Why Enterprises Need More than Firewalls and Intrusion Detection Systems

Mark Vandenwauver  
Joris Claessens  
Wim Moreau  
Katholieke Universiteit Leuven  
Dept. Elektrotechniek, ESAT-COSIC  
Kardinaal Mercierlaan 94  
B-3001 Heverlee - BELGIUM  
mark.vandenwauver@esat.kuleuven.ac.be

Calin Vaduva  
Robert Maier  
Technical University of Cluj-Napoca  
Baritiu 26-28  
3400, Cluj-Napoca, Cluj  
ROMANIA  
vaduva@el.el.obs.utcluj.ro

Abstract

At the approach of the next millennium, enterprises are facing a lot of challenges and have to decide in which direction their communication networks will evolve. At the moment we see a large shift towards using global intranets and extranets. Most of the time it is also necessary to connect these networks to the Internet. During the consultancy work we provided in this area we concluded that although there are good tools available to protect these intranets from external attackers, perfect security could not be obtained. Therefore we started the TACK project to illustrate this by working out attacks that could be launched at any intranet. In these attacks we use the technology that is on the one hand rendering the Internet extremely successful but on the other hand facilitates the work for attackers. This paper summarizes the results of this project. We conclude that fundamentally no intranet can be made completely secure.

Keywords: Active Content, Firewall, Intranet, Intrusion Detection Systems, Java, JavaScript.

1. Introduction

Many enterprises have already connected their intranet (internal network) to the Internet, and many others are starting to do the same thing. They want to expand their businesses into the world of electronic commerce, or they want to connect their different sites across the globe. Browsers such as Netscape Communicator and Internet Explorer have become standard tools that are used by all employees and customers to surf the web and read their email. The support of the Java [7] and JavaScript [6] technologies further enhances their functionality.

Firewalls and intrusion detection systems are installed to protect the enterprise’s intranet from the untrusted Internet. However, this paper shows that, despite the presence of these two (or even additional) excellent security measures, security remains a huge problem. This is mainly because the technologies that provide extra functionality also cause a possible security breach.

The remainder of this paper is set out as follows. We start with a general overview of firewalls and intrusion detection systems. Then we discuss the TACK project which was started to show the deficiencies of firewall and intrusion detection systems. The next two sections detail the attacks that we are able to mount. We finish with our conclusion.

2. Firewalls

Firewalls are a relatively recent phenomenon and are perhaps the most used of all security technologies. In their short history they have evolved tremendously: the first firewalls were simple packet filters for controlling access between networks. Now they are expected to provide numerous security functions [10]:

- Track and maintain state between multiple sessions;
- Support low overhead protocols for multimedia needs such as UDP, and broadcast protocols;
- Authenticate individual users using tokens or one-time passwords;
- Encrypt/decrypt traffic passing through the firewall;
- Create secure private tunnels and virtual networks through untrusted networks;
- Support IP address translation: to provide another layer of security by making the computers attached to the intranet non-routable on the Internet (using private addresses), and to address the shortage of IP addresses;
• Protect against viruses, ActiveX and Java (although this paper shows they are far from perfect);
• Ensure its own security is not compromised.

2.1. Inter-network Access Control

A firewall is a component or set of components that primarily restricts access between a protected network and the Internet, or between other sets of networks [2]. The firewall system controls access based on configured rules. These rules are designed to enforce an organization’s security policy.

Figure 1 shows a typical firewall system separating an organization’s network from the Internet. The aim of this firewall system is to restrict and monitor access between the Internet and internal hosts. At a minimum, the organization requires a firewall system for each connection to the Internet.

The advantages of a firewall system are:

• The organization can concentrate its security efforts at each connection point, and ensure the firewall system is as secure as possible, rather than trying to secure every system on the internal network (this is one of the reasons our attacks are very successful).
• All traffic between the Internet and the internal network can be regulated and monitored to ensure it meets the organization’s policy.

It should be noted however that no system (including a properly configured firewall system) is absolutely secure. Firewall systems are often very large and complex software and hardware configurations and it is unwise to rely completely on this system as they are usually built from untrusted components. Hence system administrators should still ensure at least reasonable efforts have been made to secure internal hosts.

2.2. Structure of a Firewall System

A firewall system should be designed with two goals:

1. To protect itself from attack.
2. To protect the network behind the firewall system from attack.

To provide such protection firewall systems are normally designed with a defense in depth principal. That is, multiple mechanisms are used that back each other up. This principal is not new to security, for example a car usually has door locks, and an ignition lock.

There are numerous configurations for firewall systems implementing the defense in depth strategy. For an overview we refer to Chapman’s work [2].

3. Intrusion Detection Systems

Any action whose target is to compromise the availability, utility, integrity, authenticity, confidentiality and/or possession of an information system can be defined as an intrusion [11]. An intrusion have different targets:

• Read protected information such as internal company data, credit cards numbers, financial records or password files;
• Change protected information such as changing file contents, deleting files;
• Use protected computer resources: CPU time, disk space, printer, audio-video tools;
• Denial of service;
• Use a computer’s identity. Use the computer as an agent to attack other systems behind a firewall.

Any type of information can be very useful for a hacker to penetrate an intranet, even something that looks unimportant. Users should therefore be aware about the kind of information that must be protected. Moreover any resources or information that do not need to be public in order for the system to work, should be kept private.

Considering the complexity of the matter, defending from an attack is actually very difficult. In order to help protecting information systems, security companies are developing elaborated tools. These tools are based on complex methodologies and algorithms. Their task is to detect intrusive actions. Intrusion detection is defined as any action whose purpose is to detect any intrusive activity. The whole system that fulfills this task is called an Intrusion Detection System (IDS). An IDS runs continuously on a system to try to detect any suspicious action and consequently to notify the security administrator.

There are some particular methods an IDS may use to detect intrusion. These methods can be classified as [11]:

Appeared in Proceedings of the Eighth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises
Stanford, California, June 16-18, 1999, pp 152–157
0-7695-0365-9/99 $10.00 ©1999 IEEE
Anomaly intrusion detection: in this case, the IDS is looking for any activity in the system that has an anomalous behaviour. The IDS knows the normal behaviour and attempts to detect a possible intrusion based on the presence of any abnormal behaviour. This kind of IDS is able to adapt to new intrusion methods but it may fail because not every intrusive activity is anomalous.

Misuse intrusion detection: the IDS uses known attack patterns and attempts to detect these patterns. This type of IDS is very efficient for attacks that fit the patterns but are completely not effective for attacks whose pattern is not implemented.

Combined intrusion detection: an IDS that uses both aforementioned techniques.

There are many IDS systems developed by different companies but not a single one can be perfect. There are two types of errors that may appear. The first and the most serious is called *false negative* and appears when the IDS does not detect an intrusive action. The second is called *false positive*, and appears when the IDS detect a false intrusion. The first error is very serious as it compromises the whole system. The second one can become serious in case that it appears too often and the security administrator comes to ignore the output of the system over time. For an excellent overview of IDS we refer to Escamilla’s work [4].

4. The TACK Project

The TACK project was started in 1999 based on the experience gathered by performing audits and consulting jobs on real intranets. For our project we targeted personal computers running a Microsoft operating system (95/98) since they are widely used inside companies. We considered the case where the firewall only allows communication on ports 25 (SMTP) and 80 (HTTP) (443 when using HTTPS). This configuration is quite paranoia and in reality we encountered systems that allow more traffic coming in and out. We used several IDS systems and although some of them were able to track some of our attacks, it was possible to find a solution and circumvent them.

4.1. Methodology

Our attacks consist of one or more of the following steps:

- Gather information. The attack looks for information on the targeted computer or stays resident and monitors the user to obtain valuable data.
- Communicate information. The attack sends out information (e.g. gathered by the previous attack) to the external hacker. In our case we mainly use ports 25 and 80 (443) to send out information.
- Change information. The attacker deliberately changes files and/or settings on the local computer. In this way an attacker can change the local configuration to bypass security functions or opens up backdoors to have access to the intranet in the future.
- Use this computer to attack other computers attached to the intranet.

5. Mail Based Attacks

In these types of attack the medium we use is electronic mail. Nowadays people have become accustomed to all the whistles and bells of HTML enriched emails and therefore use mail readers such as Eudora, Netscape Messenger or MS Outlook (Express). These modern readers undoubtedly offer many good services but at the same time constitute a big risk. We show in the following paragraphs how these readers can be (ab)used to attack an intranet.

5.1. Attached executables

The first type of mail attack only uses standard executables (including batch files). They are sent as an attachment and the user is lured into executing them (by promising nice features or gadgets).

These executables have access to all the resources of the system (because of the internal structure of Win95/98). A trojan horse approach is needed, because the user has to be persuaded to run the executable. In batch files, the malicious code can be hidden by putting the instructions behind column 80 (or even further) so even when the users edit the batch file, they will see nothing at first sight. Information gathering and changing are very easy to perform once the executable is running because all the resources of the system can be accessed. An infamous example is the powerful Back Orifice trojan program [1].

When mail is downloaded first and read while disconnected from the Internet, communicating information to the attacker might be a little trickier, as the trojan horse has to wait until the user re-establishes the network connection.

5.2. JavaScript

In our second type of attack we use JavaScript [6]. Mail readers such as Netscape Messenger automatically open and execute this code. The users do not even notice the presence of JavaScript code in their email. JavaScript can control the browser behavior and the content of the displayed page, so attacks such as the Hotmail incident [5] are possible.
The Hotmail incident occurred in August 1998 and it exploited a weakness of the free Hotmail service (it didn’t filter out JavaScript from incoming mail). It could also be directed to any other email service that has the same weakness. The attacks consists of altering the user interface to the web server. This is possible because it is based on a web-based user interface which JavaScript can control. The result of the attack is that users are lured into believing they have to re-login because of a timeout and in this way it is possible to steal their username and password.

Although JavaScript is not as powerful as Java, it can do the following:

- Control the browser behavior and the content of pages. Some of these controls can be performed only if the script acquires additional privileges from the user. The content of pages can be controlled without privileges.
- Interact and communicate with Java applets embedded in a web page. Java applets (although they need to be retrieved from a web page and cannot be embedded) are executed. The class hierarchy of Java can be accessed from JavaScript, hence all the capabilities of Java are accessible to it.

However with JavaScript it is impossible to:

- Perform direct networking. The only networking access it can achieve is by causing the browser to download arbitrary URLs and send the contents of forms to CGI (Common Gateway Interface) [3] scripts, email addresses and Usenet groups.
- Read or write files. JavaScript can read or write cookies which are kept in a file, but it does not have control over the reading/writing process itself.

One of the benefits of using JavaScript is that it is platform independent and thus other operating systems can be targeted as well. JavaScript can be disabled from the browser (user preferences), but is enabled by default.

5.3. Example

Our example is based on the idea of luring the user into believing that they accidentally (or timeout related) activated the screensaver and in order to continue working have to type in their password.

The example is implemented in several steps:

1. The only privilege the script needs is the UniversalBrowserWrite privilege. This request appears to the user as a high risk so the reason for asking such a privilege should be really good. The privilege request only reveals to the user that some Java applet or JavaScript needs the permission to modify the browser. The user is deceived into granting the privilege by pretending to offer a nice feature which needs the specific privilege. On top of this the JavaScript is also digitally signed with a spoofed certificate from a renowned software issuer.

2. The decoy for our attack consists of opening up a browser window which contains interesting information or sales figures from a competitor. While the user is reading this information the next step in the attack is executed.

3. The screen is blanked opening a new browser window. This window has no titlebar and its size is larger than the screen, its background is black, so all the user sees is the black part of the window. This gives the users the illusion of a perfect blank screen. The privilege needed is the one obtained in the first step.

4. When the user moves the mouse or types a key, a password prompt appears that looks exactly like the standard windows prompt.

5. The password obtained in this way is sent out using a hidden form submission mechanism. In order not to be forced to ask for another privilege (UniversalSendMail) the script uses a free mail service (a CGI script that mails the input of a form), because in this case the email address of the user won’t be revealed and thus no extra privilege is required. In this way the identity of the attacker is also hidden by the fact that the information is not sent directly to him.

Disadvantages of this attack are that it might be noticed by an experienced user who does not use a screensaver or uses one that is not password protected, or that the attack only works for Win 95/98 machines and for Java-enabled mail readers (Netscape Messenger, MS Outlook 98).

6. Web Based Attacks

Mobile code and in particular Java is currently used more and more on the Internet. Java is very popular. This is due to the fact that it is easy to write code, the code is platform independent and the language uses a security model. The Java security architecture [9] is based on several levels of protection: the semantic level, the bytecode verifier, the class loader, and finally the security manager. Currently, different browsers support different security policies. In some cases the user is allowed to configure these policies, thus granting special permissions to Java signed code or Java code originating from specific sites.
If attackers want to gain access to the user’s machine, they can either exploit bugs in the current version of the browser and its mobile code engine, or just politely ask the user. The first approach would certainly allow them to attack the machine without the knowledge of the user. Although these bugs undoubtedly form a serious threat, as soon as they are discovered and made public, patches are being developed and released within days, and the attack is rendered impossible.

Our approach is not to break any of the levels of Java security but to deceive the client user in order to grant special permissions to our Java code (applet). Using these permissions our applet can initiate different attack procedures to break into the client system. This approach is based on the idea that most of the users are either not well informed about security or just do not care about it.

6.1. Approach

Our Java based attack consists of the following steps.

1. Attracting the user to the web page.

In the mail based attack, the attacker takes the initiative, and can therefore target specific users and hosts. In the web based attack only random hosts can be attacked, unless the attacker is able to attract the target. This can be achieved quite simple by an email and/or setting up a page within the target user’s interest.

2. Identify the client system and adapt our Java code to that particular system.

In order to make the attack more successful, the characteristics of the client system have to be identified. Part of this identification can be done via the web server or a CGI script. Web browsers communicate their type and operating system in an http request, as well as their (or their proxy’s) IP address. Using this information, the server could make a decision whether to send an attack applet and/or what kind. This information can also be collected by a Java applet without requiring any special permissions. This includes the client operating system, IP address, browser and Java version. Some information (e.g. whether the browser is using a proxy) can be obtained by trial and error.

Both approaches are suitable, though a combined approach yields the best results, since then more data about the environment can be obtained. To convey the information collected during the attack, the applet also has to know whether the client is behind a firewall, and whether a proxy has to be used.

3. Deceiving the User by Spooing Certificates.

Since the Java security model executes an unsigned applet in the sandbox (if no extra privileges are granted to it), we need our applet to be signed. When an applet is signed it can request each privilege to the user (either when it needs them, or beforehand). At the moment no browser implementations offer fine-grained access privileges (e.g. read permission on a per file basis).

The users have to be deceived that the privileges they are granting are needed for something plausible and desirable, in order not to raise suspicion. Any code used for the attack should also be hidden inside the other code. An obfuscator makes it also more difficult to analyze the applet itself.

Certificate spooing can be used to deceive users even more. Certificate spooing is explained in [8], where it is used for spooing web server certificates. The basic problem is that browsers contain multiple root certificates, and it is easy to add others.

This can then be misused in the following way. The attacker first lures the user in installing its root certificate into the browser (in [8] the attacker is the administrator of the local intranet, and the root certificate is already part of the standard installation). The attacker then issues a false server certificate. When the user visits the spoofed site of the attacker, the browser does not give any warning as the false certificate is trusted. However, when the user explicitly checks the certificate, it turns out that it has been issued by the wrong authority. To circumvent this, the attacker can issue a false root cross certificate and use that to issue the false server certificate. When explicitly checking the certificate, users only see the first certificate of the chain, and notice nothing special, unless they validate the fingerprint of the certificate.

These techniques can also be used for spoofing object signing certificates or spooing certificates used for signing email. As such, they are used to improve our attacks.

4. Attacking the System with a Java Applet.

Gathering information is the first objective of the attack. Valuable information includes the directory structure, the OS’s configuration files, browser configuration files, location and contents of password files, which software is installed, the available hardware devices, local network information, information exchanged over the network while the applet is running. Files the user mostly works on, . . . In order to acquire this information the user has to grant special permissions to our applet.

In a second step we have to decide how to send back this information to the attacker. We expect the user’s host to be located behind a firewall, and http is possibly shielded by a proxy. Therefore we only use HTTP or SMTP to port 80 and 25 respectively on the attacker’s applet server. Encrypting
or encoding the information to be sent out would improve the attack, since it can deceive potential content checking software.

The attacker uses an anonymous email account (an alternative is to send the encrypted information to a news group or mailinglist) and a web page on a free web host, in order to obscure their identity. Obscuring the identity is not necessary for an attack to work, since it’s most important use is to confound and stall the victim after the attack is finished.

The final step is to modify the client system. Since the applet obtained privileges to write files, all sorts of attacks are possible: change the operating system’s configuration files, put Java class files in the user’s class path, install new native background applications (in order to start another attack after the applet is closed), modify registry files, access different hardware devices, . . . . It is even possible to change the browser configuration files in order to give a.o. more permissions to external Java code.

An additional step in the attack may involve using the client machine to attack other machines attached to the local intranet.

6.2. Example

Our example attack applet consists of a program that performs an installation procedure for an application. We chose a screensaver with the possibility of choosing theme packs. To install the screensaver, it is quite plausible to ask for the UniversalFileWritePrivilege. At the same time we ask for the UniversalConnectPrivilege, under the presumption that the different theme packs reside on different servers (due to the limited space on each of our free hosts). By doing this, we actually get total access to the victim’s computer.

The information gathered, is communicated through an URLConnection, disguised as an innocently looking http GET command. In this way the browser itself initiates the communication and we use the victim’s proxy to get through the firewall. The applet itself can decide if a proxy is in place and it can (with the use of some information provided by a CGI script running on the server) get its address as well. At the moment, this information is not used, but it would be possible to set up a connection using a socket.

In order for the URLConnection to work, we need access to some form of server (CGI) scripting. It should be noted that any email-sending CGI is sufficient. In that case, it is not even necessary to ask for the UniversalConnectPrivilege, since we can communicate the gathered information to the originating host. One variation of our attack applet consequently does not use the UniversalConnectPrivilege.

It is worth noting that the applet is quite generic, and thus can be used to install all sorts of programs. In this way another user can be attacked quite easily. In the above example, we can broaden our attacked audience by providing more theme packs as well.

7. Conclusion

The success of intranets have made them a potential target for attackers. Usually these intranets contain information that is essential to the future of the enterprise. Firewalls and intrusion detection systems provide excellent tools to secure these intranets but at the same time give a false sense of security.

The TACK project has shown that it remains possible to access the intranet and obtain valuable information. The reason for this success is two-fold: users are easy to manipulate and deceive, and Java and JavaScript provide the attackers with a set of tools previously unknown.

To counter these potential threats it is important that the issue of trust is centralized within enterprises. The installation and maintenance of web browsers can not be attributed to the end user. Moreover the only current solution to prevent our attacks is to disallow the use of Java and JavaScript.

References
