Inside a Black Hole

By **Gabor Szappanos**, Principal Researcher, SophosLabs

Introduction

Without exception the most actively deployed exploit kit in the past year was the Blackhole exploit kit. [1]. Now that the much heralded 2.0 version of the kit is out [2], it is safe to gradually release information about the previous 1.x version. The first portion of this paper will concentrate on the stolen 1.0.2 version of the exploit kit.

A more comprehensive version of this material was published in the October issue of the Virus Bulletin magazine [3].

History of the Blackhole kit

The Blackhole exploit kit has been regularly updated in the past two years:

Version	***Release date***
2.0	09/2012
1.2.5	07/30/2012
1.2.4	07/11/2012
1.2.3	03/28/2012
1.2.2	02/26/2012
1.2.1	12/09/2011
1.2.0	11/09/2011
1.1.0	06/26/2011
1.0.2	11/20/2010
1.0.0 beta	08/2010

Version 1.0.2 of the kit has the drawback of being dated (more than two years as of this writing), which is compensated by the overwhelming advantage of being available. None of the later versions is known to be available in wider circulation (e.g., wider than its author and the purchasers). Structurally the code did not change all that much, however, the 1.0.2 version provided valuable insight into the anatomy of current attacks. In fact, very few characteristics were observed in the current attacks that featured more than what 1.0.2 architecture could service.

The Blackhole backend is available for buy or rent from the author(s) with a very flexible pricing scheme [2]:

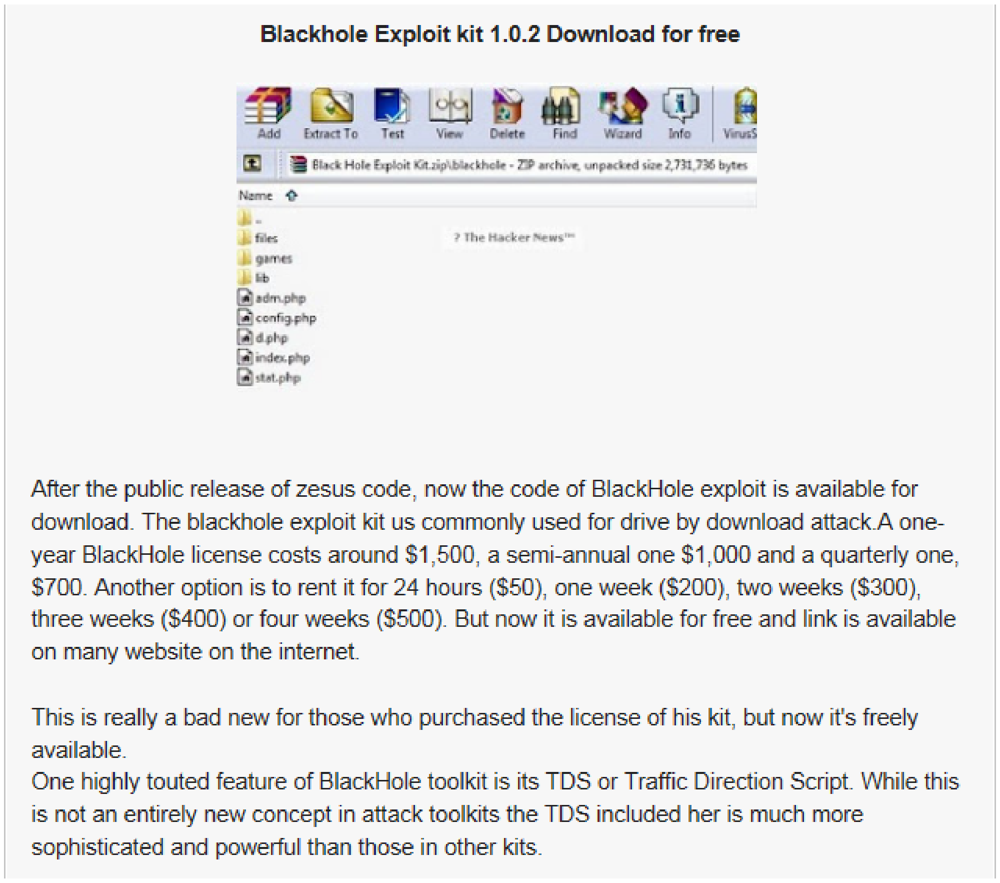
```
Annual license: $ 1500
Half-year license: $ 1000
3-month license: $ 700

Update cryptor $ 50
```

```
Changing domain $ 20 multidomain $ 200 to license.
During the term of the license all the updates are free.

Rent on our server:

1 week (7 full days): $ 200
2 weeks (14 full days): $ 300
3 weeks (21 full day): $ 400
4 weeks (31 full day): $ 500
24-hour test: $ 50
There is restriction on the volume of incoming traffic to a leasehold system, depending
on the time of the contract.

Providing our proper domain included. The subsequent change of the domain: $ 35
No longer any hidden fees, rental includes full support for the duration of the
contract.
```

Since announcing the 1.0.0 version a lot has changed in the world but the pricing remained the same [7]. With tongue decidedly in cheek it is reassuring to find that there are at least some things that keep their value.

Server side components of the exploit kits are usually hard to obtain. Occasionally law enforcement bodies can seize the C&C servers including the installed software, but these sources are not likely to surface for general availability.

For this reason it was surprising to see that in May 2011 (around the 22th, which is the earliest report [5]) a leaked version of the Blackhole exploit kit appeared on underground forums and torrent sites.



Blackhole Exploit kit 1.0.2 Download for free

Black Hole Exploit Kit.zip\blackhole - ZIP archive, unpacked size 2,731,736 bytes

..
files
games
lib
adm.php
config.php
d.php
index.php
stat.php

After the public release of zesus code, now the code of BlackHole exploit is available for download. The blackhole exploit kit us commonly used for drive by download attack.A one-year BlackHole license costs around $1,500, a semi-annual one $1,000 and a quarterly one, $700. Another option is to rent it for 24 hours ($50), one week ($200), two weeks ($300), three weeks ($400) or four weeks ($500). But now it is available for free and link is available on many website on the internet.

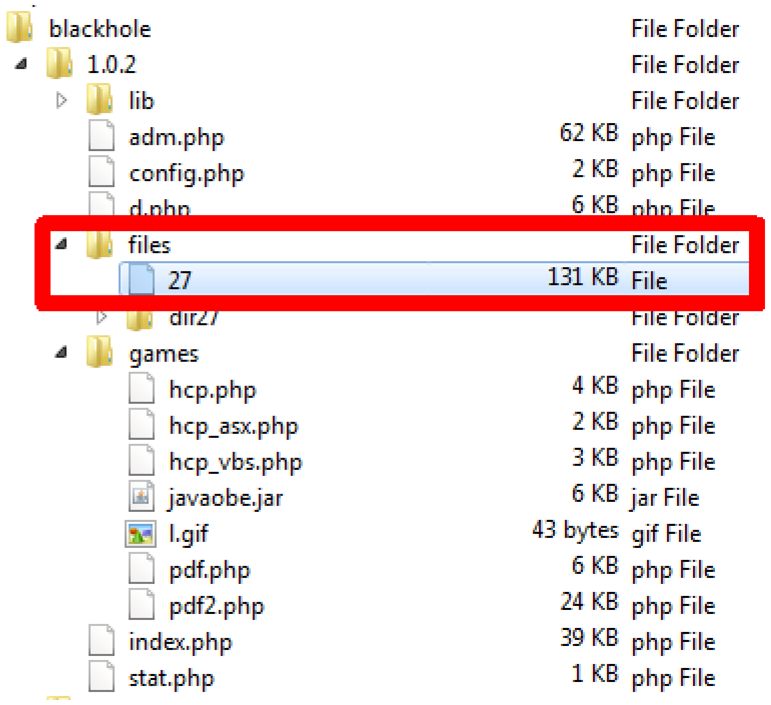
This is really a bad new for those who purchased the license of his kit, but now it's freely available.
One highly touted feature of BlackHole toolkit is its TDS or Traffic Direction Script. While this is not an entirely new concept in attack toolkits the TDS included her is much more sophisticated and powerful than those in other kits.

Security experts speculated that modifications of the source code could lead to an offspring of alternative exploit kits [3]. Similar follow-up events were observed when the source codes of some infamous malware families (e.g. ,Agobot, SdBot, Zeus) became available – resulting in an immediate explosion in new variants.

Fortunately, this did not happen in the case of the exploit kits mainly because the source code was protected with the ionCube PHP encoder [1].

Getting the source

There is no firsthand information available to answer how the source code was obtained, but a reasonable explanation is possible using the information fragments gathered following a careful examination of the source package.



```
blackhole                                File Folder
  1.0.2                                  File Folder
    lib                                  File Folder
    adm.php                      62 KB   php File
    config.php                    2 KB   php File
    d.php                         6 KB   php File
    files                                File Folder
      27                        131 KB   File
      dir27                              File Folder
    games                                File Folder
      hcp.php                     4 KB   php File
      hcp_asx.php                 2 KB   php File
      hcp_vbs.php                 3 KB   php File
      javaobe.jar                 6 KB   jar File
      l.gif                   43 bytes   gif File
      pdf.php                     6 KB   php File
      pdf2.php                   24 KB   php File
    index.php                    39 KB   php File
    stat.php                      1 KB   php File
```

The leading clue was the file named *27* in the file *upload* directory. This is the location where new executable payloads are uploaded for further distribution to the infected endpoints.

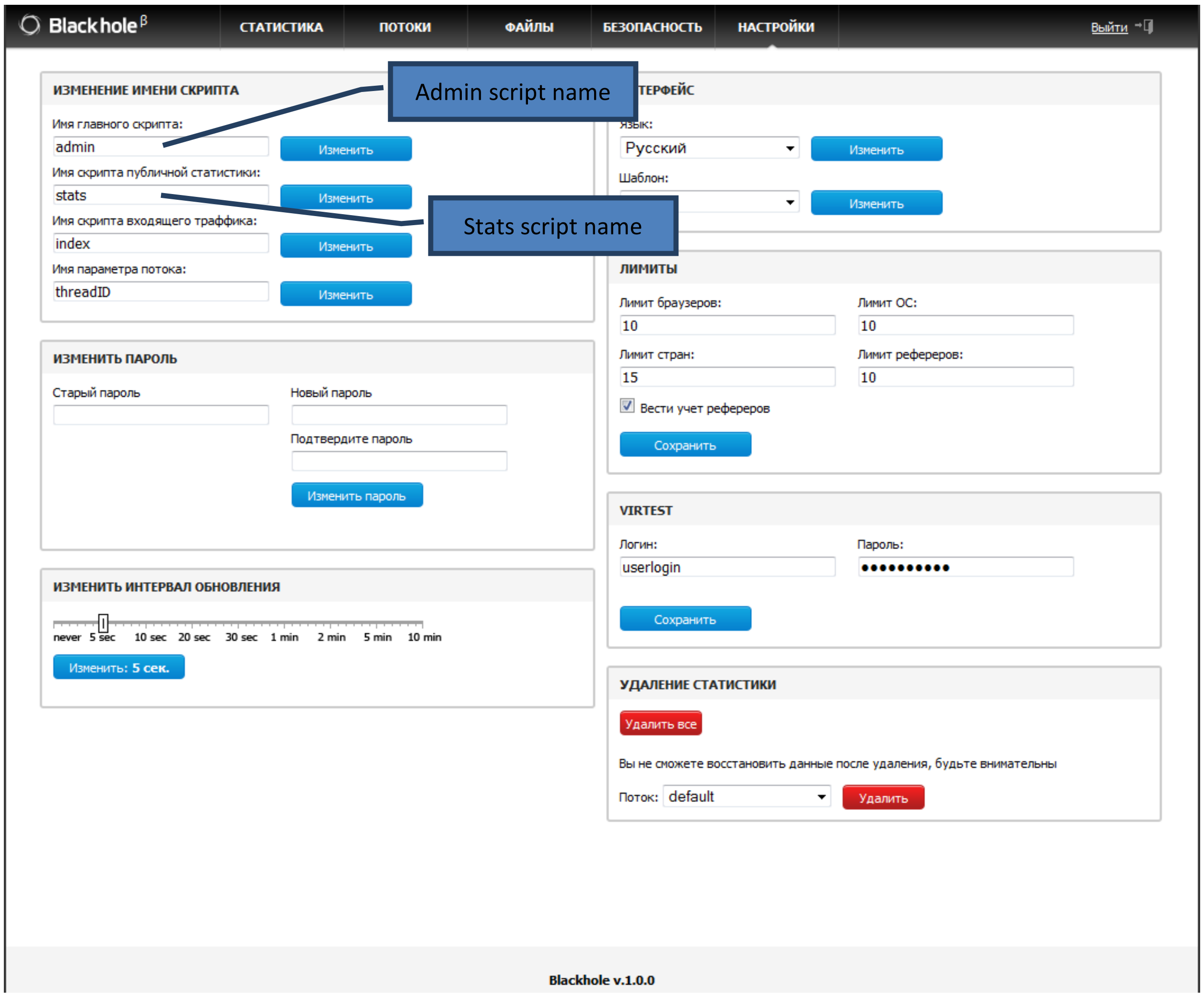
This file was not something one would generally expect to find there: not the usual botnet executable or keylogger installer, which are generally observed as payloads, but a copy of the infamous *C99Shell* backdoor, which is the most popular tool of choice for hacking into websites. This file is a foreign body in the server compound. It can't be a payload, because

- Blackhole can deliver only binary executable payloads by design
- It is not referenced anywhere in the server code
- Coding style is different

- Not protected with ionCube

The only logical conclusion is that *C99shell* was used to hack into the server and gain access to the files. There was an initial hurdle to jump before reaching to this point. The upload functionality is only available via admin authentication. How was this problem overcome?

Screen shorts of the 1.0.0 version admin panel were available on the web, which provided the path of the admin panel (/*adm.php*).



ИЗМЕНЕНИЕ ИМЕНИ СКРИПТА

Имя главного скрипта:	admin
Имя скрипта публичной статистики:	stats
Имя скрипта входящего траффика:	index
Имя параметра потока:	threadID

ИЗМЕНИТЬ ПАРОЛЬ

Старый пароль

Новый пароль

Подтвердите пароль

ИЗМЕНИТЬ ИНТЕРВАЛ ОБНОВЛЕНИЯ

never 5 sec 10 sec 20 sec 30 sec 1 min 2 min 5 min 10 min

ТЕРФЕЙС

Язык: Русский

Шаблон:

ЛИМИТЫ

Лимит браузеров: 10	Лимит ОС: 10
Лимит стран: 15	Лимит рефереров: 10

☑ Вести учет рефереров

VIRTEST

Логин: userlogin

Пароль: ●●●●●●●●●●

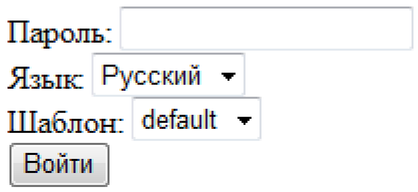
УДАЛЕНИЕ СТАТИСТИКИ

Вы не сможете восстановить данные после удаления, будьте внимательны

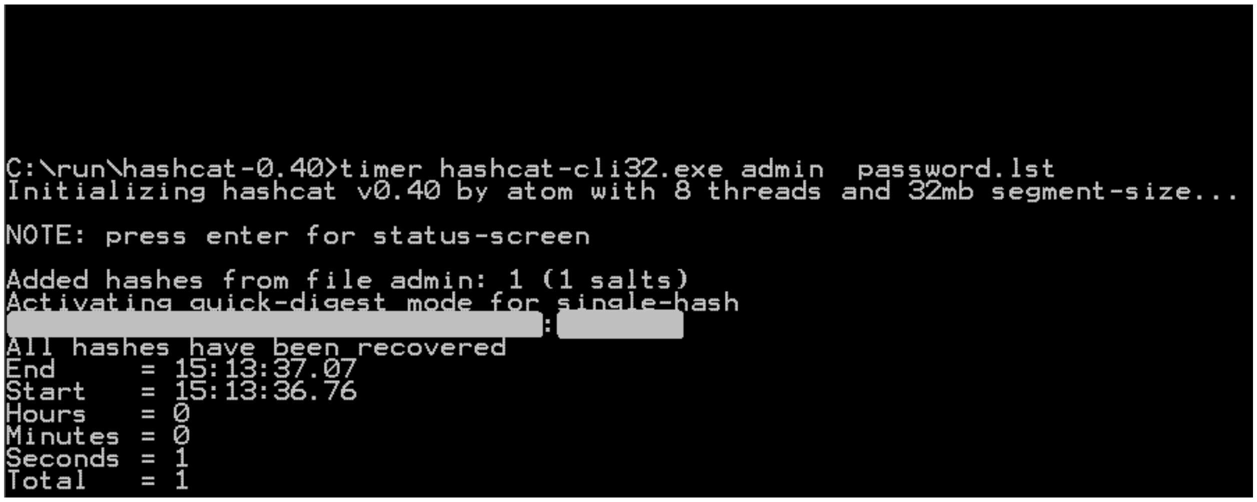
Поток: default

Blackhole v.1.0.0

Now all that was left was to log in. This was not as difficult as one would expect In fact, guessing the admin password turned out to be less difficult than dealing with the initial Russian login interface that the original stealer had to face.


Пароль:
Язык: Русский
Шаблон: default
Войти

The *config.php* file contains among many other values the md5 hash of the admin password. It is considered to be a safe approach to store only the one-way hash of the password (though even in that case md5 is not the advised choice for the hash algorithm), and on login the calculated hash of the submitted password is compared with the stored hash. What should not be considered to be safe under any circumstances was the password itself. To illustrate its weakness, it is enough to note that a dictionary attack using a common password cracking tool and an ever more common password list could break the password in about 310 milliseconds.



```
C:\run\hashcat-0.40>timer hashcat-cli32.exe admin  password.lst
Initializing hashcat v0.40 by atom with 8 threads and 32mb segment-size...

NOTE: press enter for status-screen

Added hashes from file admin: 1 (1 salts)
Activating quick-digest mode for single-hash
[redacted]:[redacted]
All hashes have been recovered
End     = 15:13:37.07
Start   = 15:13:36.76
Hours   = 0
Minutes = 0
Seconds = 1
Total   = 1
```

Not surprisingly the password used for the admin interface can be found in almost every weak password list available on the Internet, and even then right near the top.

So my educated guess for the method obtaining the source code resulted from the following process:

- The attacker identified a Blackhole attack, then traffic or static analysis lead to the C&C server.
- Gained access to the admin interface in about 5 to 50 tries.
- Uploaded the *C99Shell* file
- Opened it in a browser
- Access granted to the files within the Blackhole home directory.

One piece of advice: if you maintain a C&C server (which is both a really bad idea and a criminal act and as such, strongly discouraged), always use a strong password. It's very unprofessional if your server is cracked this easily. Gradually the Blackhole operators learned this lesson, although it was a long process. A year later strikingly similar passwords still worked on some of their servers.

There is also a lead regarding where this particular server was originally located. Blackhole kits use (among others) Java components for downloading the Win32 binaries, and these Java components were linked into the HTML page returned by the server during the attack. The actual usage had two main patterns: either referenced with full URL or the relative path within the server home. In all analyzed cases the JAR was linked from the same server that hosted the exploit kit, never referenced content on an external server.

In the specific server setup (defined in *config.php*) the path to this component was set to *195.80.151.59\pub\new.avi*. Despite the .avi extension, this file was not a media file, but a JAR archive.

This leads to the conclusion that the cracked server had to be 195.80.151.59.

This IP was known to host various malware back in February 2011 [8], then under domain name tuqidig5.co.cc (and a few others, like dubezov3.co.cc, gube2qome8.cz.cc, cepepeler28.co.cc, dofubuhud57.co.cc), registered to a company located in Belize. Nowadays it belongs to a Polish ISP, unrelated to malware.

About ionCube…

Most of the server backend code is encrypted with the commercial ionCube encoder [1], which is one of the most popular PHP encryptors – and if we are speaking about exploit kits, it is in fact the most popular.

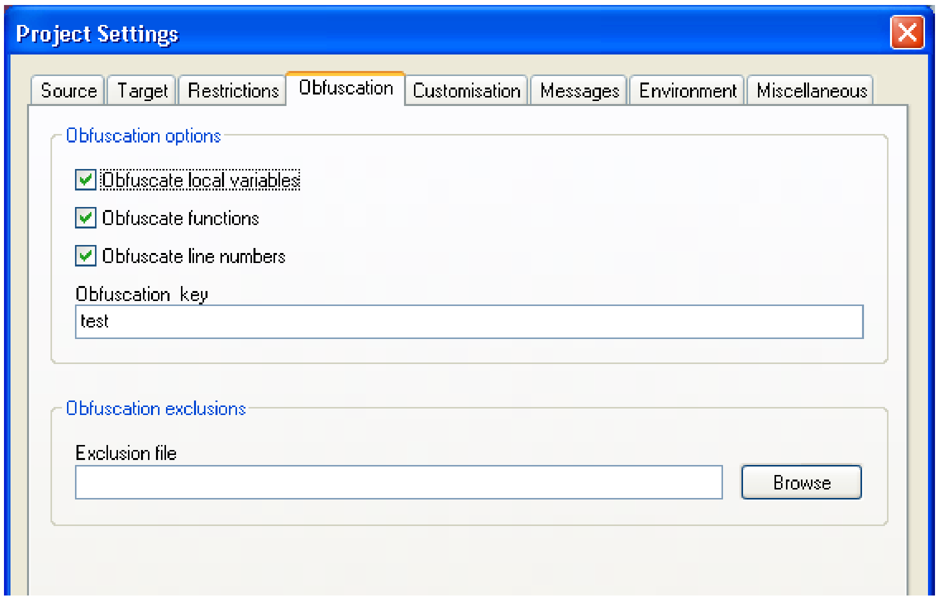
It has a rich set of features, including [6]:

- Encoding PHP scripts with compiled byte codes for accelerated runtime performance and maximum security.
- Obfuscation of byte codes after compilation for extra security.
- Selectable ASCII or Binary encoded file format.
- Prevention of file tampering through use of digital signatures.
- Prevention of unauthorized files from including encoded files.
- Generating files to expire on a given date or after a time period.
- Restricting files to run on any combination of IP addresses and/or server names.
- Restricting files to run on specific MAC addresses.
- Customized messages when files expire or aren't permissioned to run.
- Fast encoding.

The obfuscation of byte codes includes a few methods to protect from reverse engineering:

- Obfuscation of local variables
- Obfuscation of function names
- Obfuscation of line numbers

Of those the obfuscation of function names has the most devastating effect on decrypted code readability.


Project Settings

Source | Target | Restrictions | Obfuscation | Customisation | Messages | Environment | Miscellaneous

Obfuscation options

- [✓] Obfuscate local variables
- [✓] Obfuscate functions
- [✓] Obfuscate line numbers

Obfuscation key: test

Obfuscation exclusions

Exclusion file:

The popularity of the cryptor is reflected by its' prevalence among the exploit kits. Going through a representative collection of 55 different exploit kits only nine of them were protected with any kind of PHP encryption, and six of those used the ominous ionCube.

Exploit kit	***Cryptor used***
Adrenalin	Zend
Blackhole	ionCube
Bleeding life	ionCube
Crimepack	ionCube
Intoxicated	ionCube
Liberty	Php Express
Pay0c	ionCube+custom
Tornado	Zend
Yes	ionCube

Even the ones using *ionCube* have not benefited from the strictest security services provided by the cryptor. Table 3 summarizes the usage of restricted domains and function name obfuscation among these exploit kits.

Exploit kit	***Restricted domain count***	***Function name obfuscation***
Intoxicated	3	No
Blackhole	28	Yes
Bleeding-life-pack	2	No
Crimepack	1	No
Pay0c	N/A	N/A
Yes	-	Yes

Only *Blackhole* and *Yes* featured the function name obfuscation, which is a very effective way to protect against reverse engineering. And then *Yes* does not benefit from the domain restriction, which is a good defense against illegal use (as controversial as it sounds referring to a tool used in computer crimes) on unauthorized servers. This table also underlines the introductory claim that Blackhole is the most widely deployed attack kit. While the examined versions of the other kits were deployed on 1 to 3 servers, *Blackhole* was licensed to use on 28!

The *ionCube Encoder* converts the PHP source code to byte code, encrypts it based on the selected protection settings, and prepends a short and static loader code:

```
<?php //0035e
if(!extension_loaded('ionCube
Loader')){$__oc=strtolower(substr(php_uname(),0,3));$__ln='/ioncube/ioncube_loader_'.$__
oc.'_'.substr(phpversion(),0,3).(($__oc=='win')?'.dll':'.so');$__oid=$__id=realpath(ini_
get('extension_dir'));$__here=dirname(__FILE__);if(strlen($__id)>1&&$__id[1]==':'){$__id
=str_replace('\\','/',substr($__id,2));$__here=str_replace('\\','/',substr($__here,2));}
$__rd=str_repeat('/..',substr_count($__id,'/')).$__here.'/';$__i=strlen($__rd);while($__
i--
){if($__rd[$__i]=='/'){$__lp=substr($__rd,0,$__i).$__ln;if(file_exists($__oid.$__lp)){$_
_ln=$__lp;break;}}}@dl($__ln);}else{die('The file '.__FILE__." is
corrupted.\n");}if(function_exists('_il_exec')){return _il_exec();}echo('Site error: the
file <b>'.__FILE__.'</b> requires the ionCube PHP Loader '.basename($__ln).' to be
installed by the site administrator.');exit(199);
?>
4+oV584oGn8W1xWbEOlMCSe7+5xpGsdDr0UqMyicg6oxyLZb16BluFQpCr+D7yMqMhqOmkX4yABG
UKwVZfc7Fa33Xop85AWlurw0+VnDpnXgCG9sXDOnOC9ZY839Z9t1rQ5tDwpUkxvO388zFwJnhL8t
HFJiu3BxAvnoJ7SbPDuE/J0jq1PvzQJubQ00n2i0qucXQWp4DqGIIdbqP1GoaFrwVjVK80KM9uCO
4VYWKfNPrKgeOzYLfqROaektFtx8m/TYNAwAyABKV374GJ7NzOTcbJengE6+2vmu83PjyIDH/7Y1
fAtoE+RRFefDKlnBdZzPrvtowt/281w8ZQQaFaBK/P9IqxFIg/IXH8kXIuXtPAMNPNNVhKMoiLhO
Zi3scRC8k2Ez3KQZUb5LSOjjM+hQNyrRVhjOaOstjGTYbV6DvNoQkkMZDusxcYe/I3fXTw58+nCb
w+7W5H32VXXm3juUR1SovZOqejy7Vs/DqhdL1r/+SIOSGHlw7BKZUc+Y8g9NtInkpUWBaf5r3CZF
Sq0XitNW/EZopkHyT6SNoFSXnLmEtvEINqJBrkR5zNeDutXgcZ4sp3rPZ8kTiDEQ9mgjiDleJcXp
Dfw/c6/vNnjwAcSLzzYQUwLrvC55FREiVksS
```

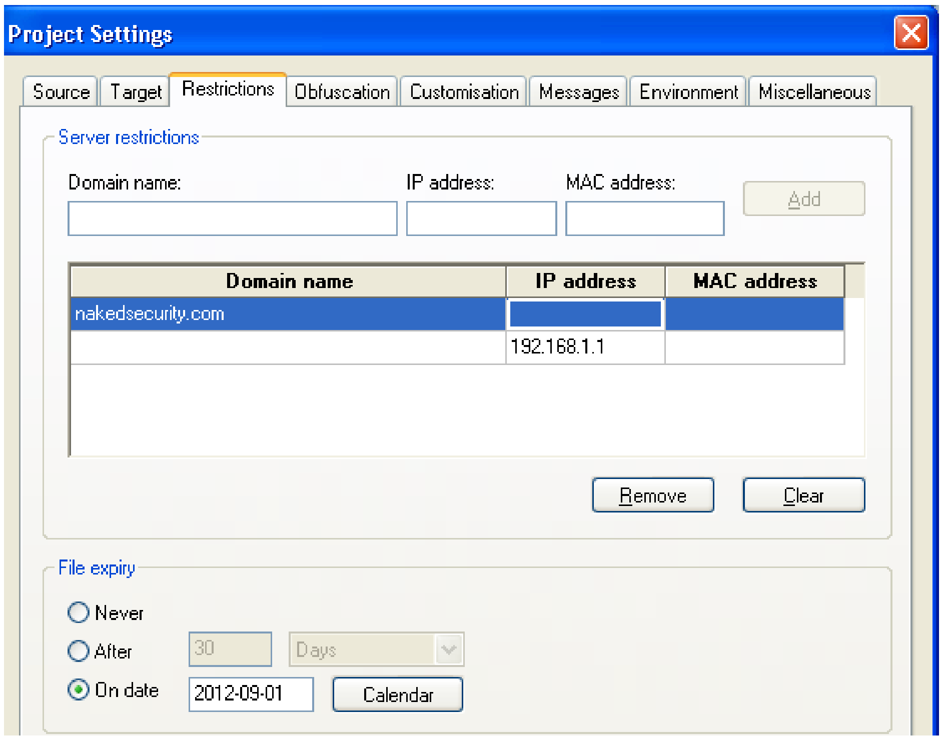
The script loader checks the availability of the *ionCube* loader, and if it is found, hands the script to the binary loader which verifies the validity of the license, and determines whether it is running on the server to whom it is licensed.

Running a protected code on an inappropriate server will result in error:

```
The encoded file C:\Program Files\EasyPHP\www\blackhole\index.php is not permissioned
for 127.0.0.1 in Unknown on line 0)
```

If all checks pass, the source decrypts and loads the byte code into the PHP interpreter.

A block of settings data is stored in the header of the decrypted block, which if interpreted, supplies useful meta-information about the exploit pack, revealing some of the settings used during the creation of the package, for example the selected obfuscation methods and the list of the server restrictions.



Server restrictions

Domain name: | IP address: | MAC address:

Domain name	IP address	MAC address
nakedsecurity.com		
	192.168.1.1	

File expiry

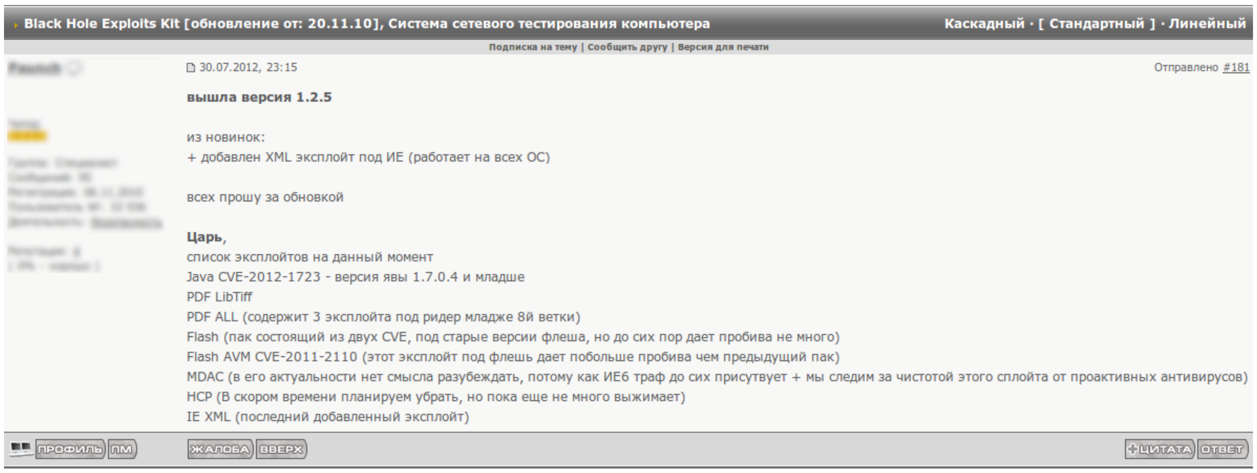
( ) Never
( ) After 30 Days
[✓ On date] 2012-09-01

Reassuringly, the IP address 195.80.151.59, which claimed to be the origin of the stolen source, was present on the server restriction list extracted from the source files.

Origins

Until only recently making a decision where a particular malware specimen originates from has never been easy. Normally we just flip a coin. If heads, then its origin is China. If tails, then Russia. If it lands on the edge, then we reasonably conclude state sponsored malware.

The first version of this kit was announced on Russian underground forums, and the author claims to be Russian, which may already be enough evidence for some but it is always good to support the claim with facts based on analysis of the code that are not as trivial to fake as a forum comment.


Каскадный · [Стандартный] · Линейный
Подписка на тему | Сообщить другу | Версия для печати
ПРОФИЛЬ
ПМ
ЖАЛОБА
ВВЕРХ
ЦИТАТА
ОТВЕТ

Black Hole Exploits Kit [обновление от: 20.11.10], Система сетевого тестирования компьютера

[redacted]

30.07.2012, 23:15 — Отправлено #181

вышла версия 1.2.5

из новинок:
+ добавлен XML эксплойт под ИЕ (работает на всех ОС)

всех прошу за обновкой

Царь,
список эксплойтов на данный момент
Java CVE-2012-1723 - версия явы 1.7.0.4 и младше
PDF LibTiff
PDF ALL (содержит 3 эксплойта под ридер младже 8й ветки)
Flash (пак состоящий из двух CVE, под старые версии флеша, но до сих пор дает пробива не много)
Flash AVM CVE-2011-2110 (этот эксплойт под флешь дает побольше пробива чем предыдущий пак)
MDAC (в его актуальности нет смысла разубеждать, потому как ИЕ6 траф до сих присутвует + мы следим за чистотой этого сплойта от проактивных антивирусов)
HCP (В скором времени планируем убрать, но пока еще не много выжимает)
IE XML (последний добавленный эксплойт)

The comments inside the source code would have been the most telling clues, but they were unfortunately removed when the *ionCube* was applied on the code. There were still enough traces left:

- The default time zone of the installation is hardcoded to *Europe/Moscow*.
- The user interface supports two languages, English and Russian, the default being set to Russian.
- The English user interface texts and the variable names are noticeably incorrect at places,the Russian interface texts are grammatically more correct.
- There are two character encodings supported in the code with conversion functions: UTF-8, which is a standard, and Windows-1251, which is a Cyrillic encoding.
- The date format in the code in all places is set to Little Endian date which excludes the other two usual suspects; USA uses Middle Endian while China Big Endian.

All the evidence supports the assumption that the development of the Blackhole exploit kit occurred in Russia.

To be continued…

The next part of the analysis will build on this knowledge and focuses on the operation of a Blackhole server. We will examine in detail what happens on the server side during a typical attack, what kind of interaction goes on between the infected-to-be host and the infecting hosting server.

Continue to check NakedSecurity and Sophos.com for follow-up entries.

References

[1] http://www.sophos.com/en-us/why-sophos/our-people/technical-papers/exploring-the-blackhole-exploit-kit.aspx

[2] http://nakedsecurity.sophos.com/2012/09/13/new-version-of-blackhole-exploit-kit/

[3] http://www.virusbtn.com/virusbulletin/index

[4] http://www.malwaredomainlist.com/forums/index.php?topic=4329.0

[5] http://www.infosecurity-magazine.com/view/18159/blackhole-exploit-kit-now-being-offered-for-free/

[6] http://thehackernews.com/2011/05/blackhole-exploit-kit-download.html

[7] http://www.ioncube.com/sa_encoder.php?page=features

[8] http://www.scumware.org/report/195.80.151.59