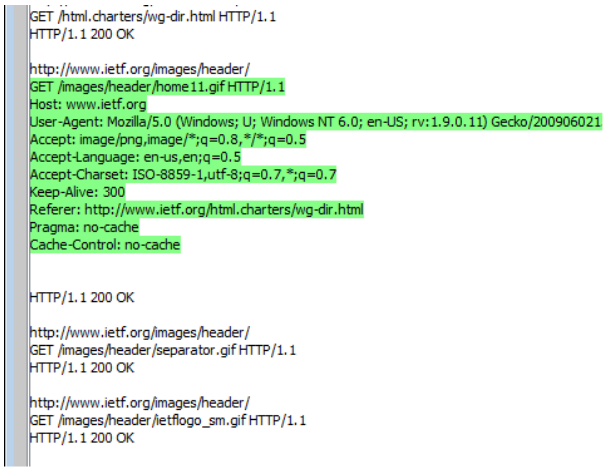
The world needs a small, fast, cross-platform graphical web proxy. People spend a lot of time debugging web applications, and often they do so with complex tools that attempt to solve highly domain specific problems (like finding weak session identifiers, improperly used cookies, etc). Web technologies have diverse enough technical representations that it's often a thorny issue to detect every vulnerability in an automatable, generic way. Even if one can generate vulnerabilities and bugs automatically, a truly thorough web security analysis needs a human performing the analysis to understand the corner cases, new technologies, and business logic vulnerabilities that an automated scanner may not find. With this in mind, gizmo is a tool for people who are doing non-automated web security analysis and want a small, concise tool with which to do so. The goal is to be a concise http editing pocket knife.

The decomposition of the process of website analysis into its constituent parts is the domain of a great many security tools, and applications attempt to attack this problem in different ways. If the website is viewed as a series of vulnerabilities, and the web security tool as an application that helps the user browse the attack surface's vulnerabilities, then a GUI tool with heavyweight analysis features is ideal. This approach works well for what it does, but it doesn't find every vulnerability, and it doesn't serve every need. For programmers, an approach that lets them treat the web session as data to be operated on is slightly more natural. Web technologies change constantly, and someone fluent with the protocols and languages involved is often going to want to work as directly with the protocol or language as possible. Languages, protocols, and technologies change over time, making automated tools in need of repair and updating. A programmer will likely prefer to infer the model from the data themselves, and test the properties of the model by interacting with it directly. For web application security analysis, this often means the programmer will want to work directly with the http traffic, having tools with a set of features targeted at letting them view and edit http traffic. This is gizmo's approach. Gizmo provides functionality for viewing, manipulating, searching, and sending http requests. It does no automated analysis.

A short introduction follows to give the give the reader a flavor of a typical gizmo session. Java 1.6 is gizmo's only prerequisite. To install gizmo, unzip gizmo.zip into a directory of your choice, then click on gizmo.jar in the resulting directory. At this point, a small window like the one below appears:



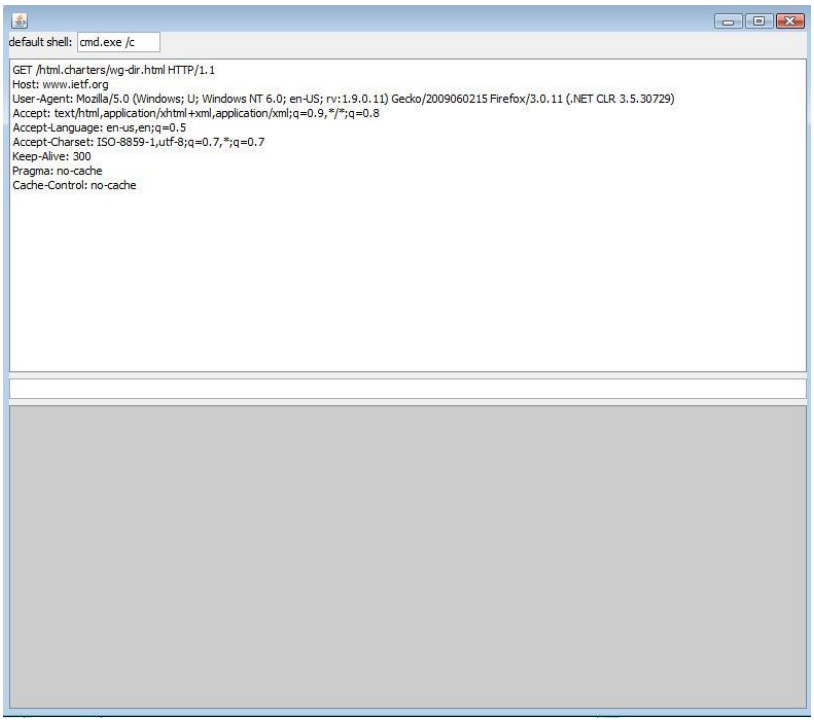
The title bar at the top is used for information about the state of gizmo. In the image above, gizmo is listening on port 8080 (on localhost), and is not intercepting. The port can be altered by selecting File->Config and entering a port. The title bar will be updated to reflect the new port setting. The user then sets their web browsers proxy to use localhost:8080. Browsing the web at this point will cause http requests to show up in gizmo in the tab marked 'summary'. The display in this tab represents each http message that passes through gizmo with the status line of the message. The status line is considered a sort of 'minimized' state of display.



If the user wants to see the full size of the request (maybe to further identify it), they can scroll down to the request (using the arrow keys, j/k, home/end or G/H) and hit <enter> to expand it. At this point the request will expand to its full size. One issue that gizmo handles slightly differently than other tools is the issue of handling different request encodings. A common tactic is to decode the request automatically or in a user-invokable way in the GUI. As a tool, gizmo deals primarily with text, and pictures or hex editors look awkward embedded in a text display. There are a number of external tools that will display pictures or provide hex

editing much better than a web proxy will. Gizmo makes it very easy to interoperate with other tools, leaving the user to decide how to process requests. Gizmo provides an editor/terminal that lets the user process requests via external programs (python scripts, shell scripts, hex editors, image viewers, etc). The integration is done in a way that should allow any program that takes command line arguments to interoperate nicely with gizmo's command shell mode.

To use editing mode, scroll down to a request you'd like to process and hit 'e'. A window like the following is displayed:



The current request is copied into the upper pane in this window after the user invokes editing mode. The request can be edited manually. When the window is closed (either by clicking it closed or hitting escape), the contents of the upper pane get placed in the summary tab of the main window as a new request. The request can then be sent by typing 's'. This editor window can also be used as a terminal to modify the request via external programs, interpreted languages, or shell scripting environments. The terminal mode is built around a java's exec function (java.lang.Runtime getRuntime().exec(String[] cmdarray, String[] envp).

Any time the user hits enter at the command line in the editor window, the strings in the default shell field and in the command line field are concatenated and run. If the default shell field were blank, and the user types:

cmd.exe /c echo 'hello'

on the command line, the command would be run. After the command is run, stdout and stderr of the resulting process are be displayed in the grey area at the bottom . If the defaults hell window had 'cmd.exe /c echo' and the user typed 'hello' on the command line, you would see the same result. The 'default shell' field is where the user enters the shell prefix needed to invoke their commands ('sh –c', 'cmd /c', 'python –c', etc). This is provided as a typing convenience to save the user from retyping their shell constantly.

The actual integration with gizmo comes in the form of an environment variable. Before the command is executed, the request is serialized to a temporary file, and the filename is stored in the BUF environment variable. Pass %BUF% or $BUF to your script, and your script can work directly with the contents of the request. After the command is finished executing, the contents of the file are re-read and used to replace the current contents of the editor window. The flow for programmably manipulating requests like this is:

1. Select request for editing
2. Pass request to script ('script.py $BUF')
3. Tweak results by hand in the editing pane
4. Close editing window
5. Send

Composing multiple decodings is easy to do with a few small scripts. For example, the user could write one small script that base64 decodes a request (base64d below), and another that URL decodes it (urld). The user can compose the two scripts to base64 decode and URL decode a request by running the command:

base64d %BUF% && urld %BUF%

Adding esoteric decodings using this method is as simple as adding another step to the pipeline or modifying existing scripts. This feature can also be used on smaller pieces of the full HTTP message. If a piece of text is selected in the request editor when the user hits enter, gizmo uses that selection as the contents of %BUF% instead of the full request. When the command finishes executing, the selection will be replaced by the new contents of %BUF%. This allows you to apply encodings selectively to different parts of the request.

Basic editing flows like this highlight the intended minimalistic experience. Gizmo includes features for searching requests, manipulating requests, and sending requests.

Intercepting Requests:

The proxy is either 'intercepting' requests, or it's not. This mode can be toggled by clicking the radio button in the lower right or pressing '<CTRL>-I'. In normal operation, requests go straight to the summary tab without user intervention. In intercept mode, requests are put into a holding state in the intercept tab. Requests in the intercept tab have not been sent to the server. Once the user hits 's' on a request, the request is sent and moved over to the summary tab.

Universal Commands:

The following commands work identically whether or not gizmo is in intercept mode.

j, <down>	– moves the cursor down one request
k, <up>	- moves the cursor up one request
G, <end>	- moves to the last request
H, <home>	- moves to the first request
<enter>	- toggles the expanded/contracted view of a request
e	- opens up the editor mode
<ctrl>- i	- toggles intercept mode (can also be done via the radio button in the lower right)

Context-Dependant:

The behavior of these commands depends on whether or not gizmo is intercepting requests.

d	- deletes an intercepted request, sending a proxy error message to the client
s	- s exists whether or not gizmo is intercepting. If gizmo is in intercept mode, it removes the sent request from the intercept tab and places it and the associated response in the summary tab. If gizmo is not intercepting, it sends the request and places the response right below its associated request in the summary pane.

Gizmo is an open source tool that's under active development, with current development focused on:

- adding new means of manipulating requests in gizmo with a focus on a keyboard-driven command interface
- expanding shell manipulation mode

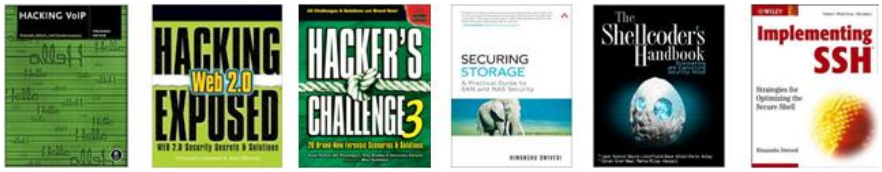
Gizmo releases and source can be found on the web at http://code.google.com/p/gizmo-proxy/.

Appendix A: About iSEC Partners, Inc.

iSEC Partners is a proven full-service security consulting firm, dedicated to making Software Secure. Our focus areas include:

- Mobile Application Security
- Web Application Security
- Client/Server Security
- Continuous Web Application Scanning (Automated/Manual)

Published Books



Notable Presentations

RSA CONFERENCE
Black Hat
web 2.0 EXPO
InfoWorld
COMPUTERWORLD STORAGE NETWORKING WORLD
SNIA

Whitepaper, Tools, Advisories, & SDL Products

- 11 Published Whitepapers
  - Including the first whitepaper on CSRF
- 32 Free Security Tools
  - Application, Infrastructure, Mobile, VoIP, & Storage
- 8 Advisories
  - Including products from Apple, Adobe, and Lenovo
- Free SDL Products
  - SecurityQA Toolbar (Automated Web Application Testing)
  - Code Coach (Secure Code Enforcement and Scanning)
  - Computer Based Training (Java & WebApp Security)