



# BurpSmartBuster

A smart way to find hidden treasures

**Patrick Mathieu**

patrick@hackfest.ca

@pathetiq

<https://github.com/pathetiq/burpsmartbuster>

**DerbyCon Recharge - Stable Talk**

September 23rd 2016

# Patrick Mathieu

E: patrick@hackfest.ca

T: @pathetiq

- Co-Founder Hackfest.ca
  - Largest hacking event in Canada
- Application pentester & Purple Teamer at [www.securitycompass.com](http://www.securitycompass.com)
- Start Hacking mid '90s
- College & University in Computer Science
- #BugBounty

---

# Anyone know about BurpSmartBuster?

Except my colleagues in front!

# Description

Bruteforcing non-indexed data is often used to discover hidden files and directories which can lead to information disclosure or even a system compromise when a backup file is found. This brute force technique is still useful today, but the tools are lacking the application context and aren't using any smart behaviour to reduce the brute force scanning time or even be stealthier. BurpSmartBuster, a Burp Suite Plugin offers to use the application context and add the smart into the Buster!

This presentation will reveal this new open-source plugin and will show practical case of how you can use this new tool to accelerate your Web pentest to find hidden treasures! The following will be covered:

- How to **add context** to a web bruteforce tool
- How we can **be stealthier**
- How to **limit the number of requests**: Focus only on what is the most critical
- Show how **simple** the **code** is and how you can help to make it even better

# What is BurpSmartBuster?

- Burp plugin - extension
- Objective to “replace” dirbuster and dirb
- Adding “smart” features to hidden files brute forcing
- It's using Burp to gather requested URLs and brute force files/directories based on this information

# BurpSmartBuster State

- Released at Defcon 24 Demolabs
- Working version on GitHub
- Today I hope/want the **community to join in** to help and make it better with more features and data.
- Listed Todos, documented code for easy addition

# Why BurpSmartBuster?

- After tons of web application pentests using **DirBuster** and **DirB**
  - Efficiency is really low
  - No application context
  - No logic options or anything
- Tons of files got uploaded on web server which should not. Dev, Sysadmin, automated Scripts, etc.

# Current problems

- Files and directories are **based on robots.txt** of alexa top 1000 website.
  - Lead to many **unwanted checks**
- Needs **manual input** to verify all existing directories
- Missing possible **file extensions**
- Not testing files based on **current context**:  
“directories, filename, domain name, etc.”
- TODO: No test based on website technologies



# Why a new tool? I wanted to...

- **Add context** to a web bruteforce tool
- Add a **stealthier** mode
- **Limit the number of requests**: Focus only on what is the most critical (No random list.txt)
- Offer **simple** and **documented code** for the community to help make it bigger and better (more features)

# Objectives

- Brute force static files based on the **application context** :
  - Existing files, extensions and directories
  - Domain name
- Brute force items **based on developers and sysadmin** extensions, files and directories
- Test based on **dynamic content** of the website and **current request**
- TODO : Offer a **stealthier option** : limit number of requests

# BSB Features

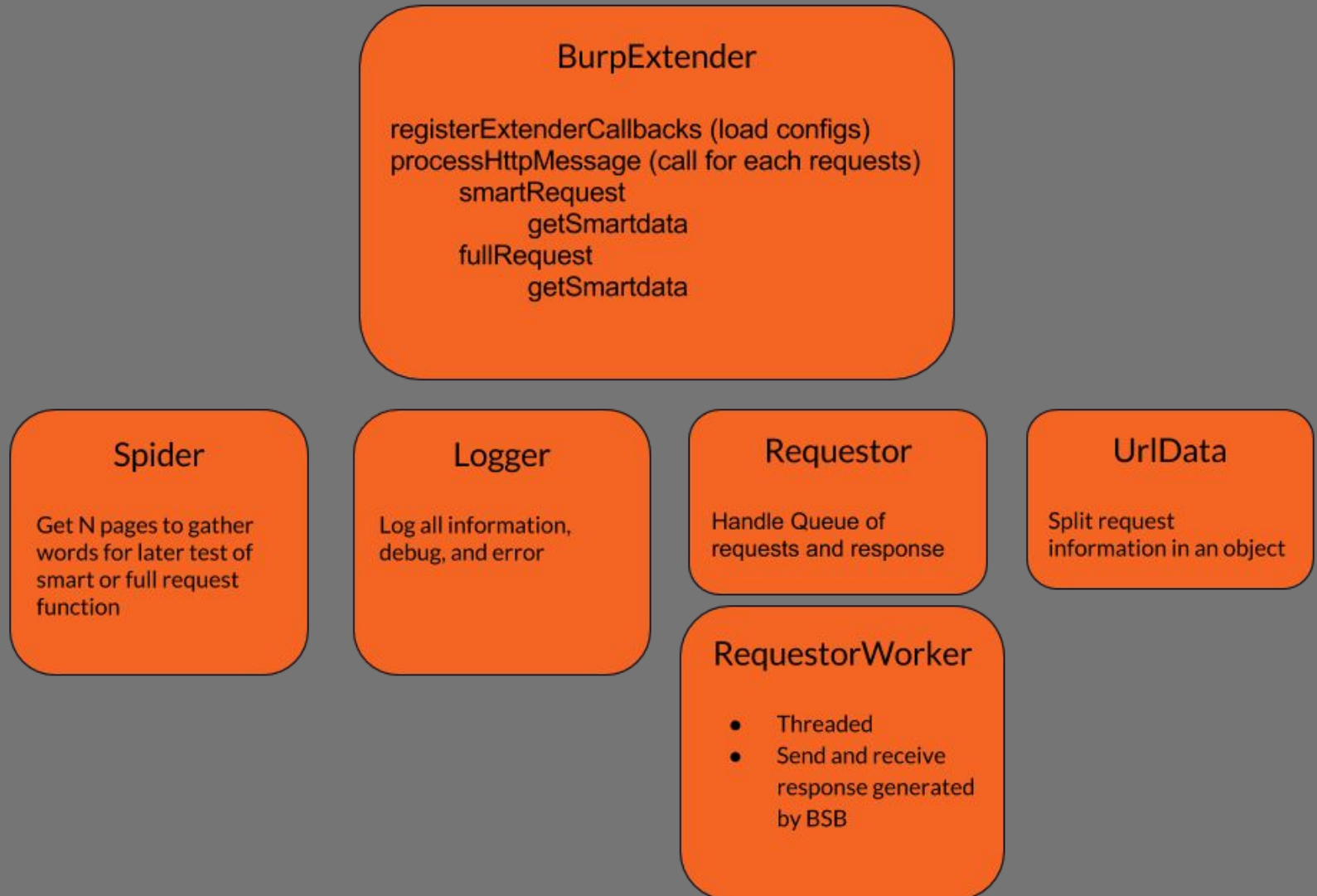
- Robots.txt
- Spider
  - Gather nouns and numbers on N pages
- Sitemap.xml
  - N URL visited
- Current directories
- Current file
  - Add prefix
  - Add suffix
- Current extension
  - Switch extension
  - Add extension

# BSB Features cont.

- Scan executes when you browse the website
  - Parse the request
  - Gather static file once (ex: robots.txt)
  - Test directories, files, extensions based on current request.
- Multithreaded
  - Works in Pro and Non-Pro version
- Logging API
- Bsb.ini configuration file
- Data.json scanning data

# How is it done?

## DIAGRAM



# How is it done?

- Burp instance
- **processHttpRequest()** intercept all request and response of Burp Suite
- **smartRequest()**
  - **getSmartData()** : robots.txt, spidering words, sitemap.xml and dynamic content (domain name, files, directories, etc)
  - Use **Requestor()** object to execute all requests threaded
    - List everything that is found in sitemap
    - Logs to file

# Logic

- Application context : What currently exist
  - Environment
    - Domain name
    - Existing files, extensions and directories name
    - Static files (Robots.txt, sitemap.xml)
    - Data: Nouns and number
- Application technology (TODO)
  - Check server header for technology
  - Check file extension (false positive possible)
  - Force a technology



# Smart : Focus/Stealth or not

- Configurable
  - Limit number of request (or not) by category (files, extensions, directories)
  - Limit the spidering
  - Chose smartRequest or fullRequest

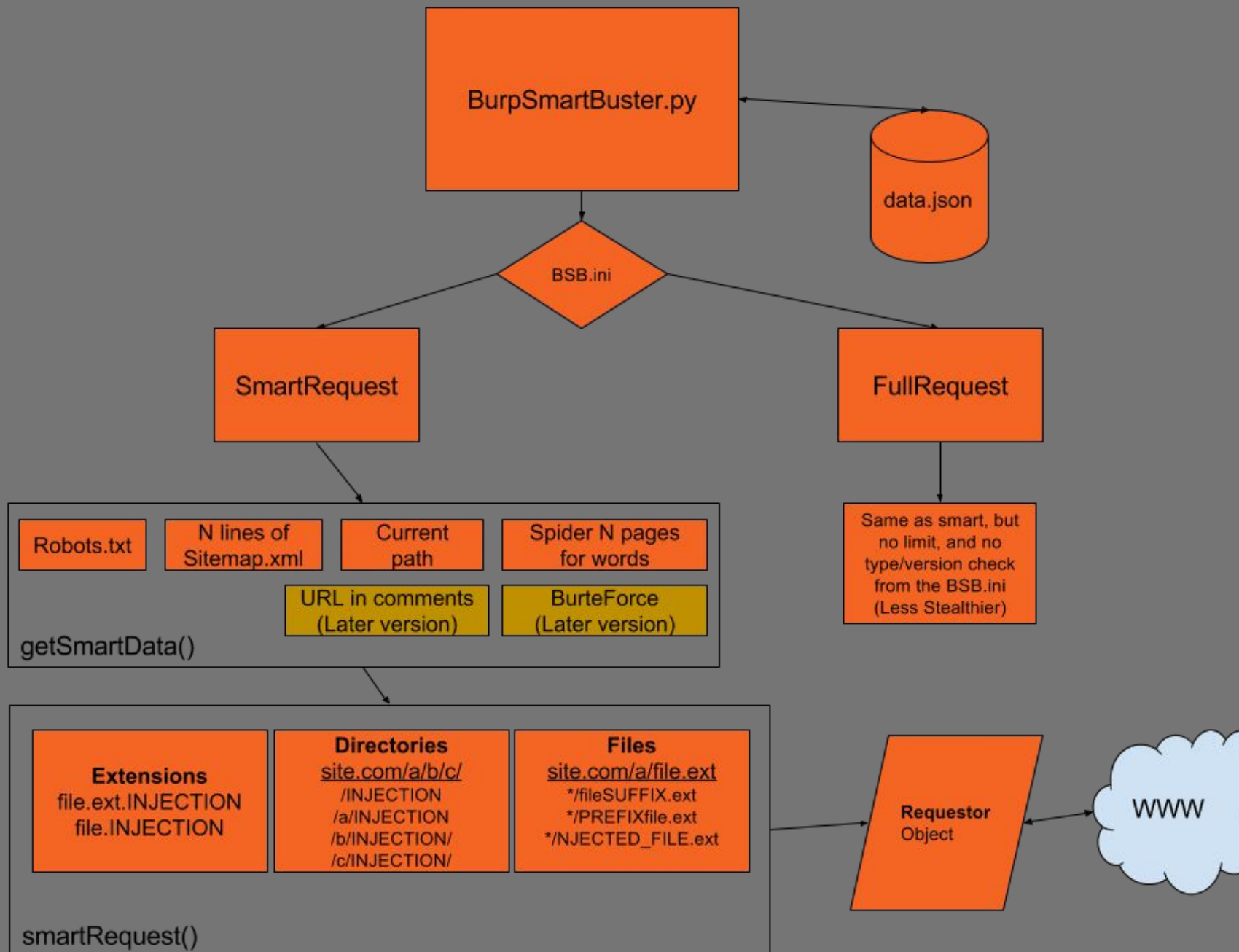


# Simple code #with comments

Python 2.7 and Straightforward

- BurpExtender
  - smartRequest()
    - getSmartData()
- Spider object (spider N pages for words)
- Requestor object (threaded web requests)
  - Queue.queue and deque
  - Add to sitemap, list if a file is found or not
- Logger, logs to file
- UrlData
  - Splitting the data from request and response of Burp Proxy intercepted by **processHttpRequest()**

# DIAGRAM



—

B

S

B

.

i

n

i

[**NumberOfTests**]

Paths: 5

Files: 5

Extensions: 5

Directories: 5

[**Spider**]

RecursiveDirs: 3

NumberOfPages: 5

[**Smart**]

Local: off

Smart: on

File: off

Spider: off

[**InScope**]

ScopeOnly: on

[**Ignore**]

FileType: gif,jpg,png,css,js,ico

[**Technical**]

TrailingSlash: on

### [NumberOfTests]

Paths: 5 → limit the number of path to look in

Files: 5 → define the number of files to test

Extensions: 5 → define the number of extensions to test

Directories: 5 → define the number of directories to test

### [Spider]

RecursiveDirs: 3 → define the recursive level for testing when we find a directory (todo)

NumberOfPages: 5 → define the number of page to spider to gather words and number from it

### [Smart]

Local: off → Only use data.json (todo)

Smart: on → Use smart information (robots.txt, sitemap.xml, spidering words, current path and files)

File: off → user file (todo)

Spider: off → spider only (todo)

### [InScope]

ScopeOnly: on → execute our test only on request which are defined in scope

### [Ignore]

FileType: gif,jpg,png,css,js,ico → ignore those filetype (todo, basically, done by Burp now)

### [Technical]

TrailingSlash: on → For a / at the end of a directory that we test.

—  
D  
A  
T  
A  
.  
j  
s  
o  
n

## Extensions

```
{ "name": ".zip", "description": "compress", "type": "default" },  
{ "name": ".bak", "description": "backup", "type": "default" },  
{ "name": ".swp", "description": "autosave", "type": "default" },  
{ "name": ".old", "description": "old", "type": "default" },  
{ "name": ".phps", "description": "development", "type": "default" }
```

## Fileprefix

```
{ "name": "~", "description": "backup", "type": "default" },  
{ "name": ".", "description": "backup", "type": "default" },  
{ "name": "Old_", "description": "old", "type": "default" },  
{ "name": "old_", "description": "old", "type": "default" },  
{ "name": "Copy%20of%20", "description": "copy", "type": "default" }
```

## Filesuffix

```
{ "name": "%20-%20Copy", "description": "copy", "type": "default" },  
{ "name": "(1)", "description": "copy", "type": "default" },  
{ "name": "%20-%20Copy", "description": "copy", "type": "default" },  
{ "name": "%20copy", "description": "copy", "type": "default" }
```

## Files

```
{ "name": "web.config", "description": "config", "type": "default" },  
{ "name": "wp-config.php", "description": "config", "type": "default" },  
{ "name": ".git/HEAD", "description": "repository", "type": "git" }
```

## Directories

```
{ "name": "config", "description": "config", "type": "default" },  
{ "name": "dump", "description": "privacy", "type": "default" },  
{ "name": "private", "description": "privacy", "type": "default" }
```

## Extensions

Extensions are used on all files that are intercept by BurpSuite.

- We test file by adding extension to them (File.ext → File.ext.ourExt)
- We test file by changing the extension to them (File.ext → File.ourExt)

## Fileprefix

Each request that is a file being intercept we will add our own prefix to the file and test if it exists.

File.ext → PrefixFile.ext

Example: Copy%20of%20File.ext → Copy of File.ext

## Filesuffix

Each request that is a file being intercept we will add our own suffix to the file and test if it exists.

File.ext → FileSuffix.ext

Example: File%20-%20Copy.ext → File - Copy.ext

## Files

All directories being intercept will be test with our files.

http://site.com/a/b/c/d/File.ext, we will test : /files\*, /a/files\* , /b/files\*, /c/files\*, /d/files\*

## Directories

All directories being intercept will be test with our directories.

http://site.com/a/b/c/d/File.ext, we will test: /directories\* /a/directories\* , /b/directories\* , /c/directories\* , /d/fildirectorieses\*



# R E S U L T S

[VERB0SE]: RequestFileExt for: http://localhost/index.html.inl  
[VERB0SE]: RequestExt for: http://localhost/index.pem  
[VERB0SE]: RequestFileExt for: http://localhost/index.html.pem  
[VERB0SE]: RequestExt for: http://localhost/index.dev  
[VERB0SE]: RequestFileExt for: http://localhost/index.html.dev  
[VERB0SE]: RequestExt for: http://localhost/index.phps  
[VERB0SE]: RequestFileExt for: http://localhost/index.html.phps  
[VERB0SE]: RequestPrefix for: http://localhost/~index.html  
[VERB0SE]: RequestPrefix for: http://localhost/.index.html  
[VERB0SE]: RequestPrefix for: http://localhost/Old\_index.html  
[VERB0SE]: RequestPrefix for: http://localhost/old\_index.html  
[VERB0SE]: RequestPrefix for: http://localhost/Copy%20of%20index.html  
[VERB0SE]: RequestSuffix for: http://localhost/index~.html  
[VERB0SE]: RequestSuffix for: http://localhost/index\_old.html  
[VERB0SE]: RequestSuffix for: http://localhost/index\_old.html  
[VERB0SE]: RequestSuffix for: http://localhost/index%20-%20Copy.html  
[VERB0SE]: RequestSuffix for: http://localhost/index(1).html  
[VERB0SE]: RequestSuffix for: http://localhost/index(2).html  
[VERB0SE]: RequestSuffix for: http://localhost/index(3).html  
[VERB0SE]: RequestSuffix for: http://localhost/index(4).html  
[VERB0SE]: RequestSuffix for: http://localhost/index(copy%201).html  
[VERB0SE]: RequestSuffix for: http://localhost/index(copy%202).html  
[VERB0SE]: RequestSuffix for: http://localhost/index(copy%203).html  
[VERB0SE]: RequestSuffix for: http://localhost/index(copy%204).html  
[VERB0SE]: RequestSuffix for: http://localhost/index(copy%201)(copy%201).html  
[VERB0SE]: RequestSuffix for: http://localhost/index%20-%20Copy.html  
[VERB0SE]: RequestSuffix for: http://localhost/index%20copy.html  
[VERB0SE]: RequestSuffix for: http://localhost/index%20(1).html  
[URL EXISTS](403) http://localhost/localhost.phps  
[URL EXISTS](403) http://localhost/.htaccess  
[URL EXISTS](403) http://localhost/.htpasswd  
[URL EXISTS](200) http://localhost/index.html.bak  
[URL EXISTS](200) http://localhost/index.bak  
[URL EXISTS](200) http://localhost/Copy%20of%20index.html  
[URL EXISTS](403) http://localhost/index.phps  
[URL EXISTS](403) http://localhost/index.html.phps



# Next Steps

1. Technology checks
2. Confidential community data of data found
3. Thread speed
4. Complete Spidering nouns and numbers
5. Technical fixes and add-on
  - a. Add the 404 checks



# 404 verification

```
self._error404[domain] = True
errorQueue = Queue.Queue(0)
code = 404
```

```
#get a 404 page
m = hashlib.md5()
m.update(str(random.random()))
```

```
url = data.getBaseUrl()+"/"+m.hexdigest()
print url
self.runRequest(url,errorQueue)
response = errorQueue.get()
```

```
#if website use standard 404 error, everything is good
if response.status_code == 404:
    return
```

```
#if website used a 30x code
if 310 - response.status_code < 11 and 310 - response.status_code > 0:
    code = 300
```

```
#if website use a 200
if response.status_code == 200:
```

```
    soup = BeautifulSoup(response.content, "html.parser")
```

```
    if soup.findAll(text=re.compile("page not found")):
        code = "page not found"
    elif soup.findAll(text=re.compile("404")):
        code = "404"
```

# Next Steps - Technology

- Add a web technology check
  - Find server type
  - Find language
  - Find platform

## Pseudo code:

If server == "apache":

    ApacheChecks(); //manual, cgi-bin, etc.

If platform == "SharePoint":

    SharePointChecks(); //check for web SP services.

If language == "PHP":

    PHPChecks(); //all files verified will be tested with .php extension

# Next Steps - Community Data

Have an anonymous web service getting data from users to share which file/extension/directories has been found.

# Next Steps - Technical fixes

- Add Thread speed and delay
- Complete the feature of spidering by using nouns and numbers gathered by the spider
- Add a GUI (arg)
  - Options (bsb.ini)
  - Box to show and save results
- Your ideas!?!





**HACKFEST.ca**

Visit beautiful  
Quebec City, Canada

Registration

<https://hackfest.ca/en/registration>

**220+ persons**  
**on-site CTF**

<https://hackfest.ca/en/ctf>



# Patrick Mathieu

**Senior security consultant**

SecurityCompass.com

**Cofounder**

Hackfest.ca

- **Email**  
[patrick@hackfest.ca](mailto:patrick@hackfest.ca)
- **Twitter**  
@PathetiQ
- **Hackfest**  
<https://hackfest.ca>
- **GitHub**
- <https://github.com/pathetiQ/>