



They're Hacking Our Clients!

Introducing Free Client-side Intrusion Prevention

Jay Beale

Creator - Bastille UNIX

Co-Founder - Intelguardians, Inc.



Penetration Testing

I work for Intelguardians, a security consulting firm. I get to do and lead a fair bit of network and application penetration testing.

A penetration test usually focuses on the compound question: could a "hacker" break in from the Internet and how far could he go?

The hard part is getting into the "internal" network.

Once you're inside, things get far, far easier.



Target: Internal Network

Our penetration tests get much, much easier from the “internal” network.

Most organizations architect their firewalls for three zones: Internet, DMZ, and Internal networks.

From any machine in the Internal network, we usually have an extreme level of access.



A Curious Trend in Ethical Hacking

Over the last year, we're finding that compromising the Internet-accessible servers has gotten far more difficult.

Increasingly, we've been getting to the internal network via client-side attack, hacking the Security or IT staff's workstations via vulnerabilities in their browsers, mail clients, Acrobat and Office programs.

These attacks have gotten easier for anyone with a copy of Core IMPACT, Metasploit, or hostile attacker toolkits.



Hacking Acrobat Reader

Our clients choose the scope of targeting, often allowing us to target only the Security staff. On a penetration test for a company about a year ago, we were allowed to target only one IT Security person. He was a tough cookie: IE7 and well-patched.

We used every single client-side exploit. We compromised his machine with the last exploit, which targeted Acrobat Reader.

But that was enough. You only need one.



Professional Hackers Started Years Ago

Real attackers moved to client-side attack years ago.

There's so much money in hacking the clients that it's become a great business for organized crime.

And this has become so successful, the attackers' chief problem became creating an easily-controlled, scalable means of managing all the systems they've compromised.

And so they brought us the botnet.



Workstation Control is Powerful

Most botnet owners have so many machines that they don't even inventory them and figure out what companies and organizations they've compromised.

Congressional computers were in a botnet. But nobody changed any laws! Or did they?



Penetration Testers

As highly-targeted attackers, penetration testing teams use these machines as a foothold to hit the internal organization. We get access to file shares, cached credentials, and applications that have never been designed or audited for security.

Further, even across their worldwide WAN, even the largest organizations have no filters.



Isn't this Social Engineering?

In the security community, we initially write off these attacks to social engineering. We blame the user.

Not all exploits require user interaction. And if they do, we'll always have some users get fooled. Even if that's 1/100 of 1%, it's bad.

But blaming the non-IT user isn't fair.



Isn't this Social Engineering?

Blaming the non-IT user isn't fair.

Your grandmother shouldn't have to understand vulnerabilities to read e-mail. You can't expect her to unless you really make a driver's license for computing.

It's our responsibility as IT architects to train the user, but to protect them from attack anyway.

What about your mortgage broker's computer? Or your dentists? High value target, no IT staff and little training. They've probably been owned.



Why is this difficult?

Most organizations' security has been focused primarily on the perimeter and on firewalls. That over-focus is decreasing, but only so fast.

Most security efforts are focused on the servers, particularly those accessible from the Internet.

This focus really has started to achieve its goal. Hacking organizations, from the Internet, through their servers, is finally getting difficult.

But the attackers have moved to attacking the workstation PCs. And few organizations have kept up with that change in focus. It's a difficult problem...



Why is this difficult?

First, the numbers are against us. As an attacker, I only need to find one workstation or laptop that has a vulnerable client out of the 10,000 you have.

And you thought protecting 150 servers was difficult!

Second, your users can stay disconnected from the network or have their machines powered off for extended periods. One special case - when someone leaves the organization, we often turn their system off for six months, until their position is filled again.

Patching has always been a race condition!



What about Patch Management?

The next thing we all think is...this is where patch management products should make the problem irrelevant.

But:

- 1) Not every organization has a commercial patch management tool.
- 2) Patch management tools may rely on a host inventory that isn't accurate. Here are some hosts commonly left off:
 - Old hosts that aren't part of the domain or inventory.
 - Dedicated scanning / machine control systems.
 - Hosts brought to the office from partner companies.
 - Legacy systems of any kind!
- 3) Patch management tools don't always track every third-party product.



The State of Internal Patching

Actually, most organizations don't patch consistently or frequently enough to avoid this threat.

Even if they can do consistent and frequent patching, they tend to only be comprehensive for Microsoft software.

Even those that do this perfectly will have trouble keeping organization or user-installed browser plug-ins up to date.

Well if we're not solving this problem via patching, what about our regular vulnerability assessments?



Vulnerability Assessments

First, most organizations don't perform vulnerability assessments more often than quarterly .

Second, their vulnerability assessments focus on the servers.

That's natural. Servers actually answer you when you probe them and usually give you their version/patch level fairly easily.

Clients aren't quite so helpful...or are they?



Clients Identify Themselves Too

A whole lot of client-side software identifies itself often. We just need to know to listen...or sniff...or read the logs...

First, web browsers tell every server they talk to what version they are:

```
HTTP_USER_AGENT = Mozilla/5.0  
(Macintosh; U; Intel Mac OS X;  
en-US; rv:1.8.1.4)  
Gecko/20070515 Firefox/2.0.0.4
```




Mail Clients Too...

Mail clients send their version string with every single message. I once had a friend e-mail me to tell me that my Thunderbird version was old and vulnerable.

Here's is string from an e-mail I got from another security conference speaker:

User-Agent: Thunderbird 2.0.0.6 (Windows/20070728)

Here are a few more from people I work with:

User-Agent: KMail/1.9.9

X-Mailer: Apple Mail (2.919.2)



Watching without Sniffing

If I just watch all the browser user agent strings as people on my network browse, I could easily give you a list of vulnerable browsers.

But what if I don't want to sniff the network?

Most large organizations use transparent web proxies to decrease their Internet bandwidth costs - why download the same CNN graphics 2,000 times today?

Squid proxies, Cisco Cache Engines and many of others, easily log browser user agent strings. I can watch these for malware and vulnerable browsers.

Sendmail can be configured to log mail client user agent strings as well.



Sniffing

Alternatively, you can sniff the internal links to your outbound mail relays and outbound transparent web proxies.

But, either way, we're missing something in the browsers, aren't we?

Can anyone tell me what it is?



Browser Plugins!

Browser exploitability sometimes relies on third party code that may not even ship with the browser.

People add their own plug-ins, often automagically when they try to use a website that needs it.

They don't necessarily know they need to look for patches. They may not even know what vendor the plug-in came from, since the site they were using sent them to the vendor site to download the plug-in.

IT Departments can find it difficult to track these plug-ins, especially when they didn't install them!

Let's look at a couple examples...



Adobe Acrobat Reader

The Adobe Acrobat Reader Plugin:

Adobe Acrobat Reader Browser Plug-in for MSIE Malformed PDF
Request DoS Dec 27, 2006

Adobe Acrobat Reader Plugin for Microsoft IE Microsoft.XMLHTTP
ActiveX CLRF Injection Dec 27, 2006

Adobe Acrobat Reader Browser Plug-in PDF XSS Dec 27, 2006

Adobe Acrobat Reader Browser Plug-in PDF CSRF Dec 27,
2006

Adobe Acrobat Reader Browser Plug-in PDF Handling Memory
Corruption Dec 27, 2006



Macromedia Flash Plug-in

The Macromedia Flash Plug-in:

Adobe Macromedia Flash Player Plug-in Multiple Browser Remote Keystroke Disclosure
Apr 11, 2007

Macromedia Flash Flash8b.ocx Flash8b.AllowScriptAccess Method DoS Dec 29, 2006

Macromedia Flash Player swf Processing Multiple Unspecified Code Execution Mar 14, 2006

Macromedia Flash Player Flash.ocx ActionDefineFunction Function Arbitrary Code Execution
Nov 7, 2005

Macromedia Flash Player Flash.ocx Unspecified Function Arbitrary Code Execution
Nov 4, 2005



Detecting Plug-ins

Rsnake announced an excellent tool for this at Toorcon Seattle:

Master Reconnaissance Tool

Visit this URL to see what your browser's plug-ins are:

<http://ha.ckers.org/mr-t/>

Here are some of the highlights from my browser:

Java Embedding Plugin 0.9.6.2

Shockwave Flash 9.0 r28

QuickTime Plug-in 7.1.5

Move-Media-Player.plugin npmnqmp 07074032

JoostPlugin.plugin



What about Non-network Software?

The applications targeted most: browsers, mail clients, browser plug-ins, and Microsoft Office.

Larry Pesce from the PaulDotCom podcast highlighted the Metagoofill tool by Christian Martorella.

<http://www.edge-security.com/soft.php>

It searches a website in Google for public documents, including PDF, DOC, XLS, PPT, SDW, MDB, and SDC.

It then parses out metadata, including creator, creation time, and version of the client.



Document Metadata

If we can pull newly-saved/sent documents from file shares or sniff them on the wire, we can parse them for metadata.

If John just created this Word document five minutes ago, with a vulnerable version of Word, on John's laptop, we can be pretty sure that his laptop still has a vulnerable Word program!



Looking Up Version Strings

So we've got version strings accessible to anyone who can read certain logs. You can get more if you sniff the network. And still more if you potentially inject a MR-T iframe in each person's browser once per day.

If you put it together with a nightly database update from OSVDB (<http://www.osvdb.org>), you've got client-side vulnerability assessment.

But we can go further than this.



Client-side IPS

We've built a new free tool called Clientside IPS.

<http://www.ClientIPS.org>

It works as an inline transparent proxy.

If your browser requests an external website while revealing that it or its plugins have significant vulnerabilities, I can keep it off the Internet, redirecting it to a captive patch portal.



Client-side IPS for Other Clients

You can do this with other protocols besides web.

In the case of a vulnerable mail client, you could:

- 1) Deny the user access to their mailbox if their mail client is vulnerable. On their next web request, we can inject a frame to explain what's going on and how to patch.
- 2) Inject a mail message into the stream that explains what is going on.



About the Speaker

Jay Beale created two well-known security tools, Bastille UNIX and the CIS Unix Scoring Tool, both of which are used throughout industry and government, and has served as an invited speaker at many industry and government conferences, a columnist for Information Security Magazine, SecurityPortal and SecurityFocus, and an author/editor on nine books, including those in his Open Source Security Series and the "Stealing the Network" series. Jay is a security consultant and managing partner at Intelguardians, where he gets to work with brilliant people on topics ranging from application penetration to virtual machine escape.