# Common Threats to Cyber Security Part 1 of 2

## Table of Contents

<table>
<thead>
<tr>
<th>Threat Type</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Malware</td>
<td>2</td>
</tr>
<tr>
<td>Viruses</td>
<td>3</td>
</tr>
<tr>
<td>Worms</td>
<td>4</td>
</tr>
<tr>
<td>Downloaders</td>
<td>6</td>
</tr>
<tr>
<td>Attack Scripts</td>
<td>8</td>
</tr>
<tr>
<td>Botnet</td>
<td>10</td>
</tr>
<tr>
<td>IRCBotnet Example</td>
<td>12</td>
</tr>
<tr>
<td>Trojans (Backdoor)</td>
<td>14</td>
</tr>
<tr>
<td>Denial of Service</td>
<td>18</td>
</tr>
<tr>
<td>Rootkits</td>
<td>20</td>
</tr>
<tr>
<td>Notices</td>
<td>22</td>
</tr>
</tbody>
</table>
Malware aka Malicious Code or Malicious Software

Hostile, intrusive code designed to infiltrate a computer without users knowledge or consent

Examples
- Viruses
- Worms
- Trojans (Backdoors)
- Downloaders
- Attack Scripts
- Botnets

**015 So, let’s talk about some common threats. Malware is a big category of bad stuff that you will see. By malware, this could be malicious software or malicious code. Generally, this is a hostile or malicious program that somebody's installed on your system to take some type of action. And what that action is depends on what it is. If it’s a virus, it might be there to destroy a program. If it’s a worm, maybe it’s there to join that computer to a botnet or something like that. Maybe it’s there just to provide access to the attackers or whatever the case may be.
Viruses

Characteristics

- Infects other applications and uses them to self-replicate when executed
- Runs on system without user’s knowledge

Examples

- Melissa, Funlove, Xorala, Spaces

Trends

- Dominant in the 1990s
- Some limited activity today
  

**016 Viruses, this is not necessarily a declining malware category, but you used to hear-- in early to mid-2000s you heard a lot of virus this, virus that. This virus is coming out. It’s going to shut down the world on this date and at this time. But basically it’s a piece of malware that infects other applications. And it self-replicates, meaning when it's run, or when the virus runs, it starts spreading to other places automatically. It doesn’t take interaction or anything else in order to replicate itself. So, it just starts spreading.

And generally, they run on systems without the user’s knowledge. The more public ones, they did something like pop up a message box that said I love you, or
something like that. I mean they were really kind of silly stuff. Some examples--so again, I think you see this was kind of a big malware category a while ago. You don’t hear too much out of this anymore because it’s a lot easier just to do a Trojan or a downloader or something like that to get access to systems. Viruses now are written for notoriety. People deploy a virus because they want their name out there. Or they want to be the guy behind the next big virus or something like that.

Worms

Characteristics

• Automated propagation via targeted exploit
• Manual propagation via Social Engineering component

Examples

• Love Letter, Slammer, Blaster, MyDoom, Netsky, Sasser, Plexus

Trends

• Popular 2001-2005
• Impact has decreased significantly with the Implementation of Windows Firewall

**017 Worms, this is another category of malware. This will take advantage of a exploit- or, sorry, a vulnerability in an operating system to spread. Could also
do manual propagation. So, it could
spread manually through some type of
social engineering, meaning I give John a
USB drive with game.exe on it. He puts it
into his PC, double clicks it. And off this
thing goes because he thinks it’s a game
or something like that when really it's a
worm. And it’s going to start spreading.

So, you see a lot of examples, certainly,
more recently than viruses. But you see a
lot of these start to spread based on a
Windows vulnerability. So, Microsoft
Windows has a vulnerability. It’s common
for malware writers to take advantage of
that and push out a worm, especially for
remote exploit vulnerabilities. So, the
worm will take advantage of that. And
then it will scan for other Windows
computers around it. When it finds it, it
will try the new exploit against it and will
try to spread.

So, within the last ten years or so, you've
seen a lot of good examples of worms
taking advantage of Windows, taking
advantage of SQL server and that sort of
stuff. Kind of have seen a decline recently
because of the Windows firewall that got
implemented with XP Service Pack 2 and
later. Although, generally, that prevent the
riff raff from running their malware on you,
but there are ways that malware writers
have gotten around that.
**Downloaders**

**Downloaders**

Typically the first infection vector – gives an attacker the ability to dynamically push malware

**Characteristics**

- Used to install another Trojan, adware, other malware
- Generally written in VBS or JS for simplicity
- Easier to circumvent AV signatures with a new downloader/dropper than to recreate a Trojan or exploit code

**Examples**

- Zlob
- Agent
- Pushdo

**018 Downloaders, this is usually the first infection vector, meaning when somebody targets you for an attack, and they send you a piece of malware, it’s going to look like this. It’s going to be a downloader. Why do they do this?**

Well, first let’s answer what a downloader is. It’s a very small simple piece of code. And its entire purpose in life is to go grab some other piece of malware, download it to your computer, and run it. So, these are called downloaders or droppers or something like that. Why is this useful to an attacker? Any thoughts?

Student: They're small.
Chris Evans: They’re small. So, it makes delivery really easy.

Student: It’s more easily undetectable.

Chris Evans: Potentially. More undetectable than say a bigger piece of malware, yep. It's easier to change this than your main malware. So, if your initial attack is this, you can change it to be whatever you want. Go out and grab this malware. Go out and grab that malware. It gives the attacker a lot more flexibility. And that's why you see this type of stuff being the-- kind of like the first wave of malware coming in nowadays.
**Attack Scripts**

**Characteristics**

- Drive-By
- Commonly use JavaScript
- Target a specific vulnerability (IE, Acrobat Reader, Real Player, etc.)
- Target web application vulnerabilities (SQL Injection, XSS)

**Examples**

- Browser Exploitation Framework (BeEF)

**Trends**

- A malicious website tool, Blackhole, was responsible for 1/3 of the drive-by malware infections
- This increases to 50% when combined with another tool, Incognito
  - Most of these sites are active only for a few hours
  - But, these sites can infect thousands of users in that window

*The Frontline, Oct 2012*

**019 Student:** I guess they could update their code, too because it’s pulling code from some external site. That code’s not working, then you could just change that.

**Chris Evans:** So, a good example when I was doing pen testing we had our own downloaders that we would install. But the reason we did it wasn’t to get around antivirus or something like that. It was so we could push custom malware, or modified malware to our-- the organization that we were pen testing. So, frequently we would restrict our tools to be bound by IP address. So, if the tool was run outside this IP address, it wouldn’t work. That’s one way that we used to control our tools to make sure nothing got out and spread.
But what we could do is change the downloader and say go grab this copy of the malware that’s now got this IP address in it, or this IP address range. And it gave us a lot more flexibility to push out the tool set that we wanted when we wanted to.

Attack scripts, so you see this very frequently within web browsers, where people are coming to websites, and they’re being attacked through what’s called drive by, meaning you come to a website. And it figures out what your computer is vulnerable to, delivers the appropriate exploit for it, and takes advantage of your computer.

So, there’s the browser exploitation framework does this. There are two other tools out there. One was called mpac. And the other is black hole. Basically, these tools, along with incognito, these are web exploitation tools. What they do is your browser, the web browser on your computer, as you come to that website, it asks you what operating system are you. What browser are you? What plug ins are you running? Do you have Flash? Do you have Shockwave? Do you have Quicktime? Do you have PDF? It will ask you all of that. And then it will say what version of all those do you have. Look at this. You gave me a version of Quicktime that’s vulnerable to this. All right, well I’ll give you the appropriate exploit for that. And so, you see these tools are really smart when it comes down to delivering a payload-- or delivering and exploit to you to take advantage of your web browser.
Botnet

A collection of compromised computers remotely controlled which can be used for

- Trojan, virus, and/or worm propagation
- Spam and/or phishing attacks
- Denial of Service attacks
- Obtaining personal information through
  - Adware
  - Spyware
  - Keystroke loggers

Initial compromise

- Web downloads
- Email attachments and links
- Instant Messaging attachments
- Unpatched systems
- Lack of network and/or host-based firewalls

**020 A botnet, these are basically a collection of computer systems that are--can be remotely controlled for various purposes. We see this with-- they usually come in through some type of Trojan or worm malware, could even be virus malware, although we don’t really see that. I think it’s more Trojans and worms. But once they infect your computer, your computer gets joined up to this collection of other computers that are controlled by a guy called the bot master, or the bot herder.

So, what he does is he's got access to all these computers that have been infected with his malware. And he can say I want all of you to send spam. I want all of you
to send malware. I want all of you to
attack this one computer over here, or
something like that. But think about how
a-- the power of having three million
computers at your command. You have
the ability to control what three million
computers do. You can do a lot of
damage with that.

You look at-- how many of you heard of
the conficker worm? So, conficker was a
really nice worm that ran around in late
2008, early 2009. And I don't think the
writers behind it really tapped the power
of the worm. What they basically did with
it was kind of fight with the defensive
actions of Microsoft, the research
community, security consultancies, and
that sort of thing. But basically, they didn't
actually do anything with it until the later
stages. So, middle of 2009, they started
sending malware out with this worm.

As of probably two months ago-- so, here
we are like three years after the fact,
there are still three million infected
computers with this worm. So, whoever
the guys are behind conficker still have
three million computers that they can tell
do-- take action X or take action Y.
**021 As an example here of what a botnet would do, we call this an IRC botnet because it's using Internet Relay Chat as its command and control function. So, you have this guy at the top. This guy right here in the center of the slide. How does he communicate with all his three million hosts? Well, it depends on the malware and how it's written. In this case, we're saying IRC, which is a chat function. So, basically, he communicates with all his three million little zombies out there with-- over the IRC protocol. So, he can send instructions to the zombies. The zombies can push information back to him. All over this existing chat protocol.
So, how this actually gets instantiated, if you start over on the left side of the screen here, you've got step one. The bot herder is basically sending a phishing email or some type of malware to a potential victim. Down here the victim actually opens that email or goes to the website or falls victim to whatever the bot herder sent him. And it turns the computer into a malware infected host, a zombie.

So, at this point, he's been infected. He'll start downloading additional malware, additional capability to maybe send spam or send malware or maybe conduct a denial of service attack, depends on what the bot herder wants.

And then the last step here-- so, the bot herder out there's going to send instructions over IRC to say take action. And then the zombie is either going to spread malware or send spam or maybe deny service to some website or something like that. Can any of you think of any really good examples recently where you had denial of service attacks against major companies or major things going on? How were those attacks accomplished?

With a botnet, right? So, you've got, again, three million-- actually it doesn't take three million. It could be five thousand hosts. It could be a hundred thousand hosts. It could be three million hosts. But all of them are doing-- sending traffic to a website. What happens to that website? Right? Because now instead of seeing my normal traffic, I'm seeing ten thousand requests every second from all
of these zombie hosts. And that tends to overload things. And that’s how you see these denial of service attacks are being done today.

Trojans (Backdoor)

Characteristics

- Don’t self propagate
- Can be used as keystroke loggers, password stealers, network sniffers, remote control, spam

Examples

- Winny, Limewire, Zeus, Sasfis
- Back Orifice, Agobot,

Trends

- Often a key component of a Botnet
- According to Eleven, a single variant of the Sasfis trojan represented 71.8% of the TOTAL malware in 2010; in 2012, a single Zeus variant represented 34.8% of total malware

- source: www.eleven.de

**022 Trojans are backdoor programs. Basically, these come masquerading as a game, a plug in, anything. But they're there to trick the user into thinking that it’s not malicious, when in fact, it’s going something in the background. It could be keystroke logging. It could be stealing passwords. It could be sniffing traffic. It could be downloading additional malware. But the point is that it’s a Trojan horse. It’s masquerading as-- it’s malware masquerading as a legitimate program.
So, a good example from this, when you go out to a website. And let’s say you’re looking at flash videos or something like that through YouTube. If you don’t have the plug in for it, what does it usually say? If you wan to view the video, download the plug in. And almost always there’s a plug in link that says download this and run it. This has a high propensity for malware.

I did a social engineering attack during a pen test. And that’s what we did. We created a website that said if you want to see this information, and it’s a silly little chart, right? Pay scales or something like that. If you want to see the information, you have to download the plug in. What do people do?

Student: Download it.

Chris Evans: Well, of course. They download the plug in, right? Well, this is a Trojan. Right? So, not only did it give them access to that information-- so, we completed the loop. So, they could see the web page. User’s happy. In the background, I’m downloading my malware and installing my tunneling programs and that sort of stuff so I can gain access to their system.

The user doesn’t know anything is different because they saw the web page. They see the information that they want to see. And they think the plug in is doing its job.

Student: How do you really know what to trust and what not to trust when it comes to those types of downloads?
Chris Evans: Don’t trust anything. So, there are some things that you can trust. If you go to the manufacturer’s site directly like adobe.com, you can get Flash, Shockwave, PDF viewers.

Student: See, even sometimes I’m a little bit leery of that because who’s to say that someone couldn’t replicate an Adobe look and still try to get into your computer?

Chris Evans: Well, that’s true. If you downloaded it straight from Adobe’s website, though, adobe.com, chances are good you’re all right. But if you grab it from adobe123.com.com42.asia, that’s probably not Adobe. That’s probably something that looks a lot like Adobe Reader or Flash, it’s probably got some other stuff going on in the background that you won’t like.

Student: Okay so, that is really your answer to tell folks to go to the originator--so, I’ll describe that. And then go back out to whatever site they needed to to get the information they needed.

Chris Evans: Yep. So, let’s say I’m looking at music videos on musicvideofiles.com. I have no idea if that’s a legitimate site. Please don’t go to it. It’s probably a porn site. But so, when you actually go to this site, and it says if you want to listen or watch the music videos, you need the Adobe Flash plug in. And there will be a link there that says download now, run this executable. And it will probably have instructions that say trust this, click yes to that. My guidance here would be instruct your users don’t click on that link. If it says it’s using Adobe Flash, go to Adobe’s website. And grab
the real flash player. That's the only way you're going to know for certain whether it's a legitimate copy of the plug-in viewer or not.

However, if you are in a managed security environment, I think the better approach is that you push those plug-ins to your browsers automatically and keep them updated. That's one thing people don't do is they'll push the copy five years ago. And they haven't updated it since then. So, you can go to any browser and look at it. And it's probably got outdated Flash and outdated PDF viewer and that sort of thing. But push those plug-ins to the browsers automatically. And if something wants a different plug-in or a different media type, you don't allow it. You block it at the proxy device. You filter those out. And you say I'm sorry. We've given you Flash. We've given you Shockwave. We've given you PDF. If it's outside of that, you probably don't need it. But you're welcome to try to convince the security people otherwise. Here's our phone number.

One of the more interesting sites that I did a pen test against, they actually had set up white listing on their proxy devices. So that if you went out to a website, if it was in the white list, you got to visit it. ESPN.com, CNN.com, you have to check sports and news during the day. Those were legitimate. But if I went to myhackerfiles.com or something, I would get a page that said I'm sorry, this site is prohibited. If, however, you need access to this legitimately for your site, or for your business, please put in your name, your phone number, and why you think you need access, and we'll turn it on in 24
hours if you need it. And so, what that did was it gave people a way to access legitimate sites, but also if they needed to access something perhaps less savory, they could. They’d just have to tell the security folks about it.

**Denial of Service**

Denial of Service (DOS) is an attack on the availability of a service

- Attempt to deny, degrade, disrupt or otherwise interfere with the ability of a provider to keep services available

DOS attacks usually target chokepoints or single points of failure within the network:

- Bandwidth – by generating more traffic than routes can handle
- Servers – either directly (e.g., the OS) or indirectly through a supporting function (e.g., the power, A/C, etc)
- Applications – attack the application’s ability to process requests
- End Users – attack users to prevent them from accessing services

**023 Denial of service attacks, these are attacks on the availability of systems. Remember, as a security person, what are you concerned with? Insuring-- what are the three? Confidentiality, integrity, and availability. Right? The CIA triad.**

So, a denial of service attack is an attack against the availability of a system. And it will usually target choke points within the
network. So, it'll go after bandwidth. Let's say that you have one link between your server-- your web server and the outside world. And let's say it's one megabit per second. If an attacker can generate two megabits of traffic-- two megabits per second of traffic on that link, they win, right? Because their capacity, the attacker capacity to generate attack traffic exceeds your capacity.

Conversely, if you have one megabit per second of traffic, and the attacker can only generate let's say ten megabits per second of traffic, you win because you have more capacity than they do. Denial of service attacks are all about exceeding capacity, be it through bandwidth, through servers, like going after the number of connections that an application will support, targeting the operating system directly, or even the end users. So, attacking the end user like changing certain things on their computer so they can't access the Internet or websites or something like that.

And an interesting point here about indirectly DOSing something, if I can walk up to a data center and pull the power cable on the air conditioner, I have a much greater chance of conducting a denial of service attack than trying to do it over the network. Right? So, if I understand how the system operates, I can try to find ways around it if my primary-- if I can't send enough traffic or something like that. If I can't do something over the network, there might be other ways around that, either by pulling cables on air conditioners, or affecting power or something like that.
Rootkits

Programs that have the ability to hide themselves and cover traces of a hacker’s activities from the user and potentially the operating system.

Typically, a rootkit will “hook” calls to operating system functions to facilitate:

- Process hiding
- File hiding
- Registry hiding

Root kit hides two files from the user

**024** Root kits, these are programs that affect the way your operating system functions. Typically root kits do what are called hooks into the operating system. So, they change the way that your operating system works. In the example here we have a root kit infected system. That’s what this big red box is. You have a user over here. And the user is asking the operating system show me all the files in the directory. Basically, directory listing, right?

Well, that call goes through the root kit because the root kit’s been installed and is hooking things within the operating system. The operating system gets the function call that says list all your files.
And the operating system responds with I have one, two, and three.txt. So, I've got three text files. Well, that answer comes back through the root kit. And the root kit modifies that answer such that only one.txt-- it basically strips out two.txt and three.txt, strips those out and only gives you back one answer. So, what has the root kit done here? It's hiding files. Right? So, now apply this to processes, network traffic, and pretty much anything that operating system can do. And you can see how advanced root kits can get if they can affect anything that operating system can try to tell you. It has pretty much control over anything that you see.

Why would an attacker use a root kit?

Student: To hide themselves.

Chris Evans: To hide themselves. Yep. Is your every day average hacker going to do stuff like this? Probably not. And the root kits that are freely available out there get hit by antivirus all the time. Or get popped by antivirus, or there are antivirus alerts for it. So, they're not going to use those. Where you see these are more in the APT category of attacks where you've got somebody who wants to stay in the system for two or three years, doesn't want to be detected. They'll do stuff like this.