DirBuster & Beyond

James Fisher
DirBuster Project Lead
james@sittinglittleduck.com

OWASP
DirBuster Project

The OWASP Foundation
http://www.owasp.org

Who is this James Fisher?

- By Day – Senior Constant at Portcullis
- By Night – DirBuster Project Lead

Overview


Java


COLLECTOR'S EDITION
COLLEGE INVASION
10
4


24 HOURS
1Hour Photo
minute clinic

What is DirBuster

- A multi thread Java application
- Designed to brute force files and directories on web/application servers

Can’t you just do that with simple code?

```
open (LIST, "$list") or die "Unable to open list;
foreach $name (<LIST>)
{
  $connection = IO::Socket::INET->new (Proto => "tcp", PeerAddr => "$host",
            PeerPort => "$port",
            ) or die "Can't CONNECT to $host on the Port specified.\n";

      $connection -> autoflush;
      chomp $name;
      print $connection "GET /$name/ HTTP/1.1\nHost: $host\n\n\n";
      $results = <$connection>;
      if ( $results =~ /($num)/g )
      {
            #do nothing
      }
      else
      {
            print "$results\n\n";
      }
}
```

Can’t you just do that with simple code?

- Yes the code will work

- But it won't be very good at the job

Why is it no good?

- Consider the following cases:
  - HTTPS
  - Directories that return 403 for everything, even if other dir's return 404,s
  - Speed
  - Servers that return 200's for 404's
  - If you have to use a proxy
  - Recursive scanning into dir's found
  - Error handling
  - Basic/Digest/NTLM auth

So how has DirBuster solved these issues?

- Directories that return 403 for everything
  - Checks EVERY dir and file type within EVERY dir to see how they handle failed attempts
  - Eg http://127.0.0.1/thereIsNoWayThat-You-CanBeThere/
  - Behaviour changes depends on the result of this test
- Speed
  - Utilises the Apache Httpclient API
  - Using “keep alive's”
  - Auto switching between HEAD and GET requests
  - Multi threaded producer consumer model

So how has DirBuster solved these issues?

- Servers that return 200's for 404's
  - Type 1: Static
    - The 200 response does not change (quite rare!!!!)
    - Easy to deal with
  - Type 2: Variable
    - The response is different each time
      - Dates
      - Random numbers
      - Displaying what was requested
    - Harder to deal with but not impossible

Type 2: Variable

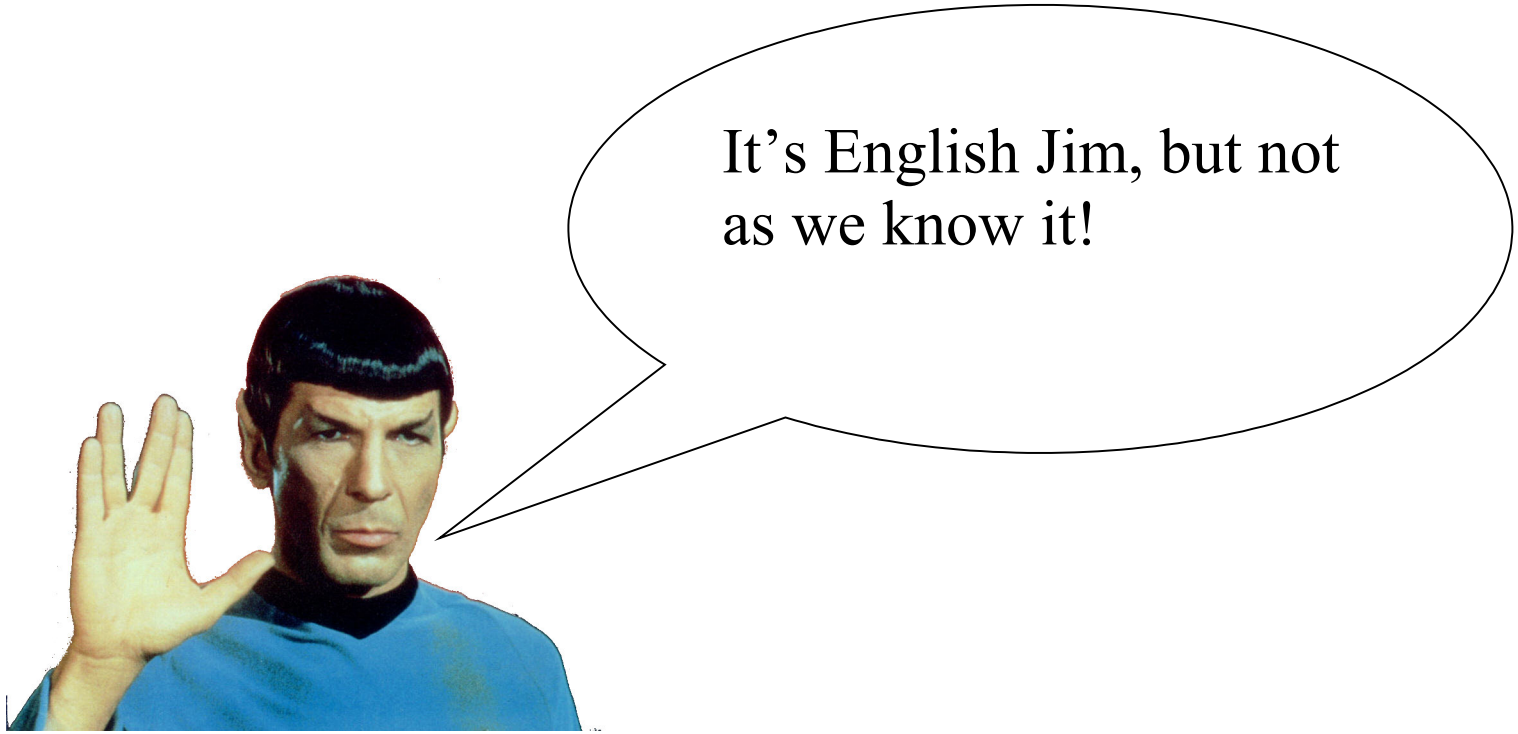
- Two approaches to deal with this
  - Content analysis mode
    - Performs a string comparison against the fail case
    - BUT only after the response has been normalised to remove things like dates, timestamps etc....
  - Regex over ride
    - Allows the user to specify a regex which if matched will count as a fail
    - Only triggers when DirBuster works out that the normalisation has failed to produce a consistent fail case

Only as good as your lists

- You can have the best scanning platform known to man
- BUT if the list it uses only has 10 items.....
- It probably won't find much!!

DirBuster lists

- Based on the concept that developers speak "developerengrilish"



DirBuster Lists

- Produced by crawling the internet
  - Custom spider written for this purpose
- Ordered by the frequency found
  - Based on the number of different hosts an item was found on
- Extensive testing to remove spam and problem dirs that sneaked their way in

pr0n

- Yep the internet is full of it
- Thus the spider found it
- So it's in the lists, as it's actually used on the internet
- Remember…

Especially your business reputation!

Demo time

What next for DirBuster

- New lists ☺
  - Crawling even more sites than last time. I would like to do 10+ million pages.
  - Collect other information that will be useful for testing
    - All get and post variable names
    - All get and post pre populated values
    - All file extensions used
    - All subdomians names used
    - All cookie names
    - Cookie values would be stupid!
  - The information can be used for other forms of testing especially fuzzing

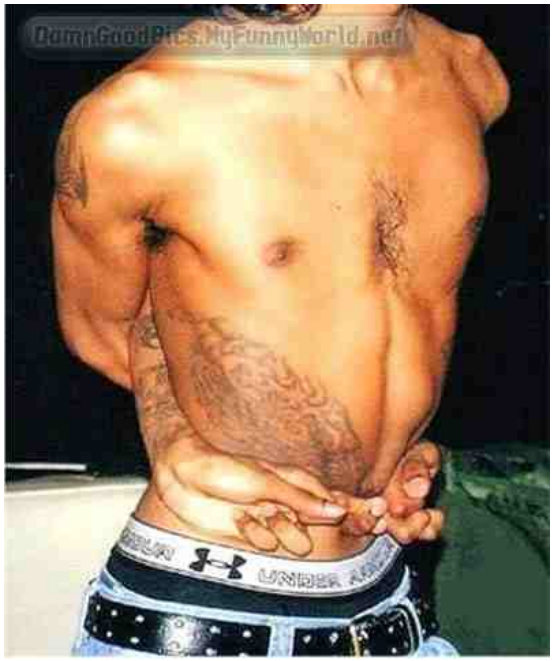
Introducing FuzzBuster

- Yep you guessed it's a fuzzer
- Not another fuzzer I hear you say
- Born out the fact I could find an open source http fuzzer to work how I wanted it to work.

Concept

- To cover 90% of fuzzing cases out of the box
- But still be flexible to deal with the other 10%
- Nice GUI, that is intuitive to use
- Plus some other features I haven't seen in other http fuzzers

Keeping it flexible



- Independent Fuzz generators
- Uses the full power of Java
- No need to create our own or use a scripting language
- Dynamically Compiles and loads the Fuzz Generators
- Only limited to what you cab do with Java

Keeping it simple

- Based on the fuzz generator, FuzzBuster will dynamically build the GUI for it
- A bit like a Metaspolit module
- So Fuzz generators can be written to take user input.

New Features

- HTTP is now a synchronous protocol
- CSRF protection & view state for example!
- FuzzBuster can fuzz over multiple requests
- Regex rules allow you to extract data from one response and use it in the next request

Demo Time

Summary

- I hope will find DirBuster useful
- FuzzBuster might be released in the next couple of months
- I have a major bug to solve first
- If you have any suggestions for either tool, please let me know!

Questions