QUTTERA

URL Scanner

User Manual

Contents

1. Introduction

Quttera URL Scanner is a freeware tool designed especially for URL scanning and detection of malicious files and suspicious scripts hidden in legitimate web content. It is provided as a command line interface (CLI) utility which core is the specially crafted, patent pending Quttera investigation engine. Quttera URL Scanner recursively downloads content of the URL and then investigates it using Quttera investigation engine. Current version utilizes an open source web crawler *wget* which is distributed under GNU GPL V2 license.

1.1. Technology description

Quttera investigation engine is a not-signature based, behavior analysis investigation engine capable to recognize and detect software vulnerability exploits, shell-codes, malicious JavaScript scripts and malicious executable code hidden in legitimate web content like HTML pages, movies, images, PDF files and others.

Investigation engine main parts:

- X86 emulator for detection of shell-codes and sensible, malicious sequences of executable CPU instructions
- JavaScript emulator for detection of malicious JavaScript code and HTML pages
- PDF reader emulator for detection of malicious PDF files

1.2. Features and functionality

- Recursively downloads the content accessible via provided URL and investigates it file by file. Types of detectable web threats could be found in "Configuration" section.
- Data downloaded from URL is deleted immediately after scan is finished
- Current version is limited to a **10Mb** of a downloaded data.
- Detects wide range of web threats types and potentially unsafe items:

  - ✓ Security vulnerability exploits referencing system internals(x86 architecture)
  - ✓ Security vulnerability exploits referencing process internals(x86 architecture)
  - ✓ Sensible sequences of CPU instructions inside text and binary files(x86 architecture)
  - ✓ Hidden Java-script code generated during emulation of the original script or web page
  - ✓ Suspicious Java-script containing code obfuscation or injection of hidden Java-script
  - ✓ Hidden HTML elements generated during emulation of the original script or web page
  - ✓ PDF files containing embedded malicious PE files
  - ✓ PDF files containing hidden suspicious actions
  - ✓ PDF files containing hidden suspicious elements
  - ✓ PDF files containing Java-script code obfuscation
  - ✓ Malformed PDF files
  - ✓ Encrypted PDF files
- Fast and easy to install.

The scan is performed by Quttera investigation emulators and depends on type of an investigated file.

1.3. License

This software is a freeware and may be distributed as long as the original installation package remains unmodified.
Free usage of this software permitted for personal purposes only. The software is provided “AS IS” without warranty of any kind.

2. Installation

2.1. Supported platforms

Currently Quttera URL Scanner supports only Windows NT based operating systems including XP, Vista and Windows 7.
Linux support is in progress and will be available in the future releases.

Supported Windows systems:

- Microsoft Windows XP Professional (SP2, SP3)
- Microsoft Windows XP Home Edition (SP2, SP3)
- Microsoft Windows XP Professional x64 Edition (SP2, SP3)

- Microsoft Windows 7 Starter
- Microsoft Windows 7 Home Basic
- Microsoft Windows 7 Home Premium
- Microsoft Windows 7 Professional
- Microsoft Windows 7 Ultimate

- Microsoft Windows Vista Home Basic
- Microsoft Windows Vista Home Premium
- Microsoft Windows Vista Business
- Microsoft Windows Vista Enterprise
- Microsoft Windows Vista Ultimate

2.2. System requirements

Quttera investigation technology is based on heuristic penetration and emulation of the investigated content and thus **requires sensible amount of operating memory** (RAM) for proper execution. It is suggested to run Quttera URL Scanner on a modern CPU (800Mz) and 768MB of RAM dedicated for investigation purposes.

2.3. Installation on Linux

Currently Linux package is not supported

3. Command line execution mode and supported commands

The following commands are supported by Quttera URL Scanner when the tool is invoked from the command line or the shell window (start=>run=>cmd)

3.1. scan

cmd# **qurlscanner scan** <*URL*>
Description: scans URL and prints out the report per each file. Scan result will be saved in the *report.log*.
Command input: URL to scan
Command output: investigation result of a content referenced by provided URL
Command execution example: **qurlscanner scan** *www.quttera.com*
Note: Scan is the default command hence **qurlscanner** *www.quttera.com* is valid as well

3.2. update

cmd# **qurlscanner update**
Description: checks for a new available version of an application and the CDB[1]. Downloads and installs them while replacing the current installation.
Command execution example: **qurlscanner update**
It is best practice to run **update** command immediately after launching the utility for the first time to ensure the components are up to date.

3.3. install

cmd# **qurlscanner install** <path-to-zipped_cdb-files>
Description: installs content of CDB file from provided package.
Command execution example: **qurlscanner install** *C:\my_path\myfile.cdb*

3.4. version

cmd# **qurlscanner version**
Description: prints version and license information of currently installed application.
Command execution example: **qurlscanner version**

4. Execution in an interactive mode

Utility can be run from Start menu as well as via Shortcut. Default installation creates them automatically unless specified otherwise in the installation wizard.
This mode supports all commands listed in the previous section but without a need to type **qurlscanner** before each command name.
It is best practice to run **update** command immediately after launching the utility for the first time to ensure the components are up to date.

[1] Learn more about CDB database at http://en.wikipedia.org/wiki/Cdb_(software)

Interactive mode can be "switched on/off" at any stage by appending or removing the "-i" from the Target value in the shortcut Properties.

5. Investigation results

According to the patterns of suspicious or potentially suspicious activity that are detected during the scan of a certain item the result of the investigation of this item is being assigned a corresponded threat level type. Currently, 3(three) major threat level types are being used in the investigation report (see report.log): POTENTIALLY SUSPICIOUS, SUSPICIOUS and MALICIOUS.

For user convenience, the scan output for each such file is highlighted in the CLI with different color. The color scheme is as follows:

Threat Level Type of the file	Scan output color in the CLI
POTENTIALLY SUSPICIOUS	Yellow
SUSPICIOUS	Pink
MALICIOUS	Red

5.1. Potentially Suspicious Types

Investigation Result	Explanation
"File contains very long sled of single byte instructions that may be used as shell-code prefix"	during content penetration X86 emulator detected sensible sequence of single byte instructions that could be a shell-code NOP sled
"Detected hidden JavaScript code"	JavaScript investigation filter detected sensible JavaScript code that was generated during execution of the original script
"Detected suspicious JavaScript execution flow"	JavaScript investigation filter detected potentially suspicious script execution flow which was previously met in suspicious scripts
"Detected hidden HTML tags"	investigation filter detected HTML element that was generated during script emulation
"Detected embedded PE/COFF file"	PDF file includes embedded PE/COFF executable file that may contain malicious program
"Detected hidden suspicious action"	PDF file contains hidden potentially suspicious actions
"Script obfuscation detected"	JavaScript investigation filter detected script implementing code obfuscation technique that maybe used to hide malicious JavaScript code
"Detected encrypted script code"	encrypted PDF document contains suspicious JavaScript code
"Detected memory consumption limit"	investigation modules ran out of memory
"Execution timeout"	during file penetration investigation modules ran out of predefined amount of time

"Detected potentially suspicious too long loop during script penetration"	JavaScript investigation filter entered into infinite loop during JavaScript code penetration
"Detected potentially suspicious too long loop during script execution"	JavaScript investigation filter entered into infinite loop during JavaScript code investigation
"Detected JavaScript code injection"	detected invocation of JavaScript code generated during execution of the original script
"Reached execution stack limit"	during script penetration JavaScript investigation filter reached execution stack limit
"Possibly detected JavaScript packer"	detected JavaScript packer that potentially may be used to hide suspicious JavaScript code
"Detected abnormal long string"	detected abnormal long JavaScript string that may contain packed or encrypted suspicious JavaScript code
"Detected JavaScript code invoked inside event related to hidden DOM element"	detected invocation of JavaScript code from hidden DOM element

5.2. Suspicious Types

Investigation Result	Explanation
"File contains executable instructions that may be used to decrypt hidden shell-code"	detected sensible sequence of executable CPU instruction that may be used for decryption of encrypted shell-codes
"File contains very long sled of fully initialized CPU instructions"	detected sensible sequence of fully initialized CPU instructions that maybe used as a shell-code
"PDF document is malformed"	investigated PDF document is malformed
"Detected hidden iframe tag"	detected hidden iframe tag during investigation of HTML page or JavaScript code
"Detected reference to hidden URL"	detected URL generated during emulation of investigated script
"Detected PDF containing obfuscation of suspicious elements"	PDF document contains hidden suspicious elements
"Detected suspicious binary file"	content of binary file is similar to known shell-codes

5.3. Malicious Types

Investigation Result	Explanation
"File contains CPU instructions referencing operating system internals"	detected sensible sequence of CPU instructions referencing operating system internals
"File contains executable code referencing process internals"	detected sensible sequence of CPU instructions referencing system internals of a process
"Detected multiple procedure calls from process import table"	detected sensible sequence of CPU instructions referencing process internals
"Detected executable code similar to shell-code decoder"	detected sensible sequence of CPU instructions identical to shell-code decoder
"Detected multiple references to process export section"	detected sensible sequence of CPU instructions referencing process internals
"Generic malware"	file detected as malicious using md5 signature database of known malicious files
"Detected embedded malware file"	investigated file contains embedded malware file

6. Application logging and log files

Quttera URL Scanner generates 3(three) types of log files that could be found in the installation directory or at path mentioned in *logs_path* configuration parameter (refer to configuration description section).

6.1. runtime.log

Runtime investigation log (runtime.log) - this log file contains list of URLs mentioned by investigated file, conditions and internal details used to make investigation.

6.2. devel.log

Development log file (devel.log) - this log file contains, mainly, execution errors for development & support teams use.

6.3. report.log

Investigation report log (report.log) - this log file contains sections with detailed investigation results info per each scanned file.

For example:

[scan_task-132620172236]
file_name = www.quttera.com/index.html
file_size = 4294967295
MD5 = 18BD8087F6A4E9BE992527506FB13740

result = 1
details = No suspicious symptoms were found.
Threat = Clean
end_time = Tue Jan 10 15:24:06 2012
scan_time = 0.008000

7. Configuration

7.1. qtreng.conf

Quttera URL Scanner configuration file (qtreng.conf) contains configuration parameters which can be edited to control and/or customize the utility functioning.

This file is logically divided into four configuration sections:

- general - contains parameters which impact execution of an entire application
- investigation - contains parameters which impact investigation process
- *jsfilter_general* - contains parameters for JavaScript investigation filter
- *jsfilter_rules* - contains investigation rules and investigation logic configuration

NOTE: Configuration parameters are still in development and may change between application versions.

Default configuration file can be found at: http://quttera.com/download/qtreng.conf

7.2. Parameters

General

Name	**Description**	**Default Value**
log_file_name	name of runtime log file	*runtime.log*
report_file_name	name of investigation results log file	*report.log*
devel_file_name	development log file	*devel.log*
max_memory_limit_b	maximal amount of memory that could be used during investigation process(Linux only)	*419430400*
cdb_path	path to location of CDB signature files	*INSTDIR*
crawler_path	path to web crawler location(currently not supported)	*INSTDIR*
logs_path	path to location of all log files	*INSTDIR*
log_charset	character set used to write log files: *ascii* or *unicode*	*ascii*

Investigation

Name	Description	Default Value
invoke_x86_emu_on_text_files	boolean value that specifies whether to investigate text files (HTML,JS,CSS...) on presence of ascii shell-codes or not	*false*

jsfilter_general

Name	Description	Default Value
js_parser_timeout_in_seconds	JavaScript parser execution timeout in seconds	*60*
js_emulation_timeout_in_seconds	JavaScript emulator execution timeout in seconds	*120*
js_emulator_stack_size	size of an execution stack used by JavaScript emulator	*200*
clifdom_file_path	path to location of clifdom.conf configuration file	*INSTDIR*

jsfilter_rules

Name	Description	Default Value
detect_infinite_loops	boolean value that specifies whether existence of infinite loop should be detected as suspicious	*true*
detect_js_packers	boolean value that specifies whether to alert upon detection of JavaScript packers or not	*true*
treat_string_procs_as_suspicious	boolean value that specifies whether JavaScript string operations should be treated as suspicious or not	*true*
detect_recursive_emulation_stack_limit	boolean value that specifies whether JavaScript emulator should alert when an execution stack limit is reached during recursive emulation or not; if this parameter is set to true, such scripts will be detected as suspicious	*true*
detect_linear_emulation_stack_limit	boolean value that specifies whether JavaScript emulator should alert when an execution stack limit is reached during linear emulation or not; if this parameter is set to true, such scripts will be detected as suspicious	*true*

detect_memory_consumption_limit	boolean value that specifies whether abnormal number of allocated JavaScript objects should be treated as suspicious or not	*true*
max_js_sring_length	script allocated string larger then this size will be treated as suspicious	*524288*
max_js_array_length	script allocated array beyond this limit will be treated as suspicious	*1024*
max_nested_suspicious_procs_limit	maximal number of nested suspicious calls permitted for script use; script that invoked more nested calls will be treated as suspicious	*2*
detect_hidden_susp_procs_invocation	boolean value that specifies whether hidden invocation of suspicious procedure should be treated as malicious or not	*true*
detect_hidden_iframes	specify how to treat hidden iframes: 0 - don't detect hidden iframes 1 - detect all hidden iframes 2 - detect hidden iframe only if it contains correct URL	*1*
hidden_iframe_size	minimal width and height values of *visible iframe* permitted for use by JavaScript code	*20*
detect_hidden_urls	boolean value to specify whether web pages and JavaScript code containing *hidden urls* should be treated as suspicious or not	*true*
detect_hidden_js_file_injection	boolean value specifies whether include of JavaScript file which name was generated during script execution should be treated as suspicious or not	*true*
detect_hidden_js_code_injection	boolean value specifies whether JavaScript code section containing script which body was generated during script execution should be treated as suspicious or not	*true*
test_strings_entropy	test and verify JavaScript strings entropy; boolean	*true*
entropy_min_test_string_length	minimal length of JavaScript string which entropy value should be verified	*100*
entropy_ref_max_repetition_to_length	test strings on maximal number of repeated 4 byte sequences; if a string contains more “repeated 4 byte patterns” than specified in the	*5*

	parameter value then such string will be treated as suspicious	
entropy_min_suspicious_repetition	maximum permitted number of repetitions of a single character; strings containing greater number of a single characters repetitions will be treated as suspicious	*300*
detect_cross_site_scripts	boolean value specifies whether to detect links containing injection of JavaScript code as a part of URL or not	*true*
detect_hidden_binary_strings	boolean value specifies whether to detect hidden binary strings of format "%uaabb%uaabb" or not	*true*
detect_dynamic_elements_creation	boolean value specifies whether generation of dynamic DOM elements should be treated as suspicious or not	*true*
suspicious_procs_name	list of procedures treated as suspicious	N/A
suspicious_procs_assignment	list of procedures which assignment treated as suspicious	N/A
suspicious_hidden_keywords	list of words which generation during JavaScript execution is treated as suspicious	N/A
max_js_loop_rounds	maximum number of script loops to be executed; loop execution will be aborted when number of cycles will reach this limit	*512*
use_heuristic_method_resolution	boolean value specifies whether to simulate execution of missed procedures and methods or not	*false*
detect_js_code_invoked_from_hidden_elements	boolean value specifies whether invocation of JavaScript code from hidden DOM elements should be treated as suspicious or not	*true*
max_permitted_iframes_per_file	maximum permitted number of iframes per file	*20*

7.3. clifdom.conf

File contains list of domains that are permitted to be used in the hidden iframes. Domains in this list won't be treated as suspicious.

8. Bugs and feature request submissions

Bugs, questions and feature requests could be sent to contactus@quttera.com. In bug submission please attach all three log files listed in "Application logging and log files" section.