OFFENSIVE SECURITY

Penetration Test Report for Internal Lab

marry.poppins@pltravers.writer

OSID: 271828

Table of Contents

1.1. MkDocs

Project documentation with Markdown.

MkDocs is a **fast**, **simple** and **downright gorgeous** static site generator that's geared towards building project documentation. Documentation source files are written in Markdown, and configured with a single YAML configuration file. Start by reading the introductory tutorial, then check the User Guide for more information.
Getting Started User Guide

1.1.1. Features

1.1.1.1. Great themes available

There's a stack of good looking themes available for MkDocs. Choose between the built in themes: mkdocs and readthedocs, select one of the third-party themes listed on the MkDocs Themes wiki page, or build your own.

1.1.1.2. Easy to customize

Get your project documentation looking just the way you want it by customizing your theme and/or installing some plugins. Modify Markdown's behavior with Markdown extensions. Many configuration options are available.

1.1.1.3. Preview your site as you work

The built-in dev-server allows you to preview your documentation as you're writing it. It will even auto-reload and refresh your browser whenever you save your changes.

1.1.1.4. Host anywhere

MkDocs builds completely static HTML sites that you can host on GitHub pages, Amazon S3, or anywhere else you choose.

1.2. Getting Started with MkDocs

An introductory tutorial!

1.2.1. Installation

To install MkDocs, run the following command from the command line:

```
pip install mkdocs
```

For more details, see the Installation Guide.

1.2.2. Creating a new project

Getting started is super easy. To create a new project, run the following command from the command line:

```
mkdocs new my-project
cd my-project
```

Take a moment to review the initial project that has been created for you.

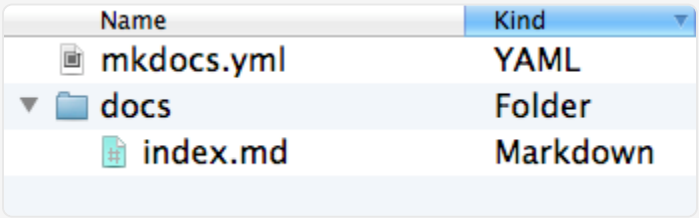

Figure 1 - The initial MkDocs layout

There's a single configuration file named `mkdocs.yml`, and a folder named `docs` that will contain your documentation source files (`docs` is the default value for the docs_dir configuration setting). Right now the `docs` folder just contains a single documentation page, named `index.md`.

MkDocs comes with a built-in dev-server that lets you preview your documentation as you work on it. Make sure you're in the same directory as the `mkdocs.yml` configuration file, and then start the server by running the `mkdocs serve` command:

```
$ mkdocs serve
INFO    -  Building documentation...
INFO    -  Cleaning site directory
[I 160402 15:50:43 server:271] Serving on http://127.0.0.1:8000
[I 160402 15:50:43 handlers:58] Start watching changes
[I 160402 15:50:43 handlers:60] Start detecting changes
[I 160402 15:50:43 handlers:58] Start watching changes
[I 160402 15:50:43 handlers:60] Start detecting changes
[I 160402 15:50:43 handlers:58] Start watching changes
[I 160402 15:50:43 handlers:60] Start detecting changes
[I 160402 15:50:43 handlers:58] Start watching changes
[I 160402 15:50:43 handlers:60] Start detecting changes
[I 160402 15:50:43 handlers:58] Start watching changes
[I 160402 15:50:43 handlers:60] Start detecting changes
```

Listing 1 - Code with lines numbers and lines highlighted

Open up `http://127.0.0.1:8000/` in your browser, and you'll see the default home page being displayed:

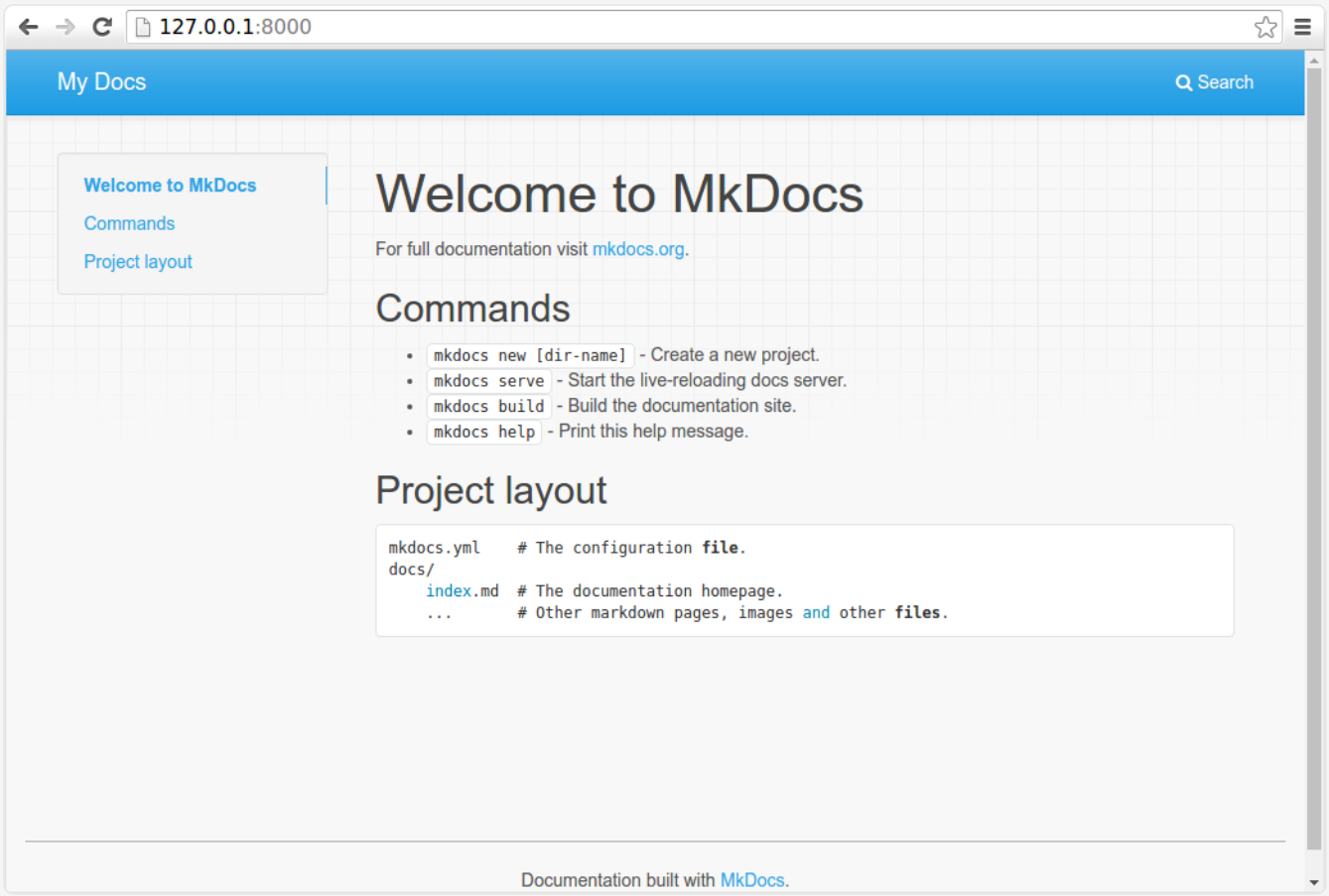

Figure 2 - The MkDocs live server

The `dev-server` also supports auto-reloading, and will rebuild your documentation whenever anything in the configuration file, documentation directory, or theme directory changes.

```
$ mkdocs serve
INFO    -  Building documentation...
INFO    -  Cleaning site directory
[I 160402 15:50:43 server:271] Serving on http://127.0.0.1:8000
[I 160402 15:50:43 handlers:58] Start watching changes
[I 160402 15:50:43 handlers:60] Start detecting changes
[I 160402 15:50:43 handlers:58] Start watching changes
[I 160402 15:50:43 handlers:60] Start detecting changes
[I 160402 15:50:43 handlers:58] Start watching changes
[I 160402 15:50:43 handlers:60] Start detecting changes
[I 160402 15:50:43 handlers:58] Start watching changes
[I 160402 15:50:43 handlers:60] Start detecting changes <halt>
[I 160402 15:50:43 handlers:58] Start watching changes >
[I 160402 15:50:43 handlers:60] Start detecting changes >
```

Listing 2 - Direct HTML `pre code mark` with classes to highlight a part of code

Open the `docs/index.md` document in your text editor of choice, change the initial heading to `MkLorum`, and save your changes. Your browser will auto-reload and you should see your updated documentation immediately.

Now try editing the configuration file: `mkdocs.yml`. Change the `site_name` setting to `MkLorum` and save the file.

```
site_name: MkLorum
site_url: https://example.com/
```

Your browser should immediately reload, and you'll see your new site name take effect.

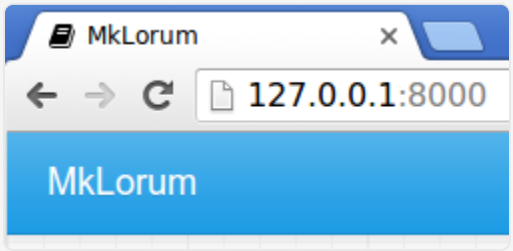

Figure 3 - The `site_name` setting

Note

The `site_name` and `site_url` configuration options are the only two required options in your configuration file. When you create a new project, the `site_url` option is assigned the placeholder value: `https://example.com`. If the final location is known, you can change the setting now to point to it. Or you may choose to leave it alone for now. Just be sure to edit it before you deploy your site to a production server.

1.2.3. Adding pages

Now add a second page to your documentation:

```
curl 'https://jaspervdj.be/lorem-markdownum/markdown.txt' > docs/about.md
```

As our documentation site will include some navigation headers, you may want to edit the configuration file and add some information about the order, title, and nesting of each page in the navigation header by adding a `nav` setting:

```
site_name: MkLorum
site_url: https://example.com/
nav:
    - Home: index.md
    - About: about.md
```

Save your changes and you'll now see a navigation bar with `Home` and `About` items on the left as well as `Search`, `Previous`, and `Next` items on the right.

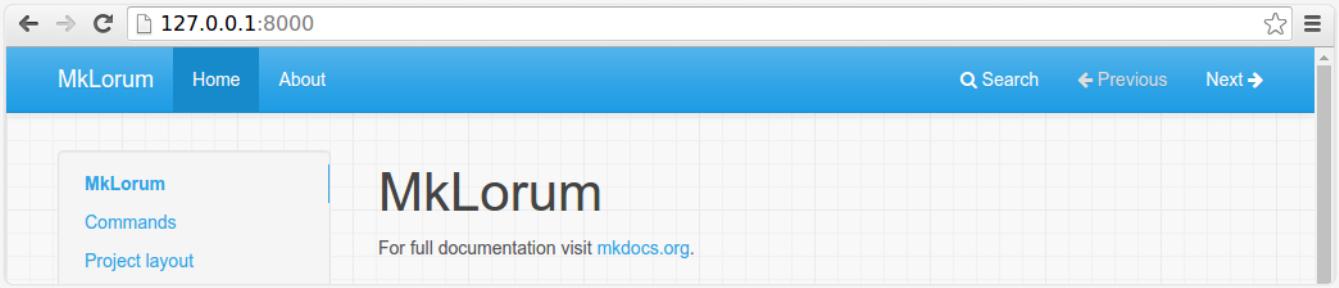

Figure 4 - Screenshot

Try the menu items and navigate back and forth between pages. Then click on `Search`. A search dialog will appear, allowing you to search for any text on any page. Notice that the search results include every occurrence of the search term on the site and links directly to the section of the page in which the search term appears. You get all of that with no effort or configuration on your part!

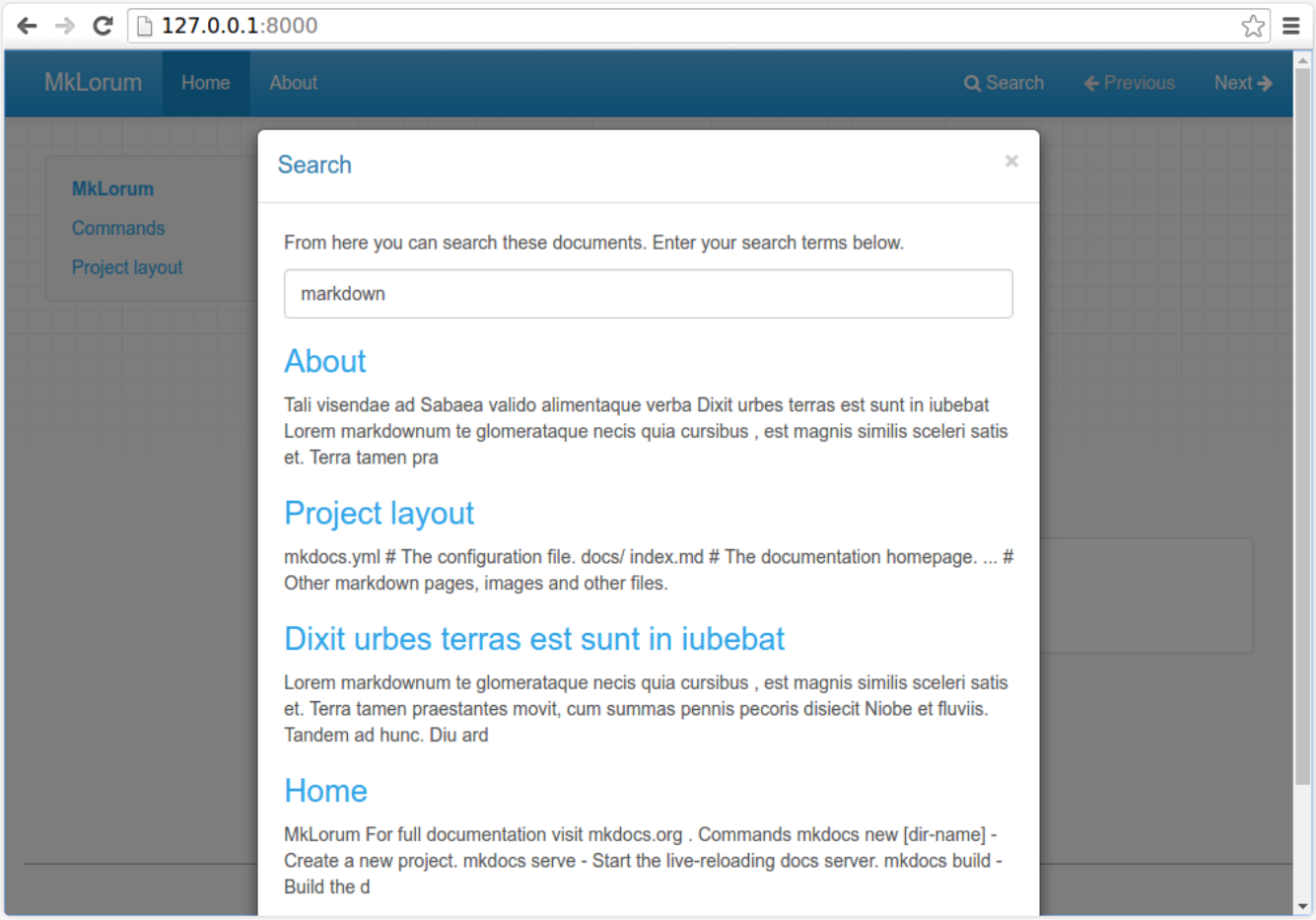

Figure 5 - Screenshot

1.2.4. Theming our documentation

Now change the configuration file to alter how the documentation is displayed by changing the theme. Edit the `mkdocs.yml` file and add a `theme` setting:

```
site_name: MkLorum
site_url: https://example.com/
nav:
    - Home: index.md
    - About: about.md
theme: readthedocs
```

Save your changes, and you'll see the ReadTheDocs theme being used.

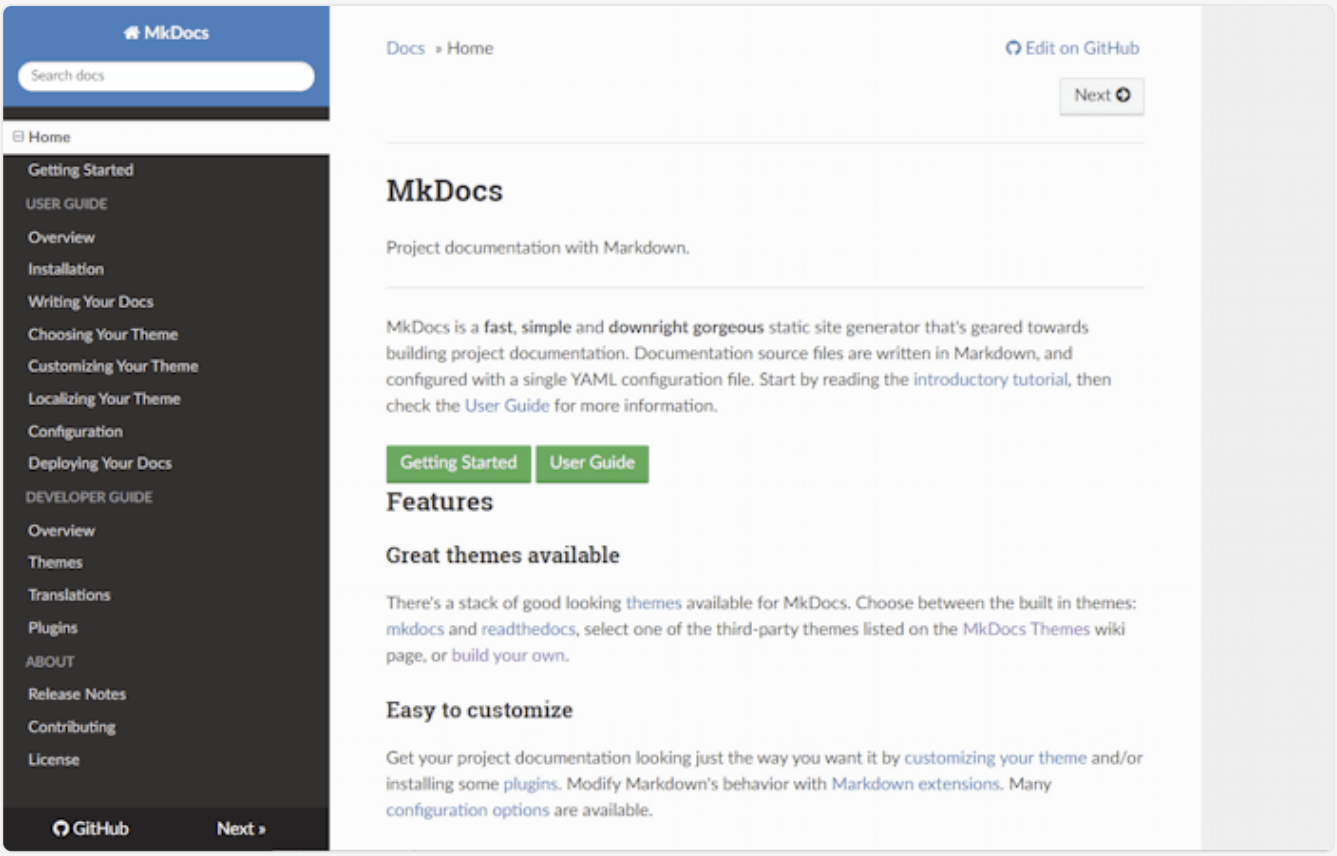

Figure 6 - Screenshot

1.2.5. Changing the Favicon Icon

By default, MkDocs uses the MkDocs favicon icon. To use a different icon, create an `img` subdirectory in the `docs` directory and copy your custom `favicon.ico` file to that directory. MkDocs will automatically detect and use that file as your favicon icon.

1.2.6. Building the site

That's looking good. You're ready to deploy the first pass of your `MkLorum` documentation. First build the documentation:

```
mkdocs build
```

This will create a new directory, named `site`. Take a look inside the directory:

```
$ ls site
about  fonts  index.html  license  search.html
css    img    js          mkdocs   sitemap.xml
```

Notice that your source documentation has been output as two HTML files named `index.html` and `about/index.html`. You also have various other media that's been copied into the `site` directory as part of the documentation theme. You even have a `sitemap.xml` file and `mkdocs/search_index.json`.

If you're using source code control such as `git` you probably don't want to check your documentation builds into the repository. Add a line containing `site/` to your `.gitignore` file.

```
echo "site/" >> .gitignore
```

If you're using another source code control tool you'll want to check its documentation on how to ignore specific directories.

1.2.7. Other Commands and Options

There are various other commands and options available. For a complete list of commands, use the `--help` flag:

```
mkdocs --help
```

To view a list of options available on a given command, use the `--help` flag with that command. For example, to get a list of all options available for the `build` command run the following:

```
mkdocs build --help
```

1.2.8. Deploying

The documentation site that you just built only uses static files so you'll be able to host it from pretty much anywhere. Simply upload the contents of the entire `site` directory to wherever you're hosting your website from and you're done. For specific instructions on a number of common hosts, see the Deploying your Docs page.

1.2.9. Getting help

See the User Guide for more complete documentation of all of MkDocs' features.

To get help with MkDocs, please use the GitHub discussions or GitHub issues.

2. Bloc de Code

2.1. Les blocs de Code MkDocs

Il y a deux types de blocs de code disponible dans MkDocs Material :

- bloc de code `SuperFences` (extension du parser Markdown de Python)
- bloc de code `HTML`

Références :

- Material for MkDocs > Code blocks
- Python-Markdown
- SuperFences

2.1.1. Bloc de Code `SuperFences` MkDocs

Ce type de bloc de code permet de :

- coloriser syntaxiquement le code si un langage est indiqué : ```` ```python ````
- mettre en évidence des lignes ```` ```hl_lines="2 3 10-15" ```` (lignes 2, 3 et de 10 à 15)
- numéroter les lignes ```` ```linenums="5" ```` (numérotation des lignes commençant à 5)

Ces différentes options peuvent être combinées :

````
```python hl_lines="2 5-6" linenums="1"
#!/usr/bin/python
# Note: see how to craft FEALIST in eternalblue_poc.
from impacket import smb
from struct import pack
import sys
import socket
```
````

Listing 3 - Syntaxe de code combinant les différentes options

```python
#!/usr/bin/python
# Note: see how to craft FEALIST in eternalblue_poc.
from impacket import smb
from struct import pack
import sys
import socket
```

Listing 4 - Résultat de code combinant les différentes options

> **Note**
>
> Les lignes mises évidences sont en **rouge OSCP** dans le rendu PDF.

```
#!/usr/bin/python
# Note: see how to craft FEALIST in eternalblue_poc.
from impacket import smb
from struct import pack
import sys
import socket
```

```
ntfea10000 = pack('<BBH', 0, 0, 0xffdd) + 'A'*0xffde

ntfea11000 = (pack('<BBH', 0, 0, 0) + '\x00')*600  # with these fea, ntfea size i
ntfea11000 += pack('<BBH', 0, 0, 0xf3bd) + 'A'*0xf3be  # 0x10fe8 - 0x1c20 - 0xc =

ntfea1f000 = (pack('<BBH', 0, 0, 0) + '\x00')*0x2494  # with these fea, ntfea siz
ntfea1f000 += pack('<BBH', 0, 0, 0x48ed) + 'A'*0x48ee  # 0x1ffe8 - 0x1b6f0 - 0xc
```

2.1.2. Bloc de Code `HTML` MkDocs

Ce sont des blocs de code standard `HTML`, des éléments peuvent être mis en évidence en utilisant la balise `<mark>`.

> **Note**
>
> Il est nécessaire de traduire les chevrons ouvrants en entité HTML `&lt;` s'il peuvent être interprété comme l'ouverture d'un balise HTML.

```
<pre><code>
<mark>$ mkdocs serve</mark>
INFO    -  Building documentation...
INFO    -  Cleaning site directory
[I 160402 15:50:43 server:271] <mark>Serving on http://127.0.0.1:8000</mark>
[I 160402 15:50:43 handlers:58] Start watching changes
[I 160402 15:50:43 handlers:60] Start detecting changes &lt;halt>
[I 160402 15:50:43 handlers:58] Start watching changes < halt>
[I 160402 15:50:43 handlers:60] Start detecting changes <halt>
</code></pre>
```

Listing 5 - Exemple de bloc de code HTML

```
$ mkdocs serve
INFO    -  Building documentation...
INFO    -  Cleaning site directory
[I 160402 15:50:43 server:271]  Serving on http://127.0.0.1:8000
[I 160402 15:50:43 handlers:58] Start watching changes
[I 160402 15:50:43 handlers:60] Start detecting changes <halt>
[I 160402 15:50:43 handlers:58] Start watching changes < halt>
[I 160402 15:50:43 handlers:60] Start detecting changes
```

2.2. Les blocs de Code OSCP

```
#!/usr/bin/python
# Note: see how to craft FEALIST in eternalblue_poc.
from impacket import smb
from struct import pack
import sys
import socket
```

```
#!/usr/bin/python
from impacket import smb
from struct import pack
import sys
import socket

'''
EternalBlue exploit for Windows 7/2008 by sleepya
The exploit might FAIL and CRASH a target system (depended on what is overwritter

EDB Note: Shellcode
- x64 ~ https://github.com/offensive-security/exploit-database-bin-sploits/raw/ma
- x86 ~ https://github.com/offensive-security/exploit-database-bin-sploits/raw/ma

Tested on:
- Windows 7 SP1 x64
- Windows 2008 R2 SP1 x64
- Windows 7 SP1 x86
- Windows 2008 SP1 x64
- Windows 2008 SP1 x86

'''

# Note: see how to craft FEALIST in eternalblue_poc.py

# wanted overflown buffer size (this exploit support only 0x10000 and 0x11000)
# the size 0x10000 is easier to debug when setting breakpoint in SrvOs2FeaToNt()
# the size 0x11000 is used in nsa exploit. this size is more reliable.
NTFEA_SIZE = 0x11000
# the NTFEA_SIZE above is page size. We need to use most of last page preventing

ntfea10000 = pack('<BBH', 0, 0, 0xffdd) + 'A'*0xffde

ntfea11000 = (pack('<BBH', 0, 0, 0) + '\x00')*600  # with these fea, ntfea size i
ntfea11000 += pack('<BBH', 0, 0, 0xf3bd) + 'A'*0xf3be  # 0x10fe8 - 0x1c20 - 0xc =

ntfea1f000 = (pack('<BBH', 0, 0, 0) + '\x00')*0x2494  # with these fea, ntfea siz
ntfea1f000 += pack('<BBH', 0, 0, 0x48ed) + 'A'*0x48ee  # 0x1ffe8 - 0x1b6f0 - 0xc

ntfea = { 0x10000 : ntfea10000, 0x11000 : ntfea11000 }
```


Did I try enough harder? ;-)