

DCSB May 4, 1999

## Client Security: You've got armored trucks, but what about the pick pockets?

Trusted paths are required for integrity of any security model. An attacker will always go for weakest link in the trusted path. In today's world that is the client system.

### **Endpoint Security Problem**

Eugene Spafford from Purdue University came up with the following armored car analogy for e-commerce. Using strong encryption with 128 bit SSL for the transit of ecommerce information is analogous to using an armored care to move money. The problem is that in today's world we are moving a delivery for someone living on a park bench to another person living in a cardboard box. This is what is known as the "endpoint security problem". Even if your transit layer is secure all you have done is secured against passive eavesdropping.

While the park bench dweller is sleeping someone can pick his pocket and while the cardboard box dweller is not home someone can tear a hole to steal the goods. This is known as an active attack.

### **Server Problems - Shopping Carts - Cold Fusion**

There are problems today seen in the wild with ecommerce servers: the myriad of misconfigured shopping cart servers that have recently come to light, the generic application server vulnerabilities such as the one in Cold Fusion that was recently exposed by the L0pht. Then there are the holes that show up, several times each month, in major operating systems such as Unix and Windows NT.

### **90% of the clients are Microsoft Windows**

I won't talk too much about the server or transport layer of an ecommerce system today. I will talk about the security problems on the client. 90% of the clients today are Microsoft Windows so I will talk mostly about that operating system. This is not a slam against Microsoft as I think the Macintosh OS has the same problems. It just isn't as popular so there are

less vulnerabilities and attack tools published for it. This ubiquity is great for attackers. The client system "monoculture" allows for widespread automated attack where Win95/98 and IE can be found on greater than 60% of client systems.

### **Single User System**

Windows 95 or 98 is a single user operating system. It has some notion of "profiles" for different users so they can customize the screen, etc. But any user can modify any files on the system. This means an attacker's code can modify any file on this system. Essentially every process is running with administrative control on the system. The way I like to put it is "If I can get code to run on your system... it's toast."

### **Secure Systems**

Unix and Windows NT takes care of this problem with file system security built into the kernel and users running code with multiple privilege levels. This doesn't mean they are secure but it is the start for any notion of security. Java takes care of the problem of running potentially dangerous code, which is everything that comes over a network, in a sandbox that limits access to the underlying operating system.

### **Lack of Security Model - Band-Aids**

Windows lack of a security model leads to all sorts of Band-Aids to try to shore up the underlying insecurity of the system. The most popular solution is that the user is prompted with all kinds of dialog boxes warning that the next action they are taking could compromise the security of their system. Most users don't know what to do with these warnings so they select OK or worse yet they click the check box not to see the warning again. This security band aid relies on the user to be security conscious so it is doomed to failure.

### **Dancing Bears - Executable Attachments**

When confronted with the question of whether to see the dancing bears and potentially compromise security or to keep a secure system most users choose the dancing bears. The operating system is not allowing the user to see the dancing bears and stay secure. The attack via an executable attachment is the most popular and the most effective. How many of you have received an email attachment virus such as happy99.exe. I have gotten

these from IT professionals who should have known better. Imagine what it is like in the wild.

Any trojan code can be wrapped in a benign looking greeting card or joke program using public domain wrapper programs.

### **Peer Service Vulnerabilities**

There are other attack vectors which do not need the user to be a willing participant in his or her own demise. There can be vulnerabilities or misconfiguration of "peer services" such as web sharing or file sharing. Many people share their hard drives world writeable or with easily guessable passwords. There are many programs that will do dictionary attacks on windows file shares. Since windows has no auditing or account lockout it is hard to detect intruders.

### **Buffer Overflows**

One of the scariest and devastating vulnerabilities in clients are buffer overflows. This is when the programmer who has written a network client or some system code, which is used by a network client, makes a coding error. Assumptions are made about inputs such as terminators, format, or length. When the input data comes in from an untrusted source such as the internet an attacker can manipulate the data to exploit the assumptions made by the programmer. Usually this takes the form of getting the stack to overflow so data is executed as code.

### **Secure Coding**

Writing secure and robust code means making no assumptions about the input data and gracefully handling problems. In a perfect world this would be everywhere. But in a rush to market things are pushed out the door before they are properly tested. Code is reused that was not built to withstand hostile input. Buffer overflows have been well known for many years but they still occur in almost every client program.

### **L0pht Releases 2 IE 4.0 Overflows**

In the months following Internet Explorer 4.0's release L0pht published 2 buffer overflows which would cause a user's system to execute any arbitrary code an attacker wished. We find it hard to believe we are the only ones with the expertise to find these problems. We just think we are the some of

the few who make this information publicly available. This means that there are more problems in the code that are not made public.

I am sure that many governments around the world have teams that find similar client vulnerabilities that can be used to execute arbitrary code on client systems. I have personally been approached by people trying to purchase this type of information as long as it was kept secret.

### **Interpreted Data - VB Macros**

Data that is interpreted such as dynamic html, JavaScript or visual basic macros is another attack vector. JavaScript and dynamic html were designed to be downloaded from the internet so they have some notion of security but visual basic has none of this. The Melissa virus is an example of this new breed of exploits. Again the security model Microsoft gives you is "all or nothing". If you want to run this macro you have to be willing to compromise your complete system. There is no sandbox.

### **Statistically Possible to Compromise Many Systems**

So there are many ways for attackers to run code on client systems. It may be difficult to attack a particular system but statistically it is easy to attack many systems out of the total pool of internet connected clients.

### **Compromised System-Back Orifice**

Once a windows client system is compromised an attacker usually sets up a "remote administration" program. On a Unix system an attacker would typically set up a root shell to control the system. On windows a shell really is not enough because there are not enough command line tools on windows. An attacker needs to bring his or her own full set of control tools. Remote administration programs such as Back Orifice, NetBus, or D.I.R.T compact many tools into one small executable program. They feature the ability to capture the screen, send and receive files, modify the registry, look at the video and sound input devices, and do network redirection. But most importantly they allow for keystroke logging.

### **Keystroke logging**

Keystroke logging is the single most dangerous aspect of these programs. 99% of consumer authentication on the internet is done via reusable keystroked passwords. They are used to access pay services, for internet email and for online banking. Once keystroke logging is turned on every character

the user types is written to a file. The file may be picked up later by the attacker or it could be transferred automatically.

### **Custom Attack Clients**

I think full blown remote administration clients are a little heavyweight for a good attack on ecommerce. A professional attacker would want a custom remote client that just watched for SSL web connections and recorded the IP addresses that were visited and all the keystrokes input by the user at that time. Every few days it would then encrypt the data and send it out anonymously. This could be by posting to IRC, Usenet, a web bulletin board. Who knows? A good system would have many fallback methods of posting the data so the attacker can pick it up.

The easiest way to get something like a custom attack client to spread is to use a technique like the happy99 virus method. It is just an executable email attachment. It modifies the TCP/IP stack on any system it is run on so that it will send out a copy of itself to everyone that you send email to.

### **Antivirus Solution**

Even with antivirus vendors detecting a trojan it usually takes a week or so before they have a handle on it. By then the thing has spread pretty far. There are also many people who do not update their virus definitions in a timely manner or even run antivirus software at all. Antivirus software is not really a good solution to the problem.

### **Banks and Online Merchants Full Speed Ahead**

So we have this great big client security problem, yet banks and merchants are relying on the infrastructure their customers have to lower their transaction costs. This is so enticing that they are willing to overlook the client security problem and pretend they are dealing with dumb terminals. They will say their system demarcation line is the end of the encrypted channel. They will require 128-bit SSL but have no requirements on the security of the client system. That is the customer's problem.

### **Assurances to authenticate customer**

This flies in the face of one of the assurances banks and merchants have which is to authenticate the customer in a transaction. How can they assure that they are authenticating the customer when a large percentage of their customers are vulnerable to having their authentication credentials stolen?

### **Online Banking-Bill Paying**

The greatest present risk is online banking. Customers with online bill paying can cut checks for thousands of dollars. Usually the attacker just needs to keystroke log an account number and a password. They can then assume the customers identity from anywhere in the world and write themselves a check.

### **Customer Liability**

We all know about the limited customer liability on credit card transactions and this is one of the reasons they are so popular, even on the net. With online banking there is no limited liability. How is a customer going to prove they didn't write a check for \$1000 to Jane Doe? Weren't they careful not to give out their password? An attacker can even make the transaction come from the compromised machine by using network redirection. This will make it very difficult for a customer to prove they didn't do it.

### **Current OS not ready**

I would say the current state of security on the client side of internet commerce is not ready for prime time. What can be done to alleviate these problems? Waiting for everyone to move to a new operating system is not a solution. There doesn't seem to be any band aid such as just in time virus definition updates that will really solve the problem.

### **Tunnel Authentication through OS**

The only thing I think will work is to take the authentication outside of the operating system and in effect tunnel through this untrusted medium. This is the approach that is taken to move the authentication information through the internet via SSL. It needs to be taken a step further through the OS. A technology that can do this is public key cryptography with the private key stored outside of the OS in hardware. A hardware token such as a smart card would store the private key and sign any challenge response authentication.

### **Attacker cannot get at authentication credentials**

With the authentication moved outside of the realm of keystroke loggers or the file system an attacker could not steal the authentication information. The best he or she could do is to use an authenticated session. Customers

would need to wary of logging out of the ecommerce system and removing their authentication token.

### **Digital Certificates alone not the answer**

One thing that should be noted is a public key cryptosystem such as x.509 client certificates does not in itself add any layer of security if the private key is stored on the hard disk in the clear or is unlocked by a key stroked password. An attacker can have full control of the file system and can be logging keys. The underlying client operating system cannot be trusted to secure the client certificate.

I don't know how quickly hardware based crypto will take to become widespread. Smart cards and readers for PCs that plug into keyboards or floppy drives are coming down in price. It should be less than \$100 per customer soon. Perhaps banks could start offering this soon as a high security option. But if customers have to pay more they probably won't.

I think we will see a few more years of the current model before people start to change. Or maybe there will be some high profile attack that will wake people up. From my experience that is the only way to get people to change in the security world. That is one of the reasons the L0pht slogan is "Making the theoretical practical." We realized that to get people to change you had to be able to demonstrate the threat and not just talk about it.