

Distress Detection
Guide to DVD content

December 27, 2012

Contents

1	Introduction	1
2	The DCA replication experiment files	1
3	The forensic investigation files	2
4	Detector files	2
4.1	system directory	3
4.2	systemovrhd directory	4
4.3	systeminstr directory	5
5	Detector effectiveness evaluation files	5
5.1	Dataset files	5
5.2	Experiment automation scripts	6
5.3	Results files	6
6	Performance study files	7
6.1	Dataset files	8
6.2	Application saturation points	9
6.2.1	Test automation scripts	9
6.2.2	Results and analysis files	9
6.3	Runtime overheads	9
6.3.1	Experiment automation scripts	9
6.3.2	Results and analysis files	9
6.4	Attack processing times	10
6.4.1	Experiment automation scripts	10
6.4.2	Results and analysis files	10
6.5	Accumulating alerts	11
6.5.1	Experiment automation scripts	11
6.5.2	Results and analysis files	12

1 Introduction

This DVD contains the source code, datasets, experiment automation scripts, results files, and data analysis files produced during experimentation with Distress Detection (DD).

2 The DCA replication experiment files

The DCA implementation, datasets, experiment automation scripts and full results from the DCA replication experiment can be found within the `dca` directory.

The `dcam-0.1` directory in `dca` contains the source files for the DCA implementation, whilst `libtissuek-1.0` contains the source files for the `libtissue` framework. This is slightly modified where tissue signals are transferred to cells without any interfering noise. Another minor modification enhances the framework with a time stamp for the last periodic signals input. This enhancement was required in order to synchronize the `libtissue` server with `tcreplay`, and to avoid the same input signals being read by the cells multiple times. These modifications concern `tissue.h` and `tissue.c`. The original `libtissue` framework is found in `libtissue-1.0`.

The `dataset` directory in `dca` contains all the attack and normal sessions as well as the probes required for the creation of the dataset. These can be found in the `sessions` and `instrumentation` sub-directories respectively. The `createtcr` sub-directory contains the scripts used to convert the probe log files into antigen and signal `tcreplay` input files that can be found in `botnet`. Finally, the `experiments` directory contains the script for computing the scaling factors from the attack/normal session logs (`scalingfactors.pl`), the experiment automation script (`botnet.expmcav.pl`), and the script utilized to compute the mean MCAVs (`computeavgmcav.pl`).

The `results` directory in `dca` contains the full results along with the `libtissue` parameters file used for the experiments. The full results consist of MCAVs for the antigen sampled during each experiment step. The `analysis` directory contains the files used to analyze the input signals.

3 The forensic investigation files

The probes, datasets, and full results from the forensic investigation can be found within the **dangersig** directory.

The **dataset** directory in **dangersig** contains all the attacks, their normal behavior counterparts, and the probes used. The **apachemods** and **phpbbvuln** sub-directories contain the vulnerable apache module and phpBB scripts respectively. The attacks exploiting these vulnerabilities can be found in **attacks**. These consist of metasploit 3.2 auxiliary modules and their front-end Perl scripts. The Perl scripts that automate the execution of the normal HTTP requests can be found in **normal**. Finally, the probes used during attack and normal HTTP request execution can be found in **instrumentation**.

The **results** directory in **dangersig** contains the results files for the nine attacks and their normal behavior counterparts. Individual results for attacks and their normal behavior counterparts can be found within the sub-directories labeled **A1** to **A9** respectively. The full results consist of the log files that were forensically analyzed. Files are named according to the probe that produces them (e.g. **netprobe** and **hostprobe**). The files whose name is prefixed with an *n* correspond to the normal behavior logs.

4 Detector files

The source code for the distress detectors are included in the **detectors** directory. Its three sub-directories contain the following:-

system Contains the detectors used for the detector effectiveness evaluation.

systemovrhd Contains the upgraded detectors used to measure runtime overheads during the performance study.

systeminstr Contains instrumented detectors used to measure the various processing times during the rest of performance study experiments.

Each of the directories is further divided into the **sigprobes** and **sandbox** directories that contain the client-side and server-side components for each detector respectively. The following file naming conventions are used:

- The *wpt*, *wrt* and *webiot* prefixes identify artifacts associated with the first, second and third detectors respectively. The prefixes represent the component-threat category pairs from which the detection scopes of the detectors are derived, namely: web process tampering (wpt), web repository tampering (wrt), and web I/O tampering (webiot).
- The *cl* and *s* suffixes identify client-side and server-side detector artifacts respectively.
- The *pamp* and *danger* infixes are associated with detector artifacts that process suspicious HTTP requests and attack symptoms respectively, and refer to the immunology terms by which they are inspired.
- All buffering and alert queue directories are identified by the *q* suffix.

4.1 system directory

This directory includes the detectors used for the detector effectiveness evaluation.

Detector 1 client-side artifacts:

wptpampcl.pl The suspect probe.

wptdangercl.pl The symptom probe.

startwpt.pl The front-end script that launches all client-side components with the parameters used during experimentation.

Detector 2 client-side artifacts - same files as detector 1 files but having a **wrt** prefix.

Detector 3 client-side artifacts:

webiotcl.pl The combined suspects and symptom probe.

startwebiot.pl The front-end script that launches all client-side components with the parameters used during experimentation.

Detector 1 server-side artifacts:

wptpamps.pl The suspect alerter.

runcode The C program used for machine-code execution attempts.

wptdangers.pl The symptom alerter and the attack request detector.

startwpt.pl The front-end script that launches all server-side components with the parameters used during experimentation.

wptpampq The directory used to buffer input HTTP requests.

execdir The directory in which content execution is attempted.

wptprimeq The suspect alerts directory.

wptdangerq The symptom alerts directory.

wptalertsq The distress alerts directory.

Detector 2 server-side artifacts - same files as detector 1 files but having a **wrt** prefix.

Detector 3 server-side artifacts:

webiots.pl The symptom/suspect alerter and attack request detector components.

startwebiot.pl The front-end script that launches all server-side components with the parameters used during experimentation.

webiotprimeq The suspect alerts directory.

webiotbufferinq The directory employed to store HTTP requests along with their local context.

webiotalertsq The distress alerts directory.

Miscellaneous server-side - **ddiptables.conf** is the **iptables** configuration file employed to restore the firewall configuration at the start of every detector execution during experimentation.

4.2 systemovrhd directory

This directory includes the version of the detectors as upgraded for the performance study, but not as yet instrumented.

Additional files for detector 3 - **webiotpreprocs.pl** contains the local context aggregation that is shifted to the server side and upgraded for performance. The **webiotbuffs** directory is used to buffer the yet un-aggregated local context events, providing random access to the aggregation process.

4.3 systeminstr directory

This directory includes the instrumented versions of the detectors used for the performance study. In all the source code files, the instrumentation code can be identified as code sections highlighted by the `#instr` comment. Further notes:

Directory size snapshots - The scripts with the `repsnaphots.pl` suffix represent external instrumentation code executed periodically that tracks the sizes of the suspect and symptom alert directories. These scripts are launched through the `start*.pl` scripts.

Interval mode instrumentation - All `start*.pl` scripts take an interval mode argument. Whenever this argument is set to 0 the instrumentation is continuously active. When set to 1, instrumentation code is active only during the first minute of every 5 minute interval. The purpose of the ‘interval mode’ is to keep log file sizes manageable during the long running ‘accumulating alerts’ experiments.

Log directories - All directories with the `*perflogs` suffix store the various log files created by the instrumented detectors.

5 Detector effectiveness evaluation files

All dataset files, experiment automation scripts, and full results for the detector effectiveness experiments can be found in the `de1`, `de2`, and `de3` directories, for detectors 1-3 respectively.

5.1 Dataset files

Dataset files can be found within the `dataset` sub-directory of each of the effectiveness directories. Resources used to create the background traffic can be found in the `normal` sub-directory, whilst resources used for attack creation can be found in the `attacks` sub-directory. Source code for vulnerable phpBB scripts can be found in the `vulnscripts` sub-directory, whilst the source code and binaries for the vulnerable apache module reside within the `dso` sub-directory.

5.2 Experiment automation scripts

Scripts that automate the entire sequence of experiment steps for each detector can be found within the **sessions** sub-directory of each of the effectiveness directories. File contents are as follows:

Selenium test suite files - These are the files names prefixed by (wpt|wrt|http), followed by **attack*** or **normal***. These files contain all attack and background traffic browsing sessions executed by **selenium**.

sessionend.pl - A Perl script hosted on the web application server that carries out a number of tasks at the start and end of each experiment step. Specifically it restarts all client and server-side distress detector processes, performs recovery of web application recovery attacks and re-starts it, and makes a copy of the results of the experiment steps.

(wpt|wrt|webiot)**testfunc.pl** - These perl scripts execute the entire sequence of experiment steps for each detector. These scripts execute the selenium sessions, launch stand-alone attacks, launch attack handlers and call **sessionend.pl** between each experiment step. Comments concerning the pre-requisites of experiment execution can be found at the start of each script.

5.3 Results files

The results files can be found within the **results** sub-directory of each of the effectiveness directories. Each **results** directory in turn contains a series of sub-directories for the results of each experiment step. Within each **stepXXX** sub-directory, there are four sub-directories containing the following content:

- (wpt|wrt)**pampq** - Contains a file for each monitored HTTP request.
- (wpt|wrt|webiot)**primeq** - Contains all suspect alert files.
- (wpt|wrt)**dangerq** - Contains all symptom alert files.
- (wpt|wrt|webiot)**alertq** - Contains all distress alert files.
- **webiotcounters** - Due to the large number involved, the third detector does not retain a copy of them. Instead, each **stepXXX** directory contains a **webiotcounters** file showing the number of HTTP requests, HTTP response chunks, and back-end request packets monitored during that experiment step.

- ***wwwfiles** - The **results** directories for detectors 1 and 2 also contain a number of **stepXXX** directories ending with the **wwwfiles** suffix. These contain files copied from phpBB indicating whether heap overflow attacks executed successfully. In detector 1, files named **stepXXX** indicates the successful establishment of a remote shell connection through which these files are created as part of its handling. In detector 2, the number of **<script>location.replace** ("http://192.168.147.130")</script> strings at the end of each phpBB cache file indicates how many heap overflow payloads executed successfully.

6 Performance study files

All dataset files, experiment automation scripts, results and analysis files from the performance study experiments are found in the **eff1**, **eff2**, and **eff3** directories, for detectors 1-3 respectively. The following file naming conventions are used:

- The *httpperf* substring is used for naming automation scripts that call **httpperf** and are deployed on the workload generation machine, as well as for the logs produced by them.
- The *perfparams* substring is used for naming automation scripts utilized for the application saturation points, and the logs produced by them. The name refers to the function of the saturation point tests in setting parameters for the performance study.
- The *monvrhd* substring is used for naming automation scripts utilized for the runtime overheads experiments, and the logs produced by them.
- The *latency* substring is used for naming automation scripts utilized for the ‘attack processing times’ experiments, and the logs produced by them. This name refers to the detection latency that attack processing times contribute to.
- The *monper* substring is used for naming automation scripts utilized for the ‘alerts accumulation’ experiments, and the logs produced by them. This name refers to the long monitoring periods involved.

6.1 Dataset files

Dataset files comprise the background web traffic encoded as **httperf** session files based on the **selenium** test suites used for the effectiveness evaluation, and an attack for each detector utilized for the ‘attack processing times’ experiment encoded as **selenium** test suites.

For each detector, its corresponding **httperf** session file was produced by first replaying the **selenium** suite, with all web traffic being recorded by a **tshark** probe. The **httperf** session files were then created out of the sequence of requests and the post content captured by **tshark**. The final session files are trimmed to a thousand requests in order to adhere to **httperf**’s limit. The intermediate and final session files resulting from this process reside in the **dataset/httperf** path within each of the directories. The final sessions are those in the ***req.perf** files. Within these session files, individual HTTP requests are defined as separate sessions, enabling full control of the request rate. This is done since **httperf**’s **-rate** command-line option controls the rate of the generated workload at the session, rather than the individual, HTTP request level¹.

These **httperf** session files undergo one final modification before they are ready to use. As **httperf** is unable to handle MIME post content correctly (observed during a number of experiment trial runs), all HTTP POST requests containing forum posts ended up raising suspect alerts. This issue was fixed by a work-around that replaced all POST content that had not raised a suspect alert during the effectiveness evaluation, with a non-MIME string. Therefore, the session files that are actually called by the experiment automation scripts are the ones named ***reqmime.perf**, and reside in the **dataset/sessions** directory.

One further **httperf** session file is found in **eff1/dataset/sessions**. It contains a session file for the baseline step of the ‘application saturation points’ test. This session comprises simply of a single request to **index.html**, which is then called as many times as required using its corresponding automation script.

The **selenium** test suites are very similar to those used for effectiveness evaluation, with minor modifications accounting for the concurrent employment of **httperf**. Given that the web traffic generated by **httperf** fails to

¹<http://www.hpl.hp.com/research/linux/httperf/httperf-man.txt>

change the state of phpBB in terms of post submissions, attacks are preceded by additional requests that ensure the necessary database updates required by the attacks to succeed. In all three cases, these additional requests consist of the creation of a new topic within phpBB's default forum. In the case of third detector, additional follow-up requests ensure the retrieval of the stored malicious forum post. The retrieval of this post ensures the completion of the attack, and therefore the possibility of detecting it.

6.2 Application saturation points

6.2.1 Test automation scripts

These tests are executed through the `*perfparams.pl` Perl scripts found in `dataset/sessions` of each directory. These scripts are intended to be executed from the workload generation machine. These scripts execute `httperf` sessions with a gradually increasing request rate.

6.2.2 Results and analysis files

The output of the test automation scripts comprise `httperf` log files, named `*perfparams.res`, that reside in `results/stepperfparams` of each directory. For analysis, these results are first transformed into a more polished form and stored in the `*perfparams.xls` files, and subsequently are visualized in `*spssgraphs.spv`. These files reside within the `analysis` sub-directory of each directory.

6.3 Runtime overheads

6.3.1 Experiment automation scripts

These experiments are automated through the `*monovrhd.pl` scripts found in `dataset/sessions` of each directory, intended for execution by the workload generation machine. These scripts execute `httperf` sessions at a fixed request rate.

6.3.2 Results and analysis files

Results produced by this experiment consist of `httperf` log files and reside in `results/stepmonovrhd` of each directory. The `*_base.res`, `*_probes.res`, and `*_full.res` files contain the results for the baseline, probes-only, and

full configurations respectively. For analysis, these results are first transformed into a more polished form and stored in `*monovrhd.xls` files, and subsequently are visualized in `*spssgraphs.spv`. All of them reside in the `analysis` sub-directories for each detector.

6.4 Attack processing times

6.4.1 Experiment automation scripts

These experiments are automated through a combination of three scripts found in `dataset/sessions` of each directory, where both `httperf` sessions and `selenium` test suites are executed. The `*latencyhttperf.pl` scripts are launched from the workload generation machine and are responsible for executing the `httperf` sessions. Given the employment of attacks, `sessionend.pl` scripts are deployed on the virtual machine hosting the web application and are responsible to recover the state of the web application following successful attacks.

Finally, `*latency.pl` scripts comprise the front-end scripts through which the experiments are launched. These scripts are deployed on the workload generation virtual machine that resides on the same physical machine as the monitored web application (the one used for effectiveness evaluation). These scripts first connect to the workload generator to initiate `*latencyhttperf.pl` execution, and then proceed to launch the web attack through `selenium`. On attack execution completion, the execution of `*latencyhttperf.pl` is interrupted, and the `sessionend.pl` script is executed in order to recover the state of the web application, restart the detector, as well as retain a copy of the results from the detector log files, as otherwise these will be lost on detector restart. The `*latency.pl` scripts are parameterized by the number of experiment steps to execute and the rate at which background traffic is generated. Through a single command it is possible to execute all experiment steps.

6.4.2 Results and analysis files

The results for this experiment reside in the `results` sub-directory within each directory. For each detector, the directories `step1` to `step100` store the results for each experiment step. In all the three experiments, steps 1-10 corresponds to 10 steps executed at a 1 req/s rate, steps 11-20 the

steps executed at a 2 req/s, and so forth. The `*latencyhttpperf.res` files store the statistics computed by `httpperf` for the background traffic at each step. The lower saturation points as a consequence of the ‘full configuration’ deployment of detectors can be observed from them.

Each `step*` directory contains the log files corresponding to the five distress detector components produced by the instrumented versions of the detectors. These are: `*pampcl.log` and `*pamps.log` for the suspect probes and alerters, `*dangercl.log` and `*dangers.log` for the symptom probes and alerters, and the `*attackid.log` for the attack request detector. Each log entry represents the processing time for a single input/correlation run and has the following structure: ‘*entry identifier*’, ‘*start time-stamp*’, ‘*end time-stamp*’. Some log files have a supplementary `*init` log file that provides for a higher-precision *start time-stamp* for processes that buffer their inputs as files. The `*attackid.log` file has a fourth entry field containing the identifiers for the matched HTTP requests and attack symptoms. This fourth field is left empty for those alert correlation runs that do not raise a distress alert.

The `*latencycompile.pl` scripts, found in the `analysis` sub-directory of each directory, are used to extract individual attack processing times from the instrumentation logs. These scripts iterate through the experiment step directories and for each directory the entries corresponding to the attack are extracted. The process starts by identifying the entry corresponding to the attack in each `*attackid.log`, recognized by its non-empty fourth field, and then iterates through all other log files in order to extract all the associated entries. The extracted results are stored in a `*latencytable.csv` file, which is subsequently transformed into a more polished form stored as `*latency.xls`. Analysis of these results can be found in `*spssgraphs.spv`, with `eff1/latency.comparative.spv` visualizing the combined results for the three detectors.

6.5 Accumulating alerts

6.5.1 Experiment automation scripts

These experiments are automated using three scripts found in `dataset/sessions` of each directory. The `*monperhttpperf.pl` scripts are launched from the workload generation machine, executing the `httpperf` sessions. The

`sessionend.pl` scripts are the same ones used for the ‘attack processing times’ experiments.

In contrast to the ‘attack processing times’ experiments, the `*monper.pl` and `*monperhttpperf.pl` scripts are disconnected from each other, and so experiment execution first requires the execution of `*monper.pl` in experiment start mode, immediately followed by the execution of the `*monperhttpperf.pl` script. At the point in time when the detector crashes, or `*monperhttpperf.pl` completes execution, `*monper.pl` script is executed again in experiment termination mode to make a copy of the log files.

At the start of each experiment step `tsharkbug.pl` is also executed simultaneously to the client-side probes. This script provides a work-around for a `tshark` bug that retains an open handle to deleted ring buffer files, thus continuing to consume secondary storage space. Whilst it is not possible to close these handles on `tshark`’s behalf without actually restarting `tshark`, this work-around flushes the content of the ring buffer files marked as deleted, relinquishing the occupied disk space. In the case of the second detector, the `backupsession.pl` script is also executed at the start of each step. This script takes care of the initiation of the maintenance procedure mid-way through each experiment run.

6.5.2 Results and analysis files

The results are stored in `results/step101-104`, each containing the results of an individual experiment run. Results are similar to the ones produced by the ‘attack processing times’ experiments, but with two differences. First, this time the log files are produced by instrumentation executed in interval mode, and therefore only contain information gathered in the first minute of each five minute interval. Log entries are filtered on the basis of their *start time-stamp*, meaning that processing times with a duration larger than a minute are recorded as long as the time-stamp falls in the first minute of a five minute interval. This approach was taken in order to keep the size of these log files manageable. Second, `*repsnapshots.log` contain the size for the suspect and symptom alert directories.

The `*monpercompile.pl` and `*monperaggr.pl` scripts, found in `analysis` sub-directory of each directory, are used to extract the processing times from the detector instrumentation logs. For each measurement, its corresponding log file(s) is processed to compute the average processing time for each

five minute interval, with the output stored in `*monpertable.csv`. Finally, `*monpertable.csv` is transformed into a more polished as `*monper.xls`. This file also contains the computation of the comparative detector up-times for the various live web-site traffic statistics. Analysis of these results resides in `*spssgraphs.spv`, with `eff1/analysis/monperfunction.comparative.spv` visualizing the combined results for the three detectors.