Bypassing Secure Web Transactions via DNS Corruption

A man-in-the-middle attack

by Coretez Giovanni
April 27, 1999

This paper has been written to inform the general public in weaknesses of secure communications via a secure socket layer, commonly referred to as a Secure Web Transactions. This paper addresses the most common configuration of a "secure transaction". It is intended not to be a how-to on the subject, but to draw attention to the needs of improved security to protect people's privacy on the Internet.

People have often asked, "Is banking online a safe thing?" The normal response in an FAQ has been that if your system is using common US encryption (128-bits strong) that your transaction could not be intercepted and deciphered. This might be true (at this time 64 bit encryption took 2-days to break), but an intruder does not need to "break" the encryption to get your account information.

A Domain Name Service (DNS) is a common Internet protocol that allows a user to type the URL (name) of the destination into their browser (telnet, ftp, you name it program) and receive the ambiguous IP address number to initiate a TCP/IP connection with a desired host. DNS is not unlike a telephone book where one can look up the name of an individual and receive a phone number or address to contact someone.

For example, when you connect to a bank online you type the name into the browser. The browser sends a domain request to the name server that returns the IP number to the browser software. The browser begins a TPC/IP connection using this IP. A message to the user is given that they are about to enter a secure connection. The two systems send their 128-bit strong public keys to each other. And then a message conversation begins on the Internet that is impossible to crack within a debatable 20 years.

Once a secure communication is established, the bank then requests the user to authenticate who they are by using an bank account number and personal identification number (PIN). With these two items of information the user can see their account, transfer money and pay bills.

So what is the problem with this scheme? If this encryption takes so long to crack then is this not a safe means of doing business on the Internet?

The first weakness is that this encrypted communication trusts the IP address received from the DNS to be correct. The DNS is not in the control of the user or their bank. The fact is that there is no Identification and Authentication (I&A) mechanism to the domain protocol to ensure the desired address.

After the connection is established the authentication between the user and the bank is *one-way*. One way authentication means that user does not validate they are connecting to the

bank's system, instead the bank validates that the user is who they say they are. This is done with the account number and PIN.

Man-in-the-Middle Attack

The following is an example of a man-in-the-middle attack. This term refers to any attack where a second element (person, system, or application) performs a communication while masquerading as the intended destination. A DNS man-in-the-middle attack can occur as follows:

An intruder (or a corrupt Internet system administrator) changes the name of your bank's IP number in the DNS table to be a machine controlled by the bad guy, which we'll call EVILSYSTEM. When you type "WWW.SecureOnLineBankingSystem.COM" into your browser the compromised DNS now returns the IP address of EVILSYSTEM. EVILSYSTEM system responds to the browser by sending its public key. At the same time EVILSYSTEM opens a connection to the real banking system by using the IP address that is in its internal host table instead of the incorrect one in the DNS table. Now there is a secure connection from the user to EVILSYSTEM and EVILSYSTEM to the bank.

EVILSYSTEM forwards the bank page back to the user, and the user enters in the account number and PIN. EVILSYSTEM then forwards that information back to the bank system after copying the user's information. EVILSYSTEM acts as a mediator capturing all the critical information during the transaction. There are no obvious signs to the user that they are not connected solely to the bank.

The Real Problem

There are a number of counter-measures that a user can do, like hard coding the IP address. But there are a number of hacks that allow an aggressor to remain one step ahead:

- Inserting a corrupted host table into user's system using BackOrifice or another Windows hacking tool (these can be inserted using any EXE file to a DOS system and having the end user play the EXE. Such an example would be any number of holiday executable cards sent via e-mail). This works since the user system will check the host table if one exists before the system checks a remote DNS.

- Changing how the router routes information, allowing the traffic to flow by a compromised system that hijacks the session and acts as a mediator in the exchange of the DNS information.

The problem is not truly in the DNS as much as it is in the *Authentication and Identification mechanism* being used.

Mutual-Authentication

In this example, and in most cases of logging into systems, the user presumes that they are talking to the correct system. A user must identify and authenticate themselves to the system, but the system does not authenticate itself to the user in an obvious way. The problem is only compounded with an increasing number of vulnerabilities in the TCP/IP protocol suite that can create misinformation to an aggressor's advantage.

To resolve this problem of man-in-the-middle, a proper mutual authentication mechanism needs to be in place. Mutual authentication is when the host authenticates the client, and the client authenticates the host. In the previous example the client fails to authenticate the host. This lack of authenticating a host is a common weakness to systems that can be attacked with misinformation and man-in-the-middle attacks.

Mutual Authentication is currently being addressed through the technique of digital signatures and third party companies.