Skipfish Package Description

Skipfish is an active web application security reconnaissance tool. It prepares an interactive sitemap for the targeted site by carrying out a recursive crawl and dictionary-based probes. The resulting map is then annotated with the output from a number of active (but hopefully non-disruptive) security checks. The final report generated by the tool is meant to serve as a foundation for professional web application security assessments.

Key features:

- High speed: pure C code, highly optimized HTTP handling, minimal CPU footprint – easily achieving 2000 requests per second with responsive targets.
- Ease of use: heuristics to support a variety of quirky web frameworks and mixed-technology sites, with automatic learning capabilities, on-the-fly wordlist creation, and form autocompletion.
- Cutting-edge security logic: high quality, low false positive, differential security checks, capable of spotting a range of subtle flaws, including blind injection vectors.

Source: https://code.google.com/p/skipfish/
Skipfish Homepage | Kali Skipfish Repo

- Author: Google Inc, Michal Zalewski, Niels Heinen, Sebastian Roschke
- License: Apache-2.0

tools included in the skipfish package

skipfish – Fully automated, active web application security reconnaissance tool

root@kali:~# skipfish -h
skipfish web application scanner - version 2.10b
Usage: skipfish [options ...] -W wordlist -o output_dir start_url [start_url2 ...]

Authentication and access options:

```
  -A user:pass      - use specified HTTP authentication credentials
  -F host=IP        - pretend that 'host' resolves to 'IP'
  -C name=val       - append a custom cookie to all requests
  -H name=val       - append a custom HTTP header to all requests
  -b (i|f|p)        - use headers consistent with MSIE / Firefox / iPhone
  -N                - do not accept any new cookies
  --auth-form url   - form authentication URL
```

--auth-user user - form authentication user
--auth-pass pass - form authentication password
--auth-verify-url - URL for in-session detection

Crawl scope options:

-d max_depth - maximum crawl tree depth (16)
-c max_child - maximum children to index per node (512)
-x max_desc - maximum descendants to index per branch (8192)
-r r_limit - max total number of requests to send (100000000)
-p crawl% - node and link crawl probability (100%)
-q hex - repeat probabilistic scan with given seed
-I string - only follow URLs matching 'string'
-X string - exclude URLs matching 'string'
-K string - do not fuzz parameters named 'string'
-D domain - crawl cross-site links to another domain
-B domain - trust, but do not crawl, another domain
-Z - do not descend into 5xx locations
-O - do not submit any forms
-P - do not parse HTML, etc, to find new links

Reporting options:

-o dir - write output to specified directory (required)
-M - log warnings about mixed content / non-SSL passwords
-E - log all HTTP/1.0 / HTTP/1.1 caching intent mismatches
-U - log all external URLs and e-mails seen
-Q - completely suppress duplicate nodes in reports
-u - be quiet, disable realtime progress stats
-v - enable runtime logging (to stderr)

Dictionary management options:

-W wordlist - use a specified read-write wordlist (required)
-S wordlist - load a supplemental read-only wordlist
-L - do not auto-learn new keywords for the site
-Y - do not fuzz extensions in directory brute-force
-R age - purge words hit more than 'age' scans ago
-T name=val - add new form auto-fill rule
-G max_guess - maximum number of keyword guesses to keep (256)

-z sigfile - load signatures from this file

Performance settings:

-g max_conn - max simultaneous TCP connections, global (40)
-m host_conn - max simultaneous connections, per target IP (10)
-f max_fail - max number of consecutive HTTP errors (100)
-t req_tmout - total request response timeout (20 s)
-w rw_tmout - individual network I/O timeout (10 s)
-i idle_tmout - timeout on idle HTTP connections (10 s)
-s s_limit - response size limit (400000 B)
-e - do not keep binary responses for reporting

Other settings:

-l max_req - max requests per second (0.000000)
-k duration - stop scanning after the given duration h:m:s
--config file - load the specified configuration file

Send comments and complaints to <heinenn@google.com>.

skipfish Usage Example

Using the given directory for output ***(-o 202)*** , scan the web application URL ***(http://192.168.1.202/wordpress)***:

root@kali:~# skipfish -o 202 http://192.168.1.202/wordpress

skipfish version 2.10b by lcamtuf@google.com

- 192.168.1.202 -

Scan statistics:

Scan time : 0:00:05.849
HTTP requests : 2841 (485.6/s), 1601 kB in, 563 kB out (370.2 kB/s)
Compression : 802 kB in, 1255 kB out (22.0% gain)
HTTP faults : 0 net errors, 0 proto errors, 0 retried, 0 drops
TCP handshakes : 46 total (61.8 req/conn)
TCP faults : 0 failures, 0 timeouts, 16 purged
External links : 512 skipped
Reqs pending : 0

Database statistics:

```
      Pivots : 13 total, 12 done (92.31%)
  In progress : 0 pending, 0 init, 0 attacks, 1 dict
 Missing nodes : 0 spotted
   Node types : 1 serv, 4 dir, 6 file, 0 pinfo, 0 unkn, 2 par, 0 val
 Issues found : 10 info, 0 warn, 0 low, 8 medium, 0 high impact
    Dict size : 20 words (20 new), 1 extensions, 202 candidates
   Signatures : 77 total

[+] Copying static resources...
[+] Sorting and annotating crawl nodes: 13
[+] Looking for duplicate entries: 13
[+] Counting unique nodes: 11
[+] Saving pivot data for third-party tools...
[+] Writing scan description...
[+] Writing crawl tree: 13
[+] Generating summary views...
[+] Report saved to '202/index.html' [0x7054c49d].
[+] This was a great day for science!
```