



28 de mayo de 2021  
Ficha N° 5 SKIPFISH  
CSIRT DE GOBIERNO

## Comando de la semana “SKIPFISH”

### I. Contexto

Este documento, denominado “comando de la semana”, tiene como objetivo ilustrar sobre herramientas que pueden ser de utilidad para el lector, a objeto de ir potenciando las capacidades locales de autochequeo, detección simple de vulnerabilidades que están expuestas a internet en sus activos de información y, a su vez, la obtención de una verificación de la subsanación de aquellas que se les han sido reportadas, facilitando la interacción con el CSIRT de Gobierno. El objetivo no es reemplazar una auditoria de código o evaluación de vulnerabilidades, sino que establecer capacidades básicas de chequeo y obtención de información de manera rápida para temas específicos, como por ejemplo la verificación de la subsanación de alertas o vulnerabilidades reportadas por “CSIRT GOB CL”.

### II. Introducción

¿Qué hacer si desde el CSIRT nos llega un ticket señalando que hay alguna vulnerabilidad en un sitio o sistema web de nuestra institución? ¿Cómo verificamos, una vez que hemos aplicado alguna mitigación correctamente y queremos probar si ha tenido efecto, antes de reportarla como “incidente solucionado” al CSIRT o a nuestros auditores internos?

Para este caso existe un comando Linux que nos ayuda a detectar algunas vulnerabilidades de una manera simple, con una herramienta de código abierto y, en base a sus resultados tomar decisiones de control de acceso, verificación de mitigación u otras estrategias de resolución de problemas: SKIPFISH.

#### ¿Qué es SKIPFISH?

Skipfish es una herramienta de reconocimiento de seguridad de aplicaciones web. Prepara un mapa del sitio interactivo para el sitio de destino mediante la realización de un rastreo recursivo y sondeos basados en diccionarios. Luego, el mapa resultante se contrasta con el resultado de una serie de controles de seguridad activos (en general no disruptivos). El informe final generado por la herramienta está destinado a servir como base para evaluaciones de seguridad de aplicaciones web.



Algunas características relevantes:

Hay una serie de herramientas comerciales y de código abierto con una funcionalidad análoga (por ejemplo, Nikto, Nessus), elija la que más le convenga. Dicho esto, skipfish trata de abordar algunos de los problemas comunes asociados con los escáneres de seguridad web. Las ventajas específicas incluyen:

- ❖ Alto rendimiento: Se ha podido observar más de 500 solicitudes por segundo contra objetivos de Internet, más de 2000 solicitudes por segundo en redes LAN / MAN y más de 7000 solicitudes contra instancias, con un impacto muy modesto en el uso de CPU, de red y de memoria. Esto puede atribuirse a:
  - ❖ La multiplexación de un solo hilo, la E/S de red totalmente asíncrona y el modelo de procesamiento de datos de datos que elimina las ineficiencias de gestión de memoria, programación e IPC ineficiencias presentes en algunos clientes multihilo.
  - ❖ Características avanzadas de HTTP/1.1 como solicitudes de rango, compresión de contenido y conexiones "keep-alive", así como la limitación forzada del tamaño de la respuesta, para mantener la sobrecarga de la red bajo control.
  - ❖ El almacenamiento en caché inteligente de las respuestas y la heurística avanzada del comportamiento del servidor se utilizan para minimizar el tráfico innecesario.
  - ❖ Implementación orientada al rendimiento, en C puro, incluyendo una pila HTTP personalizada.
  - ❖ Facilidad de uso: skipfish es altamente adaptable y fiable. Las características del escáner:
  - ❖ Reconocimiento heurístico de esquemas oscuros de manejo de parámetros basados en rutas y consultas.
  - ❖ Manejo de sitios multi-framework donde ciertas rutas obedecen a completamente diferentes, o están sujetas a diferentes reglas de filtrado.
  - ❖ Construcción automática de listas de palabras basada en el análisis del contenido del sitio.
  - ❖ Funciones de escaneo probabilístico que permiten realizar evaluaciones periódicas y limitadas en el tiempo de sitios arbitrariamente complejos.
  - ❖ Controles de seguridad bien diseñados: la herramienta está pensada para proporcionar resultados precisos y significativos:
  - ❖ Los diccionarios elaborados a mano ofrecen una excelente cobertura y permiten realizar pruebas exhaustivas de pruebas de \$keyword.\$extension en un plazo de tiempo razonable.
  - ❖ Las sondas diferenciales de tres pasos son preferibles a las comprobaciones de firmas para detectar vulnerabilidades.
  - ❖ La lógica del estilo Ratproxy se utiliza para detectar problemas de seguridad sutiles:
    - falsificación de peticiones entre sitios, inclusión de scripts entre sitios, contenido mixto,
    - problemas de desajuste de MIME y charset, directivas de caché incorrectas, etc.



\* Las comprobaciones de seguridad integradas están diseñadas para gestionar situaciones complicadas:

- XSS almacenado (ruta, parámetros, cabeceras), inyección ciega de SQL o XML
- inyección ciega de shell.
- ❖ Firmas de contenido estilo Snort que destacarán los errores del servidor, fugas de información o aplicaciones web potencialmente peligrosas.
- ❖ El post-procesamiento de informes reduce drásticamente el ruido causado por cualquier falsos positivos o trucos del servidor al identificar patrones repetitivos.

Dicho esto, skipfish no es una bala de plata, y puede ser inadecuado para ciertos fines. Por ejemplo, no satisface la mayoría de los requisitos señalados de los criterios de evaluación del escáner de seguridad de aplicaciones web WASC (algunos de ellos a propósito, otros por necesidad); y a diferencia de la mayoría de otros proyectos de este tipo no viene con una amplia base de datos de vulnerabilidades conocidas para para las comprobaciones del tipo "banner".

A continuación se presenta una lista aproximada de las comprobaciones de seguridad que ofrece la herramienta.

- ✓ Defectos de alto riesgo (que pueden comprometer el sistema):
  - Inyección de consultas del lado del servidor (incluyendo vectores ciegos, parámetros numéricos).
  - Sintaxis explícita de tipo SQL en parámetros GET o POST.
  - Inyección de comandos de shell del lado del servidor (incluyendo vectores ciegos).
  - Inyección de XML / XPath del lado del servidor (incluyendo vectores ciegos).
  - Vulnerabilidades de cadena de formato.
  - Vulnerabilidades de desbordamiento de enteros.
  - Ubicaciones que aceptan HTTP PUT.
- ✓ Defectos de riesgo medio (que pueden llevar a un compromiso de los datos):
  - Vectores XSS almacenados y reflejados en el cuerpo del documento (soporte mínimo de JS XSS).
  - Vectores XSS almacenados y reflejados a través de redirecciones HTTP.
  - Vectores XSS almacenados y reflejados a través de la división de cabeceras HTTP.
  - Recorrido de directorios / LFI / RFI (incluyendo vectores restringidos).
  - Variedad de archivos POI (fuentes del lado del servidor, configuraciones, etc.).
  - Vectores de inclusión de scripts y CSS suministrados por el atacante (almacenados y reflejados).
  - Vectores de inclusión de scripts y CSS externos no confiables.



- Problemas de contenido mixto en recursos de script y CSS (opcional).
  - Formularios de contraseña que se envían desde o hacia páginas no SSL (opcional).
  - Tipos MIME incorrectos o ausentes en los renderizables.
  - Tipos MIME genéricos en los renderizables.
  - Conjuntos de caracteres incorrectos o faltantes en los renderizables.
  - Información MIME / charset conflictiva en los renderizables.
  - Directivas de caché incorrectas en las respuestas de configuración de cookies.
- 
- ✓ Problemas de bajo riesgo (impacto limitado o baja especificidad):
    - Vectores de evasión de la lista de directorios.
    - Redirección a URLs proporcionadas por el atacante (almacenadas y reflejadas).
    - Contenido incrustado suministrado por el atacante (almacenado y reflejado).
    - Contenido externo incrustado no fiable.
    - Contenido mixto en sub-recursos no secuenciables (opcional).
    - Envío HTTPS -> HTTP de formularios HTML (opcional).
    - Credenciales HTTP en URLs.
    - Certificados SSL caducados o aún no válidos.
    - Formularios HTML sin protección XSRF.
    - Certificados SSL autofirmados.
    - Certificados SSL que no coinciden con los nombres de host.
    - Directivas de caché erróneas en contenidos menos sensibles.
- 
- ✓ Advertencias internas:
    - Intentos fallidos de obtención de recursos.
    - Superación de los límites de rastreo.
    - Comprobaciones de comportamiento 404 fallidas.
    - Filtrado IPS detectado.
    - Variaciones de respuesta inesperadas.
    - Nodos de rastreo aparentemente mal clasificados.
- 
- ✓ Entradas informativas no específicas:
    - Información general de certificados SSL.
    - Cambios significativos en las cookies HTTP.
    - Cambios en las cabeceras Server, Via o X-....
    - Nuevas firmas 404.



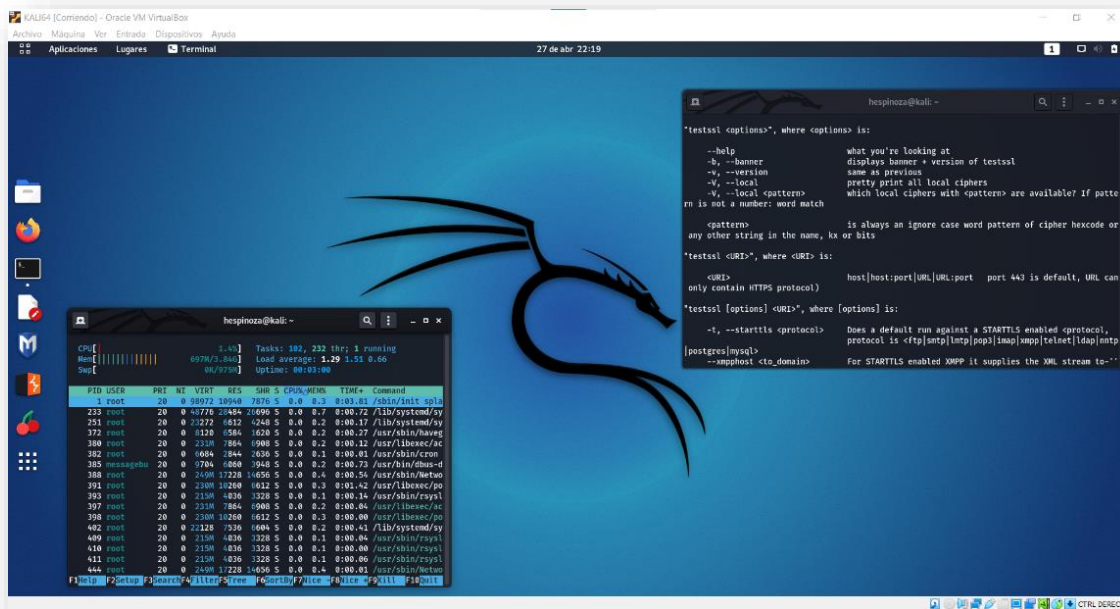
- Recursos a los que no se puede acceder.
- Recursos que requieren autenticación HTTP.
- Enlaces rotos.
- Errores del servidor.
- Todos los enlaces externos no clasificados de otra manera (opcional).
- Todos los correos electrónicos externos (opcional).
- Todos los redirectores de URL externos (opcional).
- Enlaces a protocolos desconocidos.
- Campos de formularios que no han podido ser autocompletados.
- Formularios de introducción de contraseñas (para fuerza bruta externa).
- Formularios de carga de archivos.
- Otros formularios HTML (no clasificados de otra manera).
- Nombres numéricos de archivos (para fuerza bruta externa).
- Enlaces proporcionados por el usuario que se muestran de otra manera en una página.
- Tipo MIME incorrecto o ausente en los contenidos menos significativos.
- Tipo MIME genérico en el contenido menos significativo.
- Conjunto de caracteres incorrecto o ausente en el contenido menos significativo.
- Información MIME / charset contradictoria en el contenido menos significativo.
- Convenciones de paso de parámetros tipo OGNL.



### III. Paso a Paso

#### PASO 1: Un entorno adecuado para trabajar.

Primero debe contar con una distribución de Kali<sup>1</sup> Linux funcionando ya sea en una máquina física o en una máquina virtual<sup>23</sup>.



#### PASO 2: Instalar el comando.

Una vez que se cuenta con este sistema operativo de manera funcional podemos instalar el comando “SKIPFISH”; en general este ya viene preinstalado en la distribución KALI<sup>4</sup>, pero si no fuere así puede instalarlo con los siguientes comandos, **previamente tomando privilegios de usuario “root”**:

```
apt-get install skipfish
```

<sup>1</sup> <https://www.kali.org/downloads/>

<sup>2</sup>

[https://my.vmware.com/en/web/vmware/downloads/info/slug/desktop\\_end\\_user\\_computing/vmware\\_workstation\\_player/16\\_0](https://my.vmware.com/en/web/vmware/downloads/info/slug/desktop_end_user_computing/vmware_workstation_player/16_0)

<sup>3</sup> <https://www.virtualbox.org/wiki/Downloads>

<sup>4</sup> <https://www.offensive-security.com/kali-linux-vm-vmware-virtualbox-image-download/>



### PASO3: Verificar su instalación.

Una vez que se instalado podemos verificar y explorar las múltiples opciones que ofrece para su ejecución:

```
# 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:

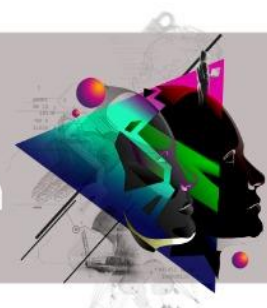
-d max_depth	- maximum crawl tree depth (16)
-c max_child	- maximum children to index per node (512)
-x max_desc	- maximum descendants to index per branch (8192)
-r r_limit	- max total number of requests to send (100000000)
-p crawl%	- node and link crawl probability (100%)
-q hex	- repeat probabilistic 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>.





#### Paso 4: Ponerlo en marcha para verificar nuestra infraestructura.

Algunos ejemplos de ejecución básica para nuestros primeros pasos:

##### EJEMPLO INICIAL

- 1) Se sugiere seleccionar el archivo de diccionario adecuado y configurarlo correctamente. Este paso tiene un que tiene un profundo impacto en la calidad de los resultados del escaneo más adelante, así que no lo omita. Con el mando “find”<sup>5</sup> puede buscar archivos en Linux:

```
# find / -name *.wl  
/usr/share/skipfish/dictionaries/complete.wl  
/usr/share/skipfish/dictionaries/medium.wl  
/usr/share/skipfish/dictionaries/minimal.wl  
/usr/share/skipfish/dictionaries/extensions-only.wl
```

Una vez que haya seleccionado el diccionario, puede utilizar -S para cargar ese diccionario y -W para especificar un archivo inicialmente vacío para cualquier palabra clave específica del sitio recién aprendida (que serán útiles en futuras evaluaciones):

- 2) Entonces crear un archivo vacío para su uso posterior (touch<sup>6</sup>):

```
#touch new_dict.wl
```

- 3) Cree un directorio para almacenar los resultados de la ejecución de skipfish, usando el comando “mkdir”<sup>7</sup>:

```
#mkdir output_dir
```

- 4) Ahora estamos en condiciones de iniciar nuestras pruebas usando “output\_dir”, “new\_dict.wl” y alguno de los diccionarios seleccionado como por ejemplo “complete.wl”:

```
#skipfish -o output_dir -S /usr/share/skipfish/dictionaries/complete.wl -W new_dict.wl  
https://www.csirt.gob.cl
```

Que se ve en una consola KALI después de la ejecución más simple:

<sup>5</sup> <https://es.wikipedia.org/wiki/Find>

<sup>6</sup> [https://es.wikipedia.org/wiki/Touch\\_\(Unix\)](https://es.wikipedia.org/wiki/Touch_(Unix))

<sup>7</sup> <https://es.wikipedia.org/wiki/Mkdir>

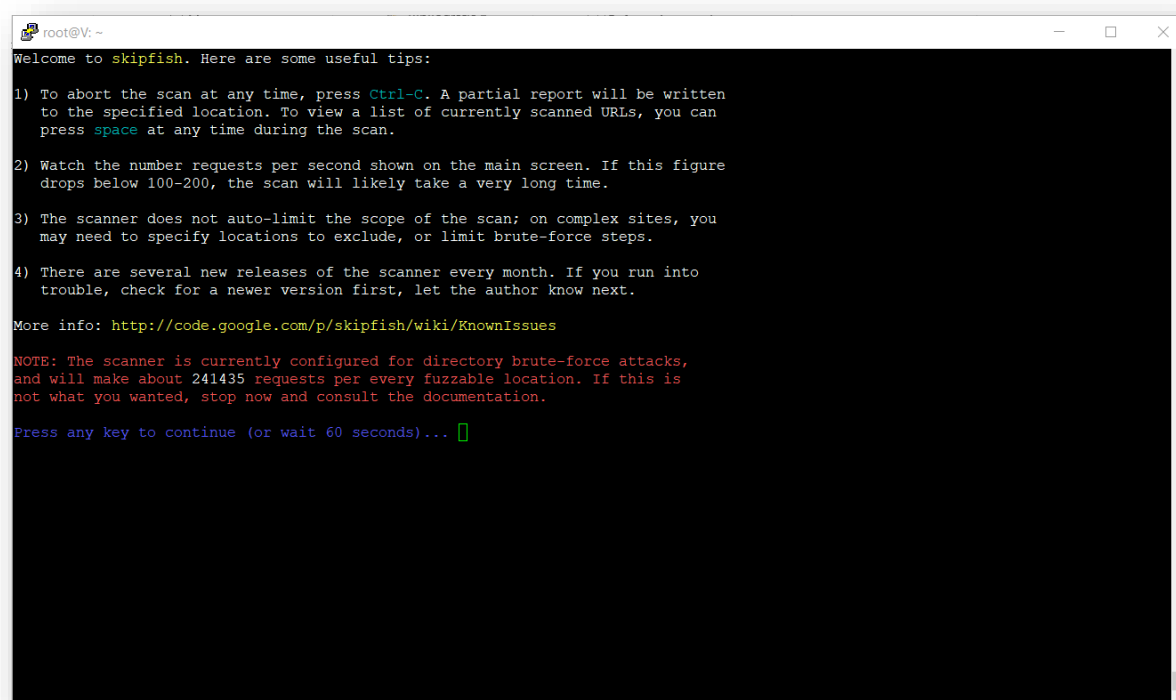


### Vista Parcia de un ejemplo:

Ejecución del comando:

```
#skipfish -o output_dir -S /usr/share/skipfish/dictionaries/complete.wl -W new_dict.wl  
https://www.csirt.gob.cl
```

Después de dar inicio al comando se muestra una nota de utilidad para el evaluador en caso que sea necesario interrumpir dicha ejecución, por ejemplo:



```
root@V: ~  
Welcome to skipfish. Here are some useful tips:  
  
1) To abort the scan at any time, press Ctrl-C. A partial report will be written  
   to the specified location. To view a list of currently scanned URLs, you can  
   press space at any time during the scan.  
  
2) Watch the number requests per second shown on the main screen. If this figure  
   drops below 100-200, the scan will likely take a very long time.  
  
3) The scanner does not auto-limit the scope of the scan; on complex sites, you  
   may need to specify locations to exclude, or limit brute-force steps.  
  
4) There are several new releases of the scanner every month. If you run into  
   trouble, check for a newer version first, let the author know next.  
  
More info: http://code.google.com/p/skipfish/wiki/KnownIssues  
  
NOTE: The scanner is currently configured for directory brute-force attacks,  
and will make about 241435 requests per every fuzzable location. If this is  
not what you wanted, stop now and consult the documentation.  
  
Press any key to continue (or wait 60 seconds)... []
```

Una vez que se presiona alguna tecla cualquiera o se esperan los 60 segundos comienza el procesamiento, mostrándose una imagen como la siguiente con los respectivos indicadores de avance:



```
root@V: ~
skipfish version 2.10b by lcamtuf@google.com

- www.csirt.gob.cl -

Scan statistics:

  Scan time : 0:01:58.252
  HTTP requests : 46478 (414.8/s), 22610 kB in, 9230 kB out (269.3 kB/s)
  Compression : 241 kB in, 1166 kB out (65.6% gain)
  HTTP faults : 0 net errors, 0 proto errors, 0 retried, 0 drops
  TCP handshakes : 474 total (139.8 req/conn)
  TCP faults : 0 failures, 0 timeouts, 1 purged
  External links : 563 skipped
  Reqs pending : 19766

Database statistics:

  Pivots : 373 total, 1 done (0.27%)
  In progress : 341 pending, 29 init, 1 attacks, 1 dict
  Missing nodes : 0 spotted
  Node types : 1 serv, 134 dir, 2 file, 0 pinfo, 234 unkn, 2 par, 0 val
  Issues found : 19 info, 0 warn, 25 low, 47 medium, 0 high impact
  Dict size : 2354 words (139 new), 113 extensions, 256 candidates
  Signatures : 77 total
```

El procesamiento y evaluación toman tiempo, y cuando finaliza se puede ver el reporte que queda en el directorio “output\_dir”.

```
skipfish version 2.10b by lcamtuf@google.com

- 192.168.14.157 -

Scan statistics:

  Scan time : 0:01:08.893
  HTTP requests : 13668 (198.4/s), 8299 kB in, 3907 kB out (177.2 kB/s)
  Compression : 2246 kB in, 6653 kB out (49.5% gain)
  HTTP faults : 0 net errors, 0 proto errors, 0 retried, 0 drops
  TCP handshakes : 307 total (44.5 req/conn)
  TCP faults : 0 failures, 0 timeouts, 9 purged
  External links : 9 skipped
  Reqs pending : 0

Database statistics:

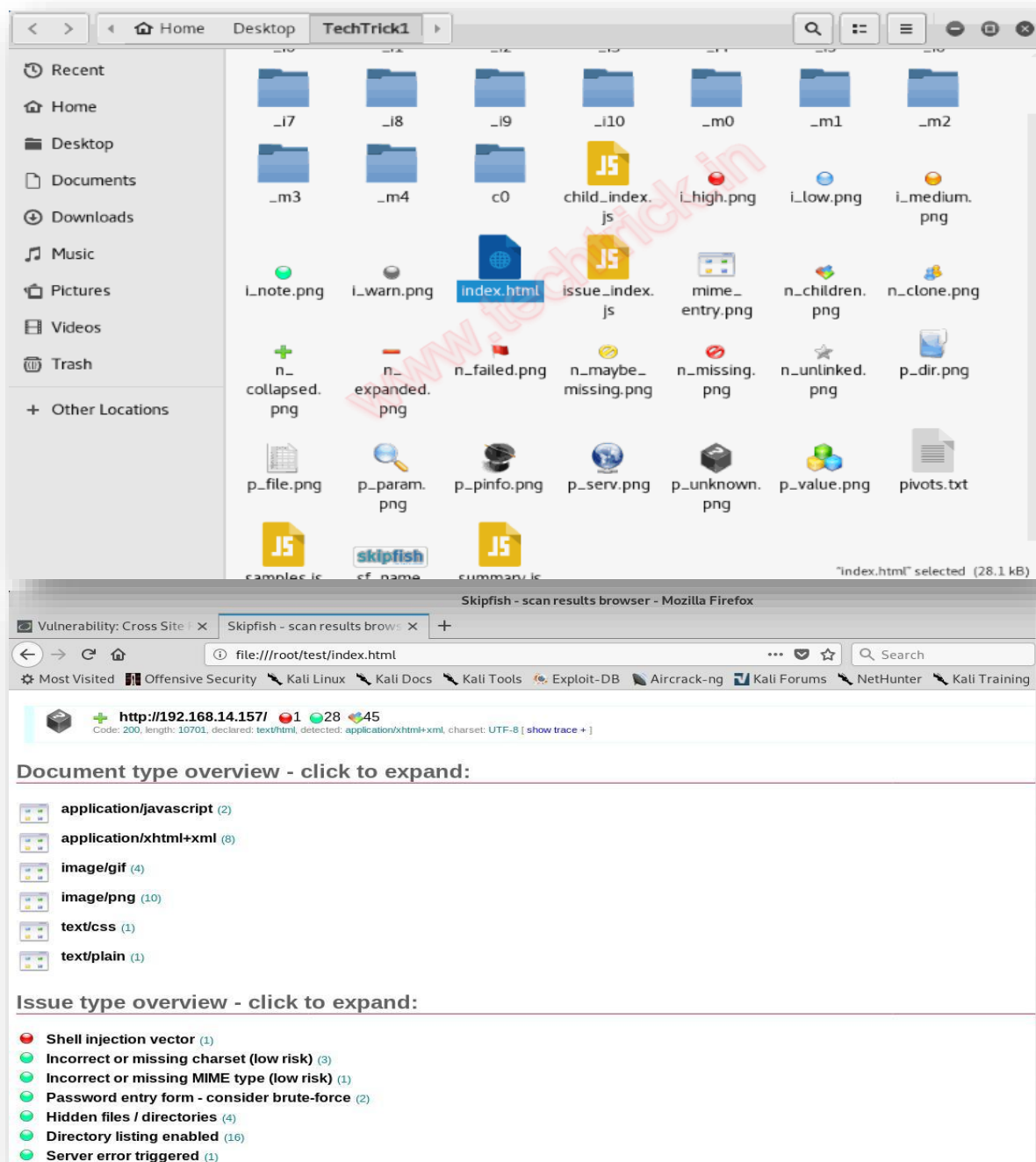
  Pivots : 57 total, 51 done (89.47%)
  In progress : 0 pending, 0 init, 0 attacks, 6 dict
  Missing nodes : 1 spotted
  Node types : 1 serv, 11 dir, 25 file, 5 pinfo, 1 unkn, 14 par, 0 val
  Issues found : 30 info, 0 warn, 0 low, 0 medium, 1 high impact
  Dict size : 52 words (52 new), 5 extensions, 256 candidates
  Signatures : 77 total

[+] Copying static resources...
[+] Sorting and annotating crawl nodes: 57
[+] Looking for duplicate entries: 57
[+] Counting unique nodes: 47
[+] Saving pivot data for third-party tools...
[+] Writing scan description...
[+] Writing crawl tree: 57
[+] Generating summary views...
[+] Report saved to 'index.html' [0x7ee48cbc].
[+] This was a great day for science!
```



Para revisar el archivo con el reporte se sugiere utilizar un browser con capacidad para cargar un documento HTML desde un archivo local. El directorio creado (output\_dir por ejemplo, contiene elementos como los siguientes, destacándose “index.html”).

Una vista ejemplo sería la siguiente:





Una vez que ha finalizado (pueden ser horas de procesamiento, así no pierdan la esperanza), entrega un reporte que se escribe en un archivo "index.html", con los hallazgos encontrados:

El resultado de este comando puede ser usado como evidencia de verificación para indicar que se han subsanado los problemas reportados por CSIRT.

Otros ejemplos de ejecuciones del comando:

Scan type: config

```
skipfish --config config/example.conf http://example.com
```

Scan type: quick

```
skipfish -o output/dir/ http://example.com
```

Scan type: extensive bruteforce

```
skipfish [...other options..] -S dictionaries/complete.wl http://example.com
```

Scan type: without bruteforcing

```
skipfish [...other options..] -LY http://example.com
```

Scan type: authenticated (basic)

```
skipfish [...other options..] -A username:password http://example.com
```

Scan type: authenticated (cookie)

```
skipfish [...other options..] -C jsession=myauthcookiehere -X /logout http://example.com
```

Scan type: flaky server

```
skipfish [...other options..] -l 5 -g 2 -t 30 -i 15 http://example.com
```

Ahora haga que skipfish utilice mywordlist.wl (mi diccionario propio de palabras) y evite que rastree (crawling) el sitio web e intente falsear (fuzz) cualquier cosa:

```
skipfish -W mywordlist.wl -l http://www.example.com/data/ -o tt -O -P -L -V -Y -d 5 -c 86400  
http://www.example.com/data/
```

Las opciones utilizadas significan lo siguiente:

- l http://www.example.com/data/ - only test URLs under this directory
- o tt - write output to "tt" directory
- O - do not submit any forms
- P - do not parse HTML, etc, to find new links
- L - do not auto-learn new keywords for the site
- V - do not update wordlist based on scan results



-Y - do not fuzz extensions in directory brute-force  
-d 5 - maximum crawl tree depth. Not sure if this is necessary.  
-c 86400 - maximum children to index per node. This one seems to be important - tells skipfish not to give up after testing default (512) filenames per directory, but try our whole list

-l only checks include 'string' in the URL

`skipfish -o test -l /aiyou/ http://192.168.1.104`

-X Do not check include 'string' of URL # example: aiyou

`skipfish -o test -X /aiyou/ http://192.168.1.104`

http-based authentication

`skipfish -A admin:password -o test http://192.168.1.104/dvwa/`

authentication based on Cookies

`skipfish -C "name=val" -o test http://192.168.1.104/dvwa/`

Estudie las múltiples opciones que tiene el comando para obtener resultados específicos o redirigir la salida de este hacia otros formatos de archivo, para su inclusión en informes posteriores.

En caso de cualquier inquietud no dudes en consultarnos a soc-csirt@interior.gob.cl.

