Web application analysis with OWASP Hatkit

Presentation

- Martin Holst Swende
  - Senior Security Consultant, 2Secure AB
  - ~3 years in IT-security
  - ~5 years as software developer
  - Nmap and Webscarab contributor
  - Project leader of OWASP Hatkit projects
  - Twitter: @mhswende
  - Speaker at Defcon19

2Secure

Web application testing

- Is very diverse: from a low-level infrastructure point-of-view to high-level application flow
- There are many tools, but a central component is an intercepting proxy
- Usually complex beasts

Typical proxy features

Feature	Requirement	Must be in proxy?	Possible alternatives
Sitemapping	Traffic data	No	Http-level: trivial. Based on html inspection : e.g. in browser DOM– javascript.
Content analysis	Traffic data	No	W3af, ratproxy, proxmon, webscarab, burp etc
Fuzzing	Traffic data	No	JBroFuzz
Spidering	Traffic data	No	Browser-based spiders with DOM-access. Many choices.
Interception	Live traffic	Yes	None
Manual request	Traffic data + sockets	No	An http/html/json/xml editor + sockets
Manual inspect	Traffic data	No	An http/html/json/xml editor
Sess. id analysis	Traffic data	No	Stompy
Search	Traffic data	No	Wide range: grep to lucene

Typical proxy drawbacks

- Resource intensive
  - All requests/responses get processed/buffered by the proxy regardless if intercepted or not
    - OS security updates, video clips, etc.
  - This usually results in a huge memory footprint
- Static view of request/response data
  - Static GUI, difficult/impossible to customize fields/views
  - Limited filtering and search capabilities
- Limited post-processing capabilities
  - What get's stored in the backend database?
  - Can I export it to other tools?
  - How do I even access the stored data?

The Hatkit Project

Http Analysis Toolkit

- Write an intercepting proxy **Hatkit:Proxy**
  - Lightweight
    - Memory-consumption does not grow with traffic
    - Streams all non-captured traffic to destination
  - Recording
    - Saves to database - MongoDB
      - Document store where parsed data is stored as JSON documents
      - Platform independent, Open Source and fast
- Write an analysis engine **Hatkit:Datafiddler**
  - Flexible
    - Using MongoDB advanced querying facilities
    - Using dynamic views for data
  - And open
    - With several different ways to analyze, export and use existing applications.

Hatkit:Proxy

the intercepting recording proxy

- Based on OWASP Proxy (by Rogan Dawes)
- Records traffic to DB
  - parsed object form
  - raw binary data
- Syntax highlighting
- FQ/NFQ intercept mode (think freedom as in telnet)
- Proxy chaining
- Reverse proxy mode
- TCP interception (early beta)
- …This is definitely not your all-in-one proxy!

TCP interception

- Setup interception
  - DNS poisoning
  - /etc/hosts
- Possible to alter packets using
  - Manual editing as hex/string
  - ”Processor” – a BeanShell script
- Processors
  - Each TCP session has their own script engine instance
  - Possible to keep state and record info in registry

Hatkit:Datafiddler

The analysis engine

- What is it?
- What does it do?
- Why use it?
- How do I get it?
- What does it run on, prerequisites?

Hatkit Datafiddler

- What is it?
  - A framework to analyse web traffic
  - A platform based on MongoDB, with additional functionality to extract and display information geared towards web application testing
  - A platform for utilising existing tools on pre-recorded data

Hatkit Datafiddler

- What does it do?
  - Displays traffic data as defined by the user (Tableview)
  - Traffic and pattern aggregation (Aggregator)
  - Traffic analysis via w3af and ratproxy (3pp)
  - Export recorded traffic to other proxies (3pp)
  - Filter and sort data (filters+tableview)
  - Cache proxying (cache-proxy, beta)

Traffic overview in Tableview

- It is simple to write the kind of view you need for the particular purpose at hand.
- Example scenarios:
  - Analysing user interaction using several accounts with different browsers, you are interested in cookies, user-agent
  - Analysing server infrastructure
    - Server headers,Banner-values, File extensions,Cookie names
  - Searching for potential XSS
    - Use filters to see only the requests where content is reflected
  - Analyzing brute-force attempt
    - Request parameter username, password, Response delay, body size, status code and body hash

Detour - storage

- Traffic is stored as parsed objects in the database.

{{ request:

{ headers: { Host: “server.com”, Cookie: …},

{ uri : { path : “/foobar” , params:{ foo: “bar”}…},

…} //Parsed request object

{ response : …} //Parsed response object

{ request-raw} //Binary raw content

{response-raw} //Binary raw content

}

Demo – Traffic overview

Demo – Traffic overview

Aggregation

- Aggregation (grouping) is a feature of MongoDB.
  - It is like a specialized Map/Reduce
- You provide the framework with a couple of directives and the database will return the results, which are different kinds of sums.
  - Pass JS right into the DB
- Example scenarios:
  - Generate sitemap
  - Show all http response codes, sorted by host/path
  - Show all unique http header keys, sorted by extension
  - Show all request parameter names, grouped by host
  - Show all unique request parameter values, in grouped by host

Demo – Using aggregator

Traffic analysis

- Datafiddler has a mechanism to run selected traffic through third-party plugins.
- Currently implemented*:
  - Ratproxy plugin. Starts ratproxy process, feeds traffic through it, and collects output.
  - Generic proxy plugin. Feeds data to a proxy (e.g Burp) which in turn uses a Datafiddler as forward proxy.
  - Webscarab export. Writes traffic data to webscarab. Useful e.g. to do manual requests edit or use fuzzer.
  - W3af greppers
  - * Defcon19-release

Demo – Ratproxy analysis

Demo – W3af greppers

Demo – Generic exporter

Hatkit Datafiddler

- Upcoming features
  - Cache proxy
    - Datafiddler can act as forwarding proxy and use collected traffic as cache. On cache miss, it can either contact remote host or issue 403.
    - This enables:
      - Resume aborted scans (Nikto, …)
      - Gather e.g. screenshots post mortem without access to target
  - Fuzzer integration
    - Send requests directly to a fuzzer.
  - Text search

Hatkit Datafiddler

- Why use it?
  - To better be able to make sense of large bodies of complex information

Hatkit Datafiddler

- How do I get it?
  - Download the source
    - https://bitbucket.org/holiman/hatkit-proxy/
    - https://bitbucket.org/holiman/hatkit-datafiddler/
  - Or the released binaries
    - https://bitbucket.org/holiman/hatkit-proxy/downloads
    - https://bitbucket.org/holiman/hatkit-datafiddler/downloads
  - And check out the documentation
    - https://www.owasp.org/index.php/OWASP_Hatkit_Proxy_Project
    - https://www.owasp.org/index.php/OWASP_Hatkit_Datafiddler_Project

Hatkit Datafiddler

- What does it run on, prerequisites?
  - Python
  - Qt4
  - PyQt4 bindings
  - Python MongoDB driver
  - MongoDB
  - (optional: w3af)
  - (optional: ratproxy)

- Tested on Linux and MacOSX

Who should care?

- Application testers
  - Hatkit is very useful for analyzing remote servers and applications from a low-level infrastructure point-of-view to high-level application flow.
- Server administrators
  - The Hatkit Proxy can be set as a reverse proxy, logging all incoming traffic.
  - The Datafiddler can analyze user interaction, eg. detect malicious activity and perform post mortem analyzis.
  - The back-end can scale to handle massive amounts of data.

Contact

- To learn more or join the project, join the mailing lists
  - Owasp-hatkit-datafiddler-project@lists.owasp.org
  - Owasp-hatkit-proxy-project@lists.owasp.org

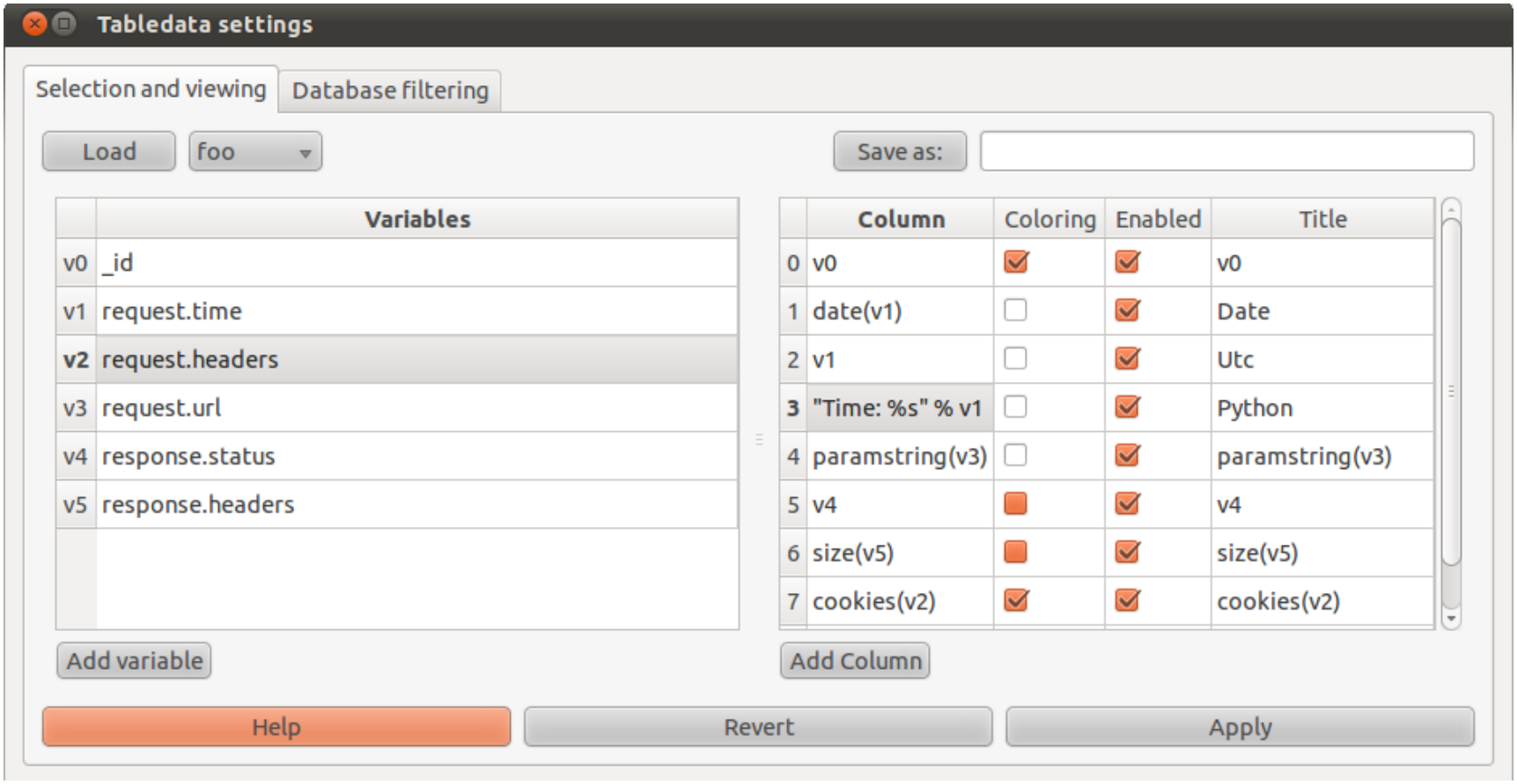
Thank you all for listening

- Questions?

Some additional screenshots…

- … for you who weren't there to see the demos…

Traffic overview



The vo parameter is the object id. This column uses 'Coloring', which means that the value is not displayed, instead a color is calculated from the hash of the value.

Save as:

	Column	Coloring	Enabled	Title
0	v0	☑	☑	v0
1	date(v1)	☐	☑	Date
2	v1	☐	☑	Utc
3	"Time: %s" % v1	☐	☑	Python
4	paramstring(v3)	☐	☑	paramstring(v3)
5	v4	■	☑	v4
6	size(v5)	■	☑	size(v5)
7	cookies(v2)	☑	☑	cookies(v2)

Add Column

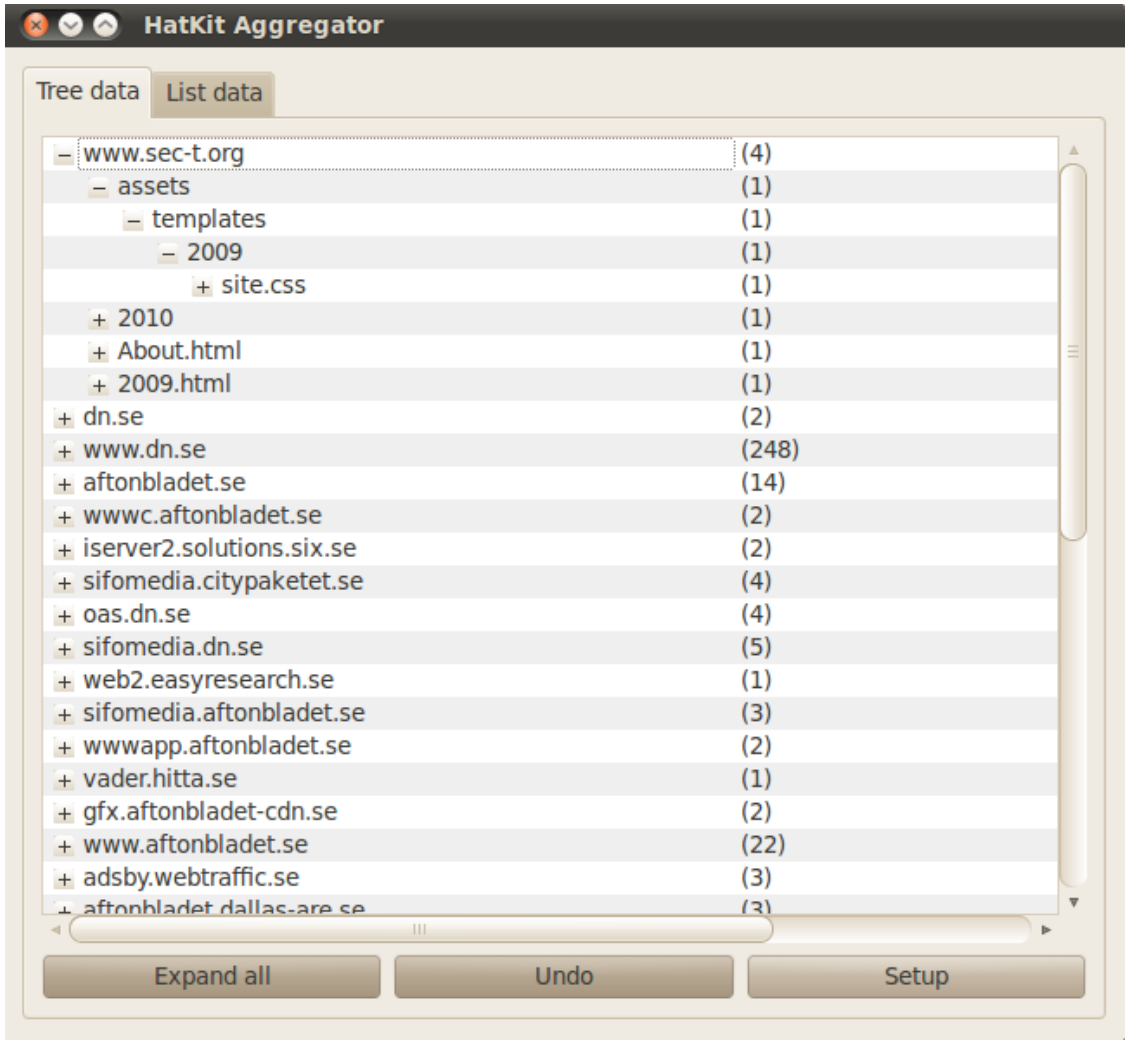
rt Apply

Hatkit Datafiddler

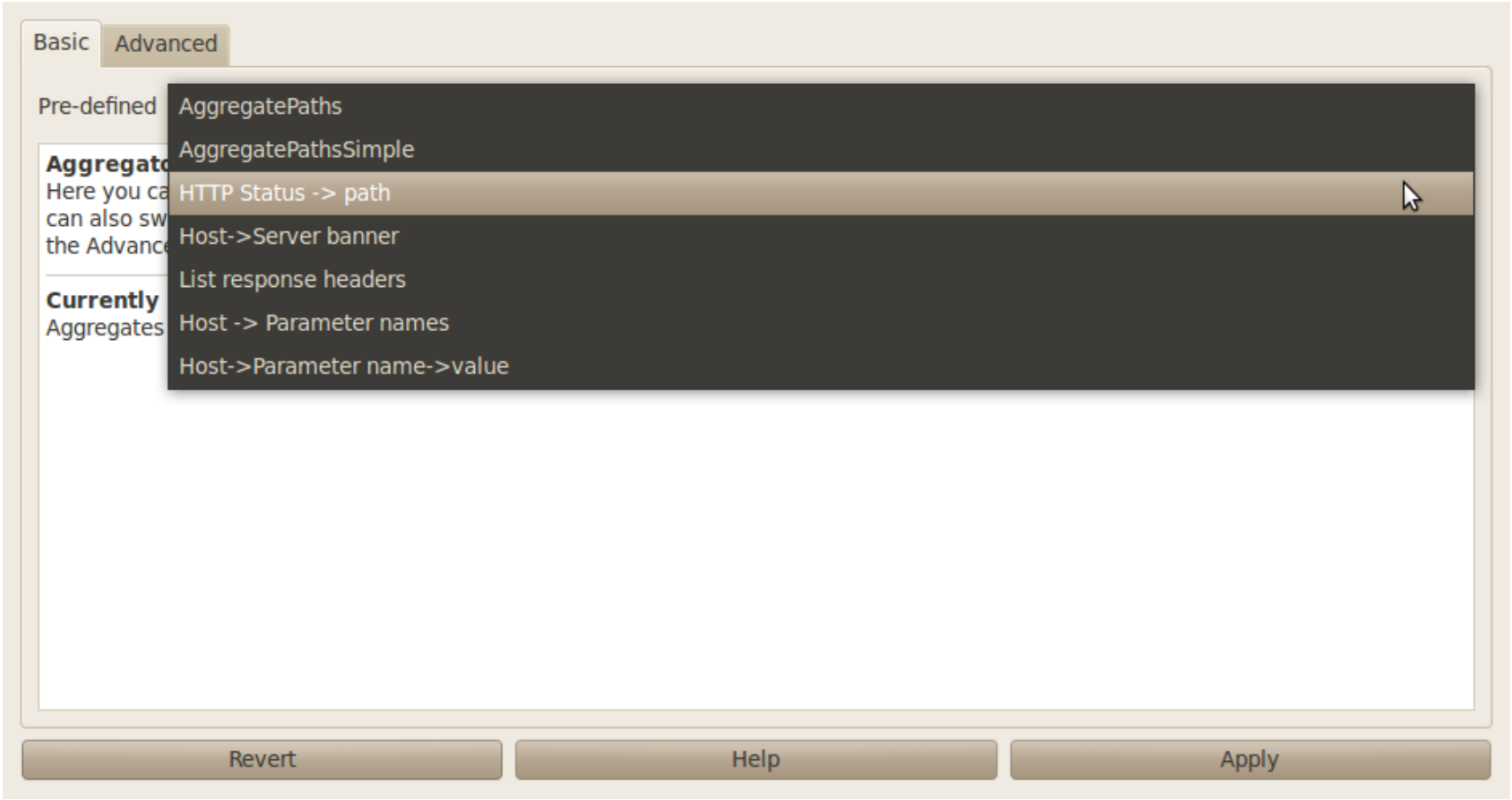
Menu

	v0	Date	Utc	Python
row 0		0317 10:43:41	1268819021004	Time: 1268819021004
row 1		0317 10:43:41	1268819021595	Time: 1268819021595
row 2		0317 10:43:41	1268819021634	Time: 1268819021634
row 3		0317 10:43:42	1268819022199	Time: 1268819022199
row 4		0317 10:43:42	1268819022731	Time: 1268819022731
row 5		0317 10:43:41	1268819021429	Time: 1268819021429
row 6		0317 10:43:41	1268819021610	Time: 1268819021610
row 7		0317 10:43:41	1268819021643	Time: 1268819021643
row 8		0317 10:43:42	1268819022186	Time: 1268819022186
row 9		0317 10:43:42	1268819022221	Time: 1268819022221
row 10		0317 10:43:42	1268819022725	Time: 1268819022725
row 11		0317 10:43:42	1268819022900	Time: 1268819022900
row 12		0317 10:43:42	1268819022920	Time: 1268819022920
row 13		0317 10:43:42	1268819022936	Time: 1268819022936
row 14		0317 10:43:42	1268819022938	Time: 1268819022938
row 15		0317 10:43:42	1268819022945	Time: 1268819022945
row 16		0317 10:43:42	1268819022921	Time: 1268819022921
row 17		0317 10:43:42	1268819022959	Time: 1268819022959
row 18		0317 10:43:42	1268819022992	Time: 1268819022992

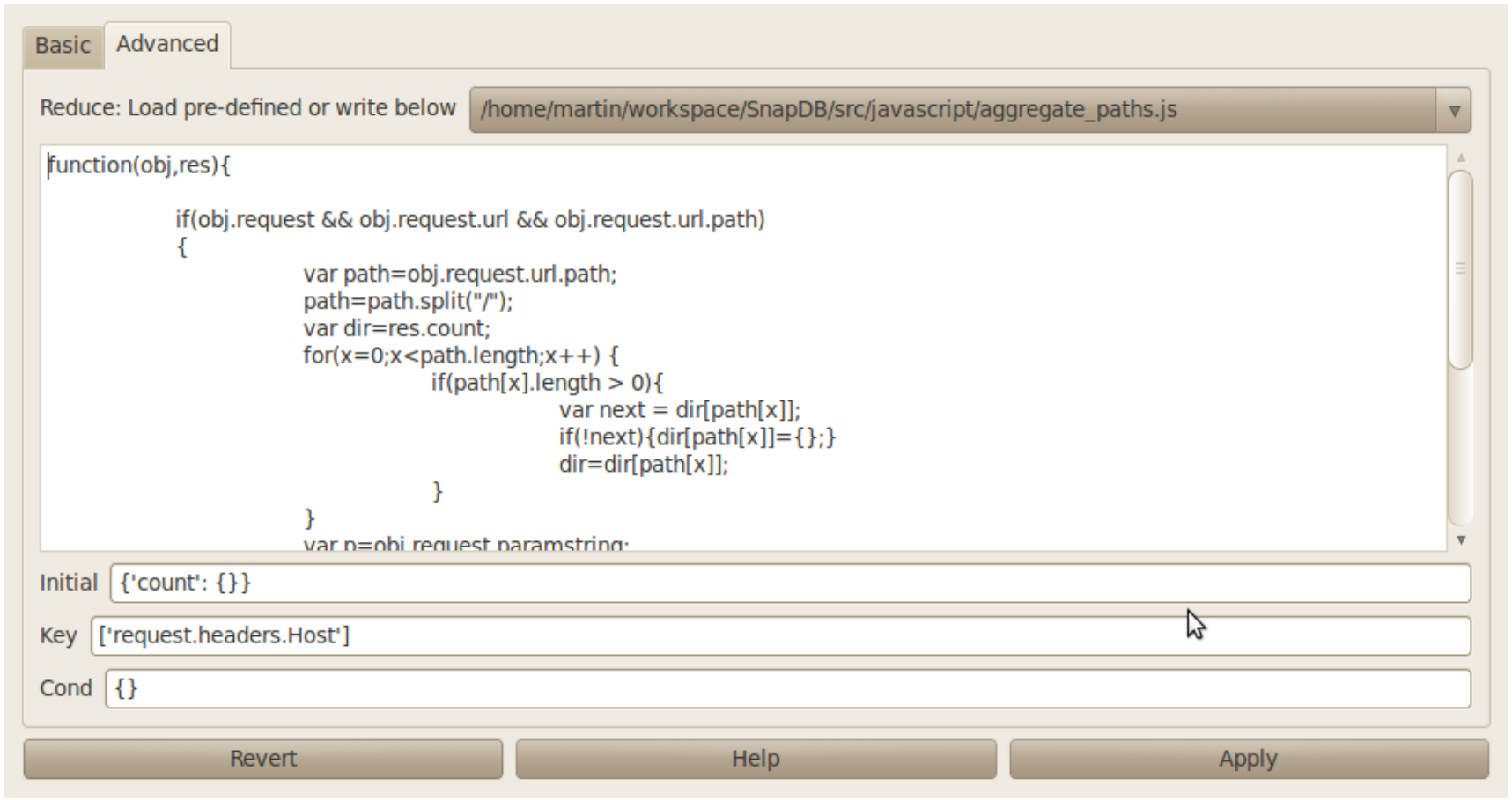
HatKit – Aggregator



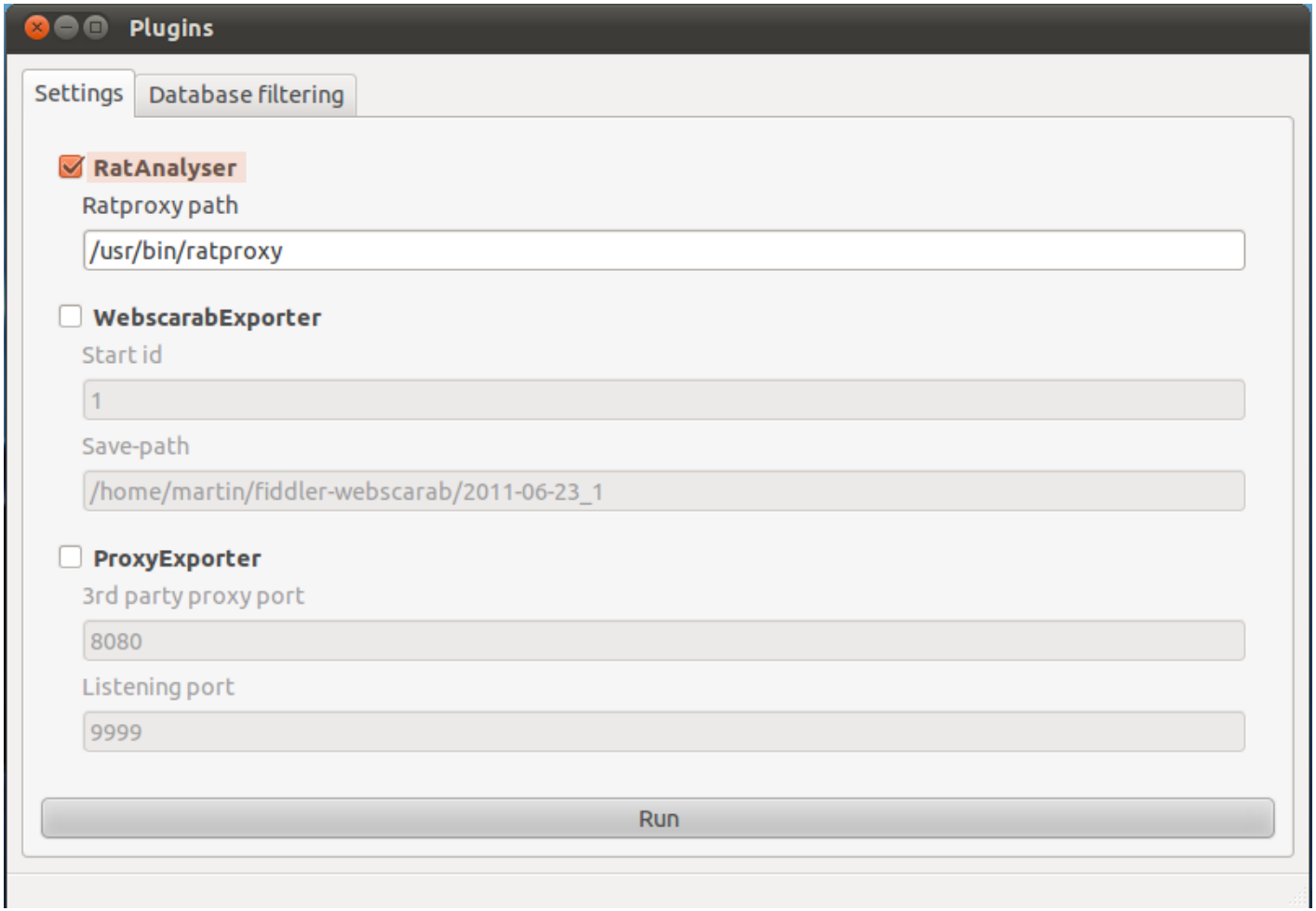
HatKit – Aggregator



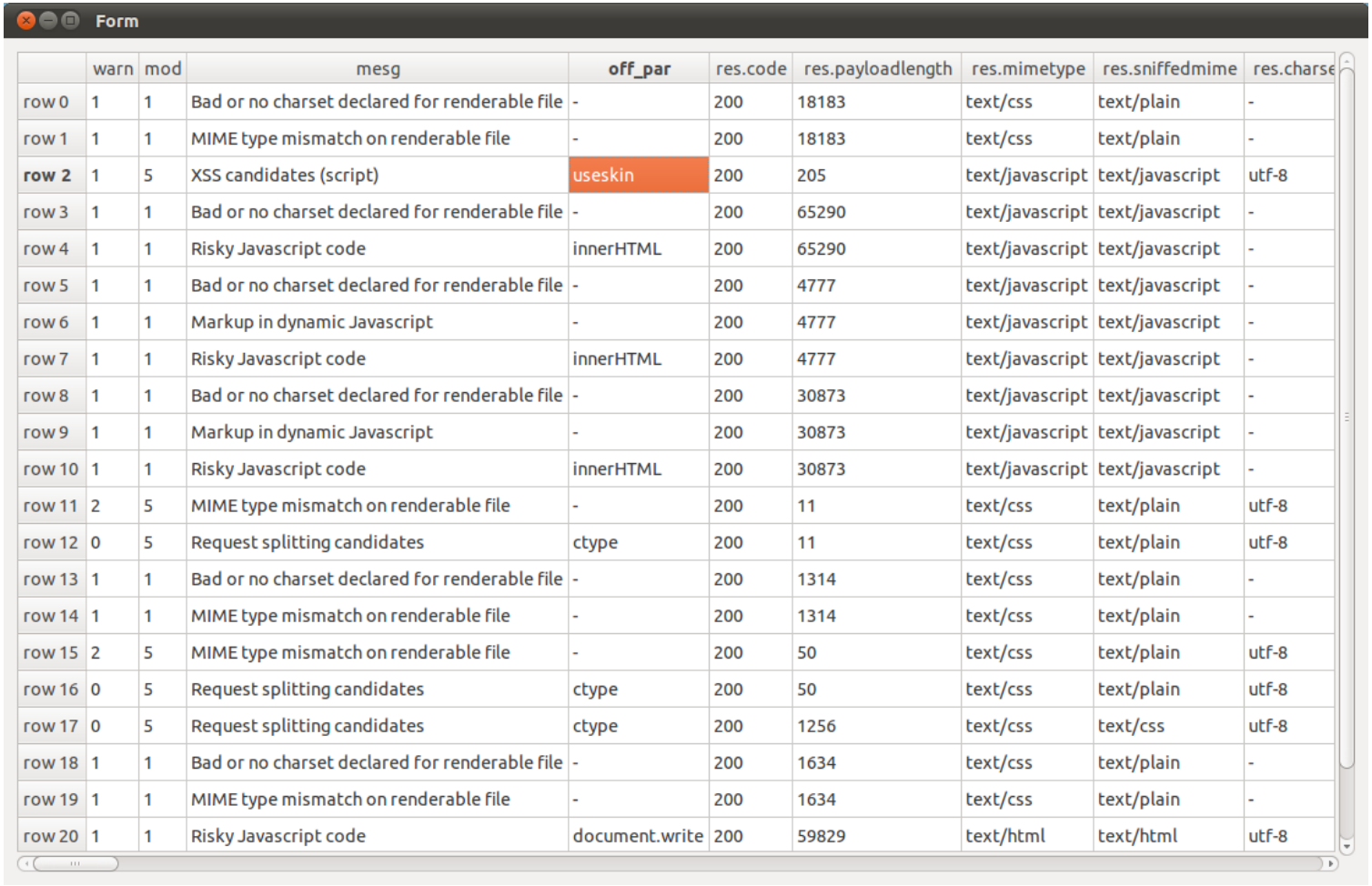
HatKit – Aggregator



Traffic analysis via ratproxy



Traffic analysis via ratproxy



	warn	mod	mesg	off_par	res.code	res.payloadlength	res.mimetype	res.sniffedmime	res.charse
row 0	1	1	Bad or no charset declared for renderable file	-	200	18183	text/css	text/plain	-
row 1	1	1	MIME type mismatch on renderable file	-	200	18183	text/css	text/plain	-
row 2	1	5	XSS candidates (script)	useskin	200	205	text/javascript	text/javascript	utf-8
row 3	1	1	Bad or no charset declared for renderable file	-	200	65290	text/javascript	text/javascript	-
row 4	1	1	Risky Javascript code	innerHTML	200	65290	text/javascript	text/javascript	-
row 5	1	1	Bad or no charset declared for renderable file	-	200	4777	text/javascript	text/javascript	-
row 6	1	1	Markup in dynamic Javascript	-	200	4777	text/javascript	text/javascript	-
row 7	1	1	Risky Javascript code	innerHTML	200	4777	text/javascript	text/javascript	-
row 8	1	1	Bad or no charset declared for renderable file	-	200	30873	text/javascript	text/javascript	-
row 9	1	1	Markup in dynamic Javascript	-	200	30873	text/javascript	text/javascript	-
row 10	1	1	Risky Javascript code	innerHTML	200	30873	text/javascript	text/javascript	-
row 11	2	5	MIME type mismatch on renderable file	-	200	11	text/css	text/plain	utf-8
row 12	0	5	Request splitting candidates	ctype	200	11	text/css	text/plain	utf-8
row 13	1	1	Bad or no charset declared for renderable file	-	200	1314	text/css	text/plain	-
row 14	1	1	MIME type mismatch on renderable file	-	200	1314	text/css	text/plain	-
row 15	2	5	MIME type mismatch on renderable file	-	200	50	text/css	text/plain	utf-8
row 16	0	5	Request splitting candidates	ctype	200	50	text/css	text/plain	utf-8
row 17	0	5	Request splitting candidates	ctype	200	1256	text/css	text/css	utf-8
row 18	1	1	Bad or no charset declared for renderable file	-	200	1634	text/css	text/plain	-
row 19	1	1	MIME type mismatch on renderable file	-	200	1634	text/css	text/plain	-
row 20	1	1	Risky Javascript code	document.write	200	59829	text/html	text/html	utf-8