[**InSecurity In Security**]

::Seminar Topic::

InSecurity In Security

by,
Abhishek Kumar[m0727]

Guide,
Mr. Ramdas Karmali

-=ABK=-

Abstract

In world full of insecurities, the only thing we can rely on are security measures to avoid the threats... but what if the security measures themselves give way to attackers to bypass all security and rule the un-informed.
We have a world of Network full of network services, each with a protocol to follow and a security mechanism to depend upon.
Now if a service is used without implementing the recommended security mechanism, its fault of service provider &/or user for running such a vulnerable service which is open for exploitation by anyone.
But, what can a service provider and user do to protect themselves if in spite of implementing recommended security measures; they are open to be exploited by hacker who exploits the very mechanism setup to protect the system.

In this respect we will discuss the latest network based attacks based on this principle.

Contents

- ➔ **Abstract** -Pg (ii)
- ➔ **Introduction** -Pg(1)
- ➔ **Security Measures** -Pg(1)
  - ➔ O.S. Log-In Service –Pg(1)
  - ➔ Visiting Websites Securely & Hosting Secure WebServer
  - ➔ Secure Log-in to Web 2.0 services –Pg(1)
  - ➔ SSL Enabled Session –Pg(2)
  - ➔ Anonymizer/Proxies –Pg(2)
  - ➔ DNSSEC (*DNS Security Extensions*) –Pg(2)
- ➔ **InSecurity In Security** -Pg(2)
  - ➔ O.S. Log-In Hack OR Bypass –Pg(2)
  - ➔ Res-Timing Attack, SMBEnum AND Slowloris –Pg(3)
  - ➔ SideJacking –Pg(5)
  - ➔ Defeating SSL –Pg(6)
  - ➔ DeAnonymize Proxy –Pg(8)
  - ➔ DNSSEC Vulnerabilities –Pg(9)
  - ➔ FORENSiC ATTACKS –Pg(10)
- ➔ **Countermeasures** -Pg(11)
- ➔ **Conclusion** -Pg(12)
- ➔ **Reference** -Pg(13)

Introduction

Security measures get exploited by attacker mainly due to three causes:

1.) The security protocol includes steps like weak encryption, lame passwords which are computationally weak and can be broken by mechanisms like brute force, reverse engineering, etc.

2.) The security protocol itself is badly designed providing a flaw for attacker to exploit bringing the entire security mechanism to its knees and access the system

3.)The steps are strong enough, security protocol design is flawless; but the implementation is bad i.e. not according to the manner it was designed for

We shall use the following flow to understand some such vulnerabilities and preventive measures that can be employed to strengthen the security
[Security → Insecurity.In.Security → Countermeasures]

Security Measures

Common mechanism implemented to provide security to users:

1.) O.S. Log-In Service

All modern Operating Systems provide facility to create different users with different access level on any system. So, we can allow multiple users to share a computer in restricted fashion. We even sometimes leave our machines in our work places, just because we have trustworthy Log-in mechanism.

2.) Visiting Websites Securely & Hosting Secure Web Server

Access to sites with legal content and seemingly secure source is a good practice designed to keep you safe from most of web attacks launched against web application attacks.
Hosting updated Web Server, keeping a check on Denial-of-Service attacks. Implement IDS/IPS to check for DoS attacks, giving a secured Web Server.

3.) Secure Log-in to Web Application

Almost all of the Web Applications like services from Gmail to Live, Mail Clients, or any session-exchange based service implementation. Authentication is secured by log-in over encrypted channel.

4.) SSL Enabled Session

You can even use SSL-based connection after signing-in several services like Gmail, Facebook, etc. for an enhanced level of security for data exchanged during service-use against attacks like sniffing.

5.) Anonymizer/Proxies

If you don't trust your network at all, establish a Secure Proxy and use all network services over it; most famous and efficient for all services is Onion Proxy popularly implemented using Tor

6.) DNSSEC (*DNS Security Extensions*)

DNS is the base of whole Name-Based Web Service. So, the Web URIs are as Secure as DNS is. Thus, we externally implement DNSSEC over normal DNS for a strong security of our WWW Services

---

InSecurty In Security

Unfortunately as we shall see all the above mechanisms can be exploited by hacker to attack your systems.

1) O.S. Log-In Hack OR Bypass

::ACTIVE:: at least you know your account have been compromised

There are lots off O.S. Log-in hacks available to crack or remove the Authentication password (esp. for Windows).
Famous methods be:

a) we have Password Removal Discs which restore the User Account to no password stage
b) using pre-computed Attack from Rainbow Tables using tools like OPHCrack, Rainbow Table Crack tool
c) even Brute Force Attack works well after professional reconnaissance

::PASSIVE:: danger of the unknown ;)

Windows and Unix/Linux load there log-in utilities on RAM after booting of O.S.; so based on this concept studying the behavior of these log-in utilities, methods normally used are:

d) We have bootable floppies, discs and USB devices like KonBoot which on booting from them load there small piece of Overflow Attack Code onto RAM and pass on the booting to default boot-device.
At log-in screen no matter what password you type, you are logged in as

authenticated user.

e) If OS X has an auto-log-in account, just boot it holding Shift Key, timing of pressing Shift key matters for this action
f) On Unix/Linux Machines without bootloader password using Grub/LiL0, do the following steps for the cakewalk to SuperUser mode
   i) press ARROW Key when Grub/Lil0 comes
   ii) press 'e' key to edit entries, (or whatever is mentioned)
   iii) select the line starting with 'kernel' (2nd line by default)
   iv) press 'e' key to edit it to use Single User Mode
   v) [Grub]
      append 'S' or 'Single' at end of the line
      [LiL0]
      use 'Ctrl+X' to goto Console Mode, write 'linux single'
   vi) press 'b' to boot
   vii) The machine boots into Single User Mode which is in SuperUser mode and leave you at command prompt, now either run command 'startx' to start XWindows for GUI mode OR 'passwd' and change SuperUser password; choice is yours.

2) Res-Timing Attack, SMBEnum AND Slowloris

Visiting Websites Securely

::Res-Timing:: the 'res(ource)://' protocol hack with CPU Cycles

i) It's an interesting method of enum of files using IE's res:// protocol with a killer combo of XSS + CSRF *(XSS is Cross Site Scripting which is Web App running unauthorized code at client side, CSRF is Cross Site Request Forgery which is User running unauthorized command at web server).*
ii) It could actually be used to finger print a drive or detect which exploits to perform against a target.
iii) The res:// protocol is built into IE4.0 or later, typically used to pull resource like Images, HTML, XSL,... etc. from a DLLs & Executables *(take a look at IE's error page, a lot resource are pulled from ieframe.dll)*. So it's possible for a remote web page to call for a local resource URI and report to server.
   This necessitates patching of IE, and now Firefox also has similar issue with resource:// protocol.
   So an attacker can use RES URI to:
iv) Enumerate softwares on your machine (in many cases even version)
v) Use enum software list to target specific attacks & exploits

::SMBEnum:: reconnaissance via simple HTML Web Page

i) It's a normal JavaScript based attack where opening a malicious Web Page in IE can let attacker enumerate certain types of files on Victim's Machine for reconnaissance. Suppose presence of 'Oracle 8' can be confirmed by presence of 'file:///c:/oracle/ora81/bin/orclcontainer.bmp' on file which gets at this location by default while installing 'Oracle 8'. Here, attacker can add a JavaScript to Web Page which simply checks for presence of resource and confirms enumeration of 'Oracle 8' if succeeds in loading it.
   It could also gather usernames through brute-force attack.

Hosting Secure Web-Servers

::Slowloris:: slow HTTP-DoS Attack

i) This is a stealth mode denial-of-service (DoS) attack against Web Servers with threaded processes and work by limiting memory usage. Rather than flooding Network, this attack allows a single machine to take down another machine's Web Server with minimal bandwidth and lesser side-effects on unrelated services and ports.
   Normal HTTP Transaction is initiated by HTTP Client (Browsers) by connecting to HTTP Server IP at HTTP Service Port (default is 80). HTTP Server starts listening to the connection till it is completed by a HTTP Request. Then HTTP Request is responded by a HTTP Response which may be resource requested, error message, or other information.
   This attack works by opening a connection at HTTP Server and keep sending subsequent request to keep connection from closing. Slowloris waits for Server socket to become available before it's successful at consuming them. So for a high traffic site it may take some time.
   It's very stealthy as no single log entry appears till attack is on due to partial connections which after attack result into lots of 400 ERROR messages, changed even to 200 OK messages by completing the connection.
   This exploit was basically written in Perl and can't be executed on Windows machine even using CygWin (a UNIX environment emulator); for Windows users there is a Python version available of Slowloris named PyLoris.
   HTTPReady can't counter it as we can use POST (handles only GET, HEAD).
   It works successfully over Apache 1.x, Apache 2.x, dhttpd, GoAhead, WebSense, etc but fails against IIS 6.0, IIS 7.0, lighttpd, squid, nginx, etc.

3) SideJacking

SideJacking is a malicious act of hijacking an engaged web session with a remote service by intercepting and using credentials used by victim to identify to service.
HTTP is stateless protocol, thus developers use cookies to maintain user's information over a session. Cookies are plain-text files stored in Browser's cache. Web Application with secured log-in mechanism use them to store current session's authentication data to maintain access for current session. While attackers trying to sniff passwords over network are being stopped by Secured Log-in mechanisms of Web services, most of these services handle the latter part of authentication data exchange in unencrypted manner. SideJacking tool could easily sniff the accounts being accessed, over other machines on the same network.
Old tools were able to sniff session IDs from the network traffic and reconstruct cookies (or relevant session files) as persistent files, place them in browser and replay them as valid cookies to access browsers. New tool is a full GUI based proxy allowing you to use the sniffed cookies to be automatically launched in Browser and access the account of that cookie. E.g. Victim using his Gmail account on network wouldn't know and I'll be actually opening the same account at same time and have full access.

Attacker can't change password, but can still use account if the passwords are changed or actively signed out, with some tweaks.
While stealing cookies to hijack websites is quite well-understood technique and greatly used in XSS attacks. This doesn't require website vulnerabilities or victim's password to be compromised. It is as simple as *Point-N-Click* with new GUI.
[Note: XSS Attacks are discussed in detail by another Seminar by m0723]

Gmail in SSL HTTPS mode is normally safe as latter transaction is encrypted, but it's found that Gmail's JavaScript code falls back to non-encrypted HTTP mode if HTTPS mode isn't available. So, use setting to force HTTPS mode in Gmail Setting.
Microsoft's Hotmail and Yahoo!'s Mail service don't have SSL modes, so no expectation or tweaks to make them secure from SideJacking.

This could even be performed by a simple Network-Sniffer and editing cookies in Firefox using its add-on 'edit Cookie'.

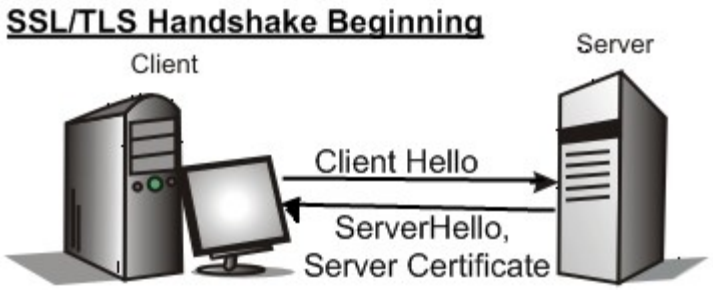
Some Website's require URLs to set the cookie correctly, so if feel inconsistent log-in privileges, browse few URLs on the site so that cookie may adjust its values for current browser.

4) Defeating SSL

SSL, most commonly used Security Mechanisms has been faulty at both design and implementation.

Earlier, SSL had a design flaw which allowed any Digital Certificate to verify any Digital Certificate. So, any digital certificate owner can make fake Digital Certificate of any Authority and sign it by it's own digital certificate and forge the identity.

Currently, most common method used is a forged Digital Certificate with a field 'Expiry Date' with date just expired. So, what most of the Client-Side applications do due to their bad code is firstly check for 'Expiry Date'. If the Certificate has expired, they show a Warning Message to Client regarding accepting the Certificate and ask if certificate can be accepted. Normal practice is Client take a chance accepting it due to a slight time passed after Certificate expired. But applications then don't check for any other parameter including Certificate Validity and accepts the certificate.



SSL Stripping

i) ***SSL stripping attack*** : In most of Web Application, when we start surfing a site, by default applications create an HTTP based connection, which is later on shifted to a HTTPS connection by either a 302 Redirect to HTTPS page or clicking on a link/action to HTTPS URL. This attack works on principle of 'Best to attack SSL before User gets to it'.
So, for SSL Stripping attack the most common thing we can do is:
   - ❖ Redirect all web traffic of Victim to our machine and then forward it; we can use dual side ARP spoofing mechanism for it which works by fooling the Victim's Gateway MAC Entry and Gateway's Victim MAC Entry by Attacker's MAC.
   - ❖ On attacker's machine redirect all incoming PORT 80 traffic to any unused PORT say '#P'.
   - ❖ Then we can run SSL Stripping tool to listen PORT '#P' and log all details.

Here, SSL Stripping tool will look for all incoming data to Victim's Machine and changes all links with HTTPS protocol to HTTP protocol. This force the Victim's Web Application to send log-in credentials in unencrypted fashion as it will use HTTP link instead of HTTPS link. Then attacker can easily sniff the log-in credentials from the un-encrypted channel.

The only change victim can notice is no symbol (lock ) for secure connection in the Client-Side Web Browsers.

5) Exploiting Bad Code

SSL Digital Certificate MOD attack : There are tools to exploit bad code in Client-Side Applications. It targets all browser engines like Mozilla, Opera, Chrome and Trident.

Like NSS (Network Security Services) is set of libraries, APIs, etc. used by Mozilla to support cross-platform development of client-server application, but has several flaws in its huge library; few of them even deal with Digital Certificates, making all Mozilla Applications vulnerable.

Even Digital Certificate granting application has a flaw. If anyone requests a certificate for A.X.com, then Authority just picks up last host part i.e. X.com; run WHOIS command for it to get administrator's e-mail ID for sending confirmation mail, and on being confirmed issues Digital Certificate. So, its issued to X.com even if *A* is replaced with ***www.PayPal.com\0*** , Digital Certificate for ***www.PayPal.com\0.X.com*** is issued **to *X.com***.

i) ***Null Character Insertion***

Here, bad coded client-side application stores Domain Info in Digital Certificate in a Normal Char[] Array and tries to check with web services' domain name using functions like 'strcmp'. So, if I give a domain name like ***www.X.com\0.A.com***, char[] array stores it like ***www.X.com\0.A.com\0*** and returns ***www.X.com\0*** due to ***NULL***. So, this way ***www.A.com*** can pose as ***www.X.com*** .

ii) ***Wildcard * Match - One Card Rules All Sites***

Even NSS accepts ****\0.ABK.org*** and compare it as Wildcard character and thus providing you single certificate to work for all Sites Forgery.

iii) ***Wildcard | Match***

Even Pipe symbol for OR is allowed like get a certificate for ***www.paypal.com|mail.google.com|allSites.com\0.ABK.org*** allowing all piped sites forgery.

This way the LOCK symbol for secured connection is present on Browsers even if SSL is compromised; revealing no sign of attack to victim.

Chrome/Safari and Opera bypass NULL characters so are safe for this

attack; but for them we have following attack:

iv) ***Null Character Bypass Attack***
getting a certificate for www.Pay\0Pal.com; allows attacker to pose PayPal in case Null Characters are bypassed and browser reads it as www.PayPal.com from Pal.com .

Defeating Security's Security

i) ***Certificate Revocation*:** Uses OCSP (Online Certificate Revocation Protocol) with fields ResponseStatus (0-success, 1-malformed Request, 2-interval, 3-tryAgain) and ResponseBytes (data about Status); but if we set its ResponseStatus to 3(try later), it has no ResponseBytes (so no use of signature protection over it).

ii) ***Software Updates*:** Unencrypted channel is not safe and we just saw encrypted channel can be defeated. Real updates can't reach the application, instead hacker can replace them with malicious updates.

5) DeAnonymize Proxy

TOR working on Onion Proxy is one of the best Anonymizer, discovery of a Trojan infected TOR Client on User machines has posed a problem.
Even bigger problem is running malicious exit node (where data is supposed to exit Tor's Chain-of-Proxies and go in un-proxie'd' way to destination) using tools like HackedTor.exe. When passing unencrypted traffic (such POP3, IMAP, etc.) through these malicious Tor exit nodes, you are not only handling the content of your e-mail but also your credentials.
Researches analyzing Tor's traffic found Germany was country with largest TOR users followed by China, USA, Italy, Turkey, UK.
There were thousands of e-mail, telnet sessions unencrypted, and found one malicious exit node doing Reverse DNS Lookups on POP3 e-mail connection, presumably part of a scheme to harvest usernames and passwords.
Theoretically, One TOR Client is suppose to carry 4.x% traffic but it was found 98% traffic is going through 25% of Tor nodes and other 75% carry 2% of traffic.

6) DNSSEC Vulnerabilities

DNS Servers had a core bug that allows arbitrary cache poisoning even when

host is behind a firewall. DNS is even open for DNS Redirection and Man-in-the-Middle attack. SSL can't help as SSL Certificates are themselves dependent on DNS. Most of the Web Attacks occur because of authentication problem at DNS protocol, DNSSEC is required to implement authentication at DNS.

How DNS works is

a) Victim asks Victim's NameServer for IP of Host.com
b) Victim's NameServer doesn't have the IP address; so replies '*Unavailable, check with NameServer2*'
c) This procedure repeats till a NameServer like NS1.Host.com replies '*Available, Host.com is at CorrectIP'*.

Here, when Victim asks NameServer for IP, it passes on information about Victim's IP/Port and a random Transaction ID. In DNS reply, transaction ID is also sent back to authenticate the source of answer.

Now, attacker can tweak it, but fails to do so because of unique transaction ID (1 to 65535). Attacker can start guessing it before NS1.Host.com and that too for multiple times. This reduces Attackers probability to 2 years if attacker tries 100 times before actual NameServer i.e. ((65535id/100tries)/365days).

But looking at protocol closely, attacker can do it easily by doing lookup for a NameServer name at Host.com that doesn't exist and spoof the non-existing NameServer. Say NS16.Host.com doesn't exist, so attacker spoofs it with AttackerIP. Now this spoofed NameServer contacts intermediate NameServers and on victim's dns query replies 'Unavailable, check with Host.com at AttackerIP'.

This fools Victim to believe Host.com is at AttackerIP as its base of how NameServers are checked for DNS entries. This is one of the forms of DNS Cache Poisoning attacks.

DNSSEC provide origin authentication, integrity protection as well as means of Public Key distribution; even mechanisms for authenticated denial of existence of DNS data. It doesn't secure DNS against DDoS (Distributed Denial of Service) attack.

But these don't provide CONFIDENTAILITY (refer RFC4033 document), as Network-Sniffers can still find out the request made in DNS Query.

The basis of attacks is the Forgery Resilience draft that mentions longer the DNS TTL, lesser is the chances of attackers succeeding, and currently DNS TTL is of 1 Day, attacker will take more than 82 years.

Attacker gets his IP returned for all sites like this. And once attacker controls DNS, no Web Service is secure.

DNSSEC is supposed to secure DNS, but NameServer enumeration is much deeper in DNSSEC. DNSSEC have no protection from DNS Query Espionage (due to promiscuous network sniffing) and CPU Flooding (due to extensive computation during encryption and negative DNSSEC Responses require

extensive online computation). Backward Compatibility is also a problem.

7) FORENSiC ATTACKS

These attacks are possible mostly when physical access to machine is available, few major Forensic Attacks are:

a) Cold Boot Attack

Cold Boot Attack is one of the most interesting attack, only possible if you can get access to RAM of victim's machine just within few minutes of turning off the machine. Plugging it into your machine with necessary precaution and right set of tools you can get the data in state it was before power to RAM was turned off.

b) Imaging RAM

There are tools available as DD, etc. which can dump entire content of a RAM into a file for later Forensic analysis of files, information, etc. Its more dangerous on Windows machine as SAM file (Windows stores User Accounts login credentials in it in a secured mode) is replicated in RAM and thus can be stolen directly by imaging RAM.

c) Data Carving

Data Carving is a data recovery technique from Formatted or Corrupted or Damaged Data Storage Devices by finding pattern of file signatures in RAW mode (reading disk from First Byte to Last Byte irrespective of any File System). Several secrets have been revealed using this mechanism over machines donated or dumped by Corporate, Government, etc. when replaced with new machines. It’s highly dangerous. There are many professional Linux forensic tools available for this purpose.

d) Dig Information from O.S.

This attack depends on O.S., as on Windows machine HiberFil.sys (storing entire machine state on hibernation) and PageFile.sys (storing buffered data, as Swap Space in UNIX) can be replicated for data carving to extract information.

Slacker Tool can recover data from Slack space (the space left in Linux’s block or Windows’ clusters partially filled by file).

On NTFS partitions ADS can be searched for hidden information in alternate stream.

e) Dig Information from Files

Any agency (like Government, Military, Corporate Sector, etc.) tends to upload documents, images, data sheets, etc. on their Web portals for the general public to view. Some files not directly available can be searched using intelligent Google Search. Most of them are created & edited on Machines in organization only. Every time a tool edits a file most of them tend to add data as trace or metadata to the files which

is normally invisible. Tools like ExifReader, BinText, etc. can extract information from file itself to add more data to reconnaissance, like User Names, DNS Servers, Softwares installed, etc.

f) Timestomp

This is very nifty utility which can be used to manipulate timestamp for accessed, modified 'n created to any desired time.

Countermeasures

Now we shall look at possible solution:

- **O.S. Log-In Hack OR Bypass**
  The only solution is physical security of the machine, nothing else can safeguard you.
- **Res-Timing Attack, SMBEnum AND Slowloris**
  - Turning off Javascript is a partial solution for Res-Timing attack but still can't prevent SMBEnum attack; for that avoid using IE; but better is to report to Microsoft, Mozilla, Apple and Opera repetitively till they release an actual working update.
  - Slowloris can only be stopped by patching up the Server and IDS (Intrusion Detection Service), but no patch is fully functional currently.
- **SideJacking**
  - Using private secure VPN is a solution and not accessing sensitive data at just at any public Hotspot.
- **Defeating SSL**
  - Use a Secure Proxy Channel for SSLStripping, check for Lock Symbol for secure connection on browser.
  - For SSLSniff attack, use Chrome or Opera as they bypass NULL characters, but these are also vulnerable so Proxy is final solution.
  - Check for URL in certificate with one in Address bar and try searching information on the URL using WHOIS service.
- **DeAnonymize Proxy**
  - Select a secure reliable proxy, test it first reverse engineering and reading expert’s report on it.
- **DNSSEC Vulnerabilities**
  - Use Static IP Address mapping for sites which carry your sensitive data because till now there is no Patch available and entire Internet is insecure.
  - Use all DNSSEC patches available, this still doesn’t guarantee total security.

- Use DNSCurve (a similar implementation from security expert who popularized DNS attacks) which uses high-speed high-elliptical cryptography and provides Confidentiality along with Integrity and Availability.

- **FORENSiC ATTACKS**
  - ***Encrypt Disc's Content***
    Encrypt the entire Disc with application like Truecrypt which even allow you to encrypt entire Disc, Drive, Virtual Partition with loads of tested Open Source Encryption algorithms like AES, etc.
  - ***Secure Delete***
    Delete any information using Shredder, Eraser, or similar tools that will try their best to make file recovery nearly impossible.
  - ***Zip Bomb***
    This is a specially crafted Zip file with size in Kilobytes, but when any attacker tries to spy around your content and unzip it. It unzips to Petabytes of data filling up entire disc and crashing the attacker's machine.

Conclusion

The conclusion of this entire seminar on "Insecurity In Security" is that no matter how hard you try, you are never fully secure.
So, keep track of latest security breaches and stop or start using resources according to their vulnerabilities. Websites like ***www.SecurityFocus.com***, ***updates.zdnet.com/tags/security.html***, ***www.cert.org/vuls/***, etc. are one of good sources to stay update.
Just devising strong security parameters never make you secure, perfect implementation is as important as perfect design.

Most of the Insecurity In Security comes from badly written piece of code and we have only careless developers to thank for them.

Reference

For this discussion mentioning the list of all sites, papers would fill another report; so instead I would mention the names of researchers whose work I referred to...

- Thorkill (*piotrbania, KryptosLogic*)
- Billy Rios (*Security Engg., Verisign*)
- Robert 'Rsnake' Hansen (SecTheory Internet Security)
- Joshua 'Jabber' Abraham (Rapid7)
- Robert Graham (CEO, Errata Security)
- Moxy Marlinspike 'mox·ie mar·lin·spike' (Instt. Of Disruptive Studies)
- Dan Kaminsky (Director, IOActive)
- Adrian Crenshaw (InfoSec Enthusiast)

Also, mentioning two major conferences actively participating on it..

- BlackHat Conference
- Defcon Conference