WEBARROW: A CASE STUDY OF SECURE WEB DEPLOYMENT

Namzak Labs White Paper, 2002-02

Version 1

September 30, 2002

Overview

As deployment of computer applications over the Internet becomes more prevalent, companies face the difficulty of deciding how to take advantage of the features of such applications without exposing themselves to security risks. This paper helps companies to understand the risks of Internet deployment, and to understand how to evaluate how well companies address these risks.

We first present an overview of the theory and state-of-the-practice of Internet security, covering the major topics of encryption, firewalls, authentication, and digital certificates.

In addition to the theory, we present a practical case study, showing how practical security considerations were addressed in a real product's architecture and its deployment. The product is Namzak Labs' ***WebArrow***.

CONTENTS

1. INTRODUCTION

Computer security issues are large and growing larger with every passing year. The annual CSI/FBI Computer Crime and Security Survey of 538 organizations shows the following surprising (and daunting) statistics:

- 85% detected an attack in the preceding year.
- 64% had financial losses. The biggest losses were theft of proprietary information and financial fraud.
- Internet attacks are more frequent than internal attacks (70% versus 31%)— a reversal of patterns in the earliest years of the survey (which began in 1996).
- Only 36% reported intrusions to law enforcement agencies.

Source: Computer Security Institute, "2001 CSI/FBI Computer Crime and Security Survey," Computer Security Issues and Trends, Vol. VII, No, 1, 2001.

In addition, spending on computer security is steadily rising. The estimates vary, but according to at least one survey of security spending, security spending worldwide was about $5.8 billion dollars in the year 2000 and it is predicted to rise to about $20 billion by the year 2004. The compound annual growth rate (CAGR) of security spending is an enormous 36%.

Spending Area	2000	2004	CAGR
Core security	$2,029	$5,519	37%
Access control	$1,365	$5,074	48%
Incident response	$858	$4,107	39%
Planning and management	$1,420	$5,019	28%
Total	$5,672	$19,719	36%

Table 1: The Size of Security Spending (in millions of $)

Source: Mike Fratto, "The Survivor's Guide to 2001: Security," Network Computing, December 11, 2000. http://www.networkcomputing.com/1125/1125security1.html

Clearly this is a major problem for industry. In the past, most security breaches were internal, caused by employees and ex-employees. Today the major threats are external: hackers breaking into organizations from the outside, typically via the internet. So anyone building or deploying an internet-based product or service today must be well schooled in the types of threats that they face, and the counter-measures available to defend against those threats.

The good news is that, with some careful planning and attention to detail, most threats can be successfully warded off.

1.1 NETWORK SECURITY THREATS

The major network security threats that an organization faces today fall into 5 categories: interception, impersonation, denial of service, root user attacks, and content threats. We will briefly introduce each of these threats.

Interception occurs when a third party intercepts messages sent from or received by your organization. If the interceptor cannot read these messages, you have *confidentiality* (and hence privacy). If the interceptor cannot modify these messages without detection, you have *message integrity*.

Impersonation occurs when an impostor claims to be someone else, for example, when an impostor claims to be a legitimate employee or the legitimate holder of a credit card number. To guard against impersonation you need to *authenticate* the sender— prove that they are who they claim to be.

Denial of Service (DOS) attacks occur when an attacker overloads a system with a flood of messages or when an attacker sends a single message that crashes the system. In either case the legitimate users of the system are denied the services of the system for some period of time.

Root user attacks occur when an intruder is able to remotely login to a system as a root user. This requires cracking the root login password. Once this password has been cracked the intruder is able to remotely control the system. Once in control, the intruder may read private system information, may steal information, may damage data (e.g. erase the hard disk) or corrupt existing data, or may create a backdoor account that will let the intruder easily gain access to the system at a later time (perhaps to use as a launching pad for further attacks on other systems).

Content threats occur within applications that run on the system. The content of malicious or infected applications may cause severe problems, such as the installation of viruses. In many ways, viruses are the most severe security problem in corporations today. To guard against viruses you must examine the contents of application messages (at the minimum you must examine the extensions of files or the MIME type of data being transferred). Another, less well-known form of a content threat is the *Trojan horse*. A Trojan horse is an application that allows an intruder internal access to a system, without the "host" knowing that this intruder has this access.

To guard against these security threats we employ a number of techniques including encryption, firewalls, passwords, and digital certificates. These topics will be discussed in the next few sections of this document.

2. ENCRYPTION FOR CONFIDENTIALITY

The purpose of *encryption* is to transform a message so that an interceptor cannot read it. For confidentiality purposes, you typically want to make it extremely difficult for an interceptor

to read your messages (containing, for example, financial data or personal messages). The encryption process is sketched in Figure 1.

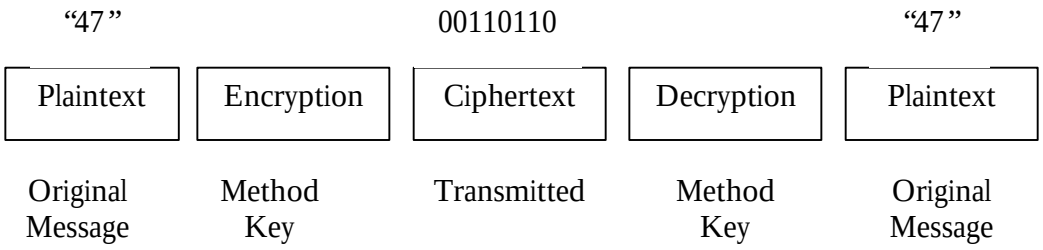


Figure 1: The Encryption Process

In this figure "Plaintext" is the original message, say the number "47". The ciphertext is a transformed version of the plaintext message that is suitably encoded for confidential transmission. An encryption method is used to transform the original method into a stream of ones and zeros. In this form an interceptor cannot easily make sense of it. The receiver must then decrypt the ciphertext back to the plaintext, using some decryption method.

Encryption requires a method and a key. The encryption method is the specific mathematical process used to transform a message into a stream of ones and zeros. The key is a string of bits used in the method. Applying different keys to the same plaintext message results in different ciphertexts. The method cannot be kept secret if the encryption is to be widely used. Encryption techniques are available, for example, on everyone's web browser. The key, on the other hand, *must* be kept secret.

No key, and no encryption method, is perfect. Keys can be "guessed" by exhaustive search. An attacker merely has to try all possible keys to see which one decrypts the message. However, by using very long keys the process of exhaustive search is made difficult, if not impossible to achieve in a timely fashion. If the key length is n bits, then the attacker needs 2^n tries (maximum) to guess the string. Today, keys of less than 100 bits are considered "weak". Encryption methods that use more than 100 bits are considered "strong" (e.g. 2^{128} = 18,446,744,073,709,551,616). Tomorrow, of course, we will need even longer keys. This is an arms race, and there is no such thing as a permanent victory.

2.1 METHODS AND ALGORITHMS

There are two common ways of doing encryption: symmetric key or public key encryption. Within each of these there are different encryption method algorithms. These algorithms are specific ways of doing encryption. For example, for *symmetric key encryption*: there are algorithms such as DES, 3DES, AES, IDEA, Blowfish, Twofish and RC5. For p*ublic key encryption*: the RSA, elliptical curve cryptosystem (ECC), and Diffie-Hellman algorithms are the most common.

Symmetric Key Encryption

In symmetric key encryption, both sides use a *single* key to encrypt & decrypt a message: when A sends a message to B: A encrypts with the key, and B decrypts with the key; when B sends a message to A: B encrypts with the key, and A decrypts with the key. Symmetric key encryption and decryption processes are simple enough for fast encryption/decryption, and hence fast enough for long messages. However, there are some problems with this scheme:

- Problem 1: Symmetric key must be *distributed* secretly between partners or interceptors can read subsequent messages
- Problem 2: You need a *different* symmetric key for *each business partner* (or else one business partner could potentially read messages meant for another partner). This complicates the process of symmetric key distribution.

Public Key Encryption Methods

With public key encryption the keys for encryption and decryption are different, and everyone has both a public and private key. You keep the private key secret and distribute the public key to everybody . There is no security risk in distributing your public key.

To send a message, you encrypt with the receiver's public key, and the receiver decrypts with their private key. Thus even the sender cannot decrypt the ciphertext; the sender doesn't have receiver's private key. Then anyone can encrypt messages to you using your public key, but only you can decrypt the messages. The one drawback to this scheme is that it is typically highly processing-intensive (approximately 100 times slower than symmetric key encryption). So you typically only want to encrypt small messages. Another limitation is that you can only encrypt messages that are approximately the size of the public key (typically a few thousand bits) because of limitations in the algorithms.

2.2 COMBINING PUBLIC AND SYMMETRIC KEY METHODS

Symmetric key encryption and public key encryption each have their advantages and disadvantages. In fact, these advantages and disadvantages are largely complementary, so they may be combined in a way that capitalizes on both their strengths. This is done as follows.

Typically the communicating partners first communicate using public key encryption, including the initial authentication. Then one side, the "originator", generates a symmetric key. The originator encrypts the symmetric key using the partner's public key, and sends it to the partner. Now, both sides have the symmetric key. Afterward, both sides may communicate using the symmetric key. This symmetric "session key" is good only for this session– a single flow of communications, but it is highly efficient for that session and has the advantage that it is never used for any other communication (with a different partner).

3. AUTHENTICATION

Confidentiality, as we have previously discussed, exists when an interceptor cannot read your private messages. *Authentication*: is the process of proving the sender's identity, to deal with the problem of impostors. As we shall see, authentication, like confidentiality, often uses encryption.

Although some esoteric authentication methods have been proposed and are even in limited use (fingerprint analysis, iris analysis, etc.), these are relatively new and not standardized. The most common software-based authentication methods are described below.

3.1 PASSWORDS

The most common authentication method is the use of userids (which are typically public) and passwords (which are supposed to be private). The passwords are intended to authenticate the user (the owner of the user ID). But this is only meaningful if it is extremely difficult for an attacker to guess the password. As a consequence, mandating appropriately complex passwords is an important part of authentication security.

Short passwords are easy to guess with exhaustive search. Passwords that include common words or repetitive letter combinations are even easier to guess. To protect against users choosing such passwords, many systems employ exactly the same tools as attackers, *automated password cracking*. Typically an automated password cracking program is run on every machine where passwords are stored. If any of the passwords are "guessed" by the automated password cracking program then it is rejected and the user is forced to choose a new password. Sometimes passwords are pre-screened, meaning that a user is forced to re-enter a password until it meets some minimal standards of security. Users must be forced to pick long passwords containing case changes and numerals, such as **Tri6Vial.**

3.2 PUBLIC KEY AUTHENTICATION

In public key (challenge-response) authentication the idea is to prove that the sender holds their private key, which only they should know. It works as follows: the v*erifier* is the party who wishes the other party to authenticate themselves. The a*pplicant* is the other party, which wishes to prove its identity. The verifier sends the applicant a challenge message, a string of bits. The applicant sends back a response message; this is the challenge message encrypted with the applicant's private key. If the verifier can decode the response message using the applicant's public key, arriving at precisely the original string of bits, then it must be the case that the applicant encoded that message. This authenticates the applicant.

There is, however, an attack to which a public key authentication scheme is susceptible. This is called "public key deception". In public key deception an *impostor* claims to be a *true party*. The true party has a public and private key. The impostor also has a public and private key. The impostor sends its own public key to the verifier, stating, "This is the true party's public key" (This is the critical step in the deception). The impostor would be "authenticated" using any public key authentication method. To guard against such confidentiality attacks, we commonly use digital certificates.

3.3 DIGITAL CERTIFICATES

Digital certificates are created by a *certificate authority*. The certificate authority is a trusted third party. (Currently these certificate authorities are unregulated.) The certificate asserts that a true party (named) has the public key contained in the digital certificate. Thus the certificate provides a (name, public key) pair. This is sufficient information to prevent public key deception.

The fields and content of the digital certificate are standardized by the ITU-T *X.509* Standard. Note that the certificate does not vouch for the integrity of the named party— it only proves that their public key is really theirs.

Certificate authorities may revoke digital certificates. Revoked certificate ID numbers are placed in a *certificate revocation list (CRL).* The verifier must check with the certificate authority to determine if a digital certificate is on the CRL. Without the CRL check, digital certificates do not support authentication.

4. INTEGRATED SECURITY SYSTEMS

We have now described enough security "machinery" to sketch out a complete, integrated approach to software-based internet security. An extended security example, using Namzak Lab's ***WebArrow***, will be presented in section 7of this white paper.

When two parties communicate using an integrated security system, the goal is that their software handles all the security details. First the parties' software negotiates the security methods to be used. Once this is done:

1) They can authenticate one another
2) They can exchange symmetric session keys
3) They can communicate securely using symmetric session key and message-by-message authentication

Probably the most widely used integrated security system is SSL.

4.1 SSL INTEGRATED SECURITY SYSTEM

SSL stands for Secure Sockets Layer. SSL was originally developed by Netscape and is now supported by almost all web browsers (whenever you see a URL beginning with HTTPS you are employing SSL). As a result, SSL is now almost universal in Web financial transactions.

The SSL integrated security system is based on the following process:

1) The two sides negotiate security parameters;
2) The web server authenticates itself;
3) The browser may authenticate itself (but rarely does; typically it is the user who is being asked to provide confidential information, such as a credit card number, and so the main concern for the user is to know that the server is authenticated);

4) The browser generates a random symmetric session key, and sends this to the web server;

5) The server adds a digital signature and encrypts all messages with the symmetric key for the remainder of the session.

In this way, the browser and server have negotiated mutually satisfactory security parameters, and the user is typically uninvolved with the details of this process.

5. FIREWALLS

Another common network security component is a *firewall.* A firewall sits between an internal network (a corporate network or a home network) and the internet, preventing unauthorized access from the internet into the internal network. The firewall may also facilitate an internal users 'access to the internet.

There are two major categories of firewalls: packet filter firewalls and application (proxy) firewalls.

A packet filter firewall examines each incoming IP packet. Specifically, it examines the IP and TCP header fields. If bad behavior is detected (such as an attempt to access a port that should not be accessed), the firewall rejects the packet. A packet filter firewall usually maintains no knowledge of previous communication: it analyzes each packet in isolation.

Application (or proxy) firewalls, on the other hand, filter based on specific application behavior, such as known viruses, extensions of files transferred, etc. Application firewalls do not examine packets in isolation, as packet filter firewalls do; they use history as a guide. For example, in HTTP, as a rule you should not accept a response from a site unless an HTTP request has recently been issued to that site.

Application/proxy firewalls provide security in another way: they can serve to hide internal IP addresses through a process known as Network Address Translation (or NAT). With NAT, the firewall accepts a packet from an internal host. At this point the packet has the internal host's IP address. The proxy program replaces the internal IP address with another IP address, and sends that packet to some external host. The proxy server later receives a returning IP packet from the other IP address, and passes it on to the internal host. This provides security in the following way: an intruder with a packet sniffer program will only see the external IP address and hence will not learn the true internal IP addresses to identify potential victims.

Application firewalls have some disadvantages. You typically need a separate program (proxy) for each application. Not all applications have rules that allow filtering and hence not all applications are will benefit from this kind of firewall.

6. PUTTING IT ALL TOGETHER

The various hardware and software elements of network security, as we have discussed, are crucial, but they are only part of the solution. To have effective computer security you need to concern yourself with a host of other issues as well. For example:

- *Server Security:* occasionally, weaknesses are discovered in server operating systems. Hackers use these known security weaknesses to hack into or crash servers. Companies must therefore regularly install patches to fix these weaknesses. The sad truth is that they often fail to do so.
- *Client PC Security:* many known security weaknesses exist on client PCs, but patches may not be downloaded, either because users are not technologically sophisticated or do not understand the nature of the threats that they are facing. Also, users often have no passwords or weak passwords on their home computer. Given the vulnerabilities it can be easy for an adversary to take over a client PC and hence take over control over SSL and other (secure) communication protocols.
- *Managing Users:* Users often violate security procedures, making technical security worthless. For examples often an attacker tricks a user into violating security procedures, for example, revealing their user id and password.

For these reasons a total security policy is required in an organization. It must be understood that security is a complex techno-social issue. Therefore it must be addressed by a wide variety of technical and social policies and procedures. An organization needs to understand and manage every security technology and procedure that they have in place. And they need to train and manage their users.

7. ACHIEVING NETWORK SECURITY: *WEBARROW*

Security cannot be added as an afterthought. Applications must be designed from the ground up to provide state of the art security and privacy protection. To demonstrate this point, we will now show a practical example of how a high level of network security may be achieved in a practical product, if properly architected. The product is Namzak Labs '***WebArrow***.

7.1 TYPICAL DEPLOYMENT OF *WEBARROW* AND THE GRIZZLY CALL CENTER

A typical deployment of ***WebArrow***, and the Grizzly Call Center (***GrizzlyCC***) is shown in Figure 2 below. In this deployment a customer, located anywhere on the internet, wishes to contact an operator (for help, for sales information, or for any other collaboration purpose). Instead of contacting the operator directly, the customer contacts the ***GrizzlyCC***, via a web site and enters a queue. The ***GrizzlyCC*** informs the operator that a new customer has

arrived and is waiting for service. The operator may then choose to communicate directly with that customer, using one or more interaction modalities.

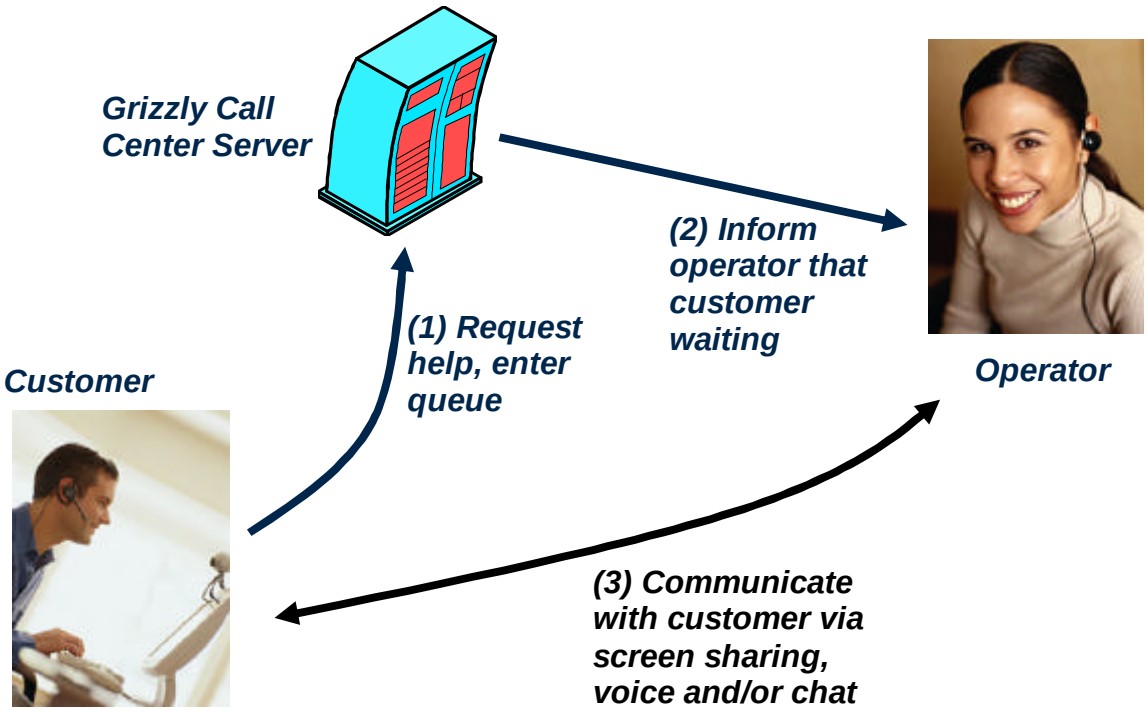
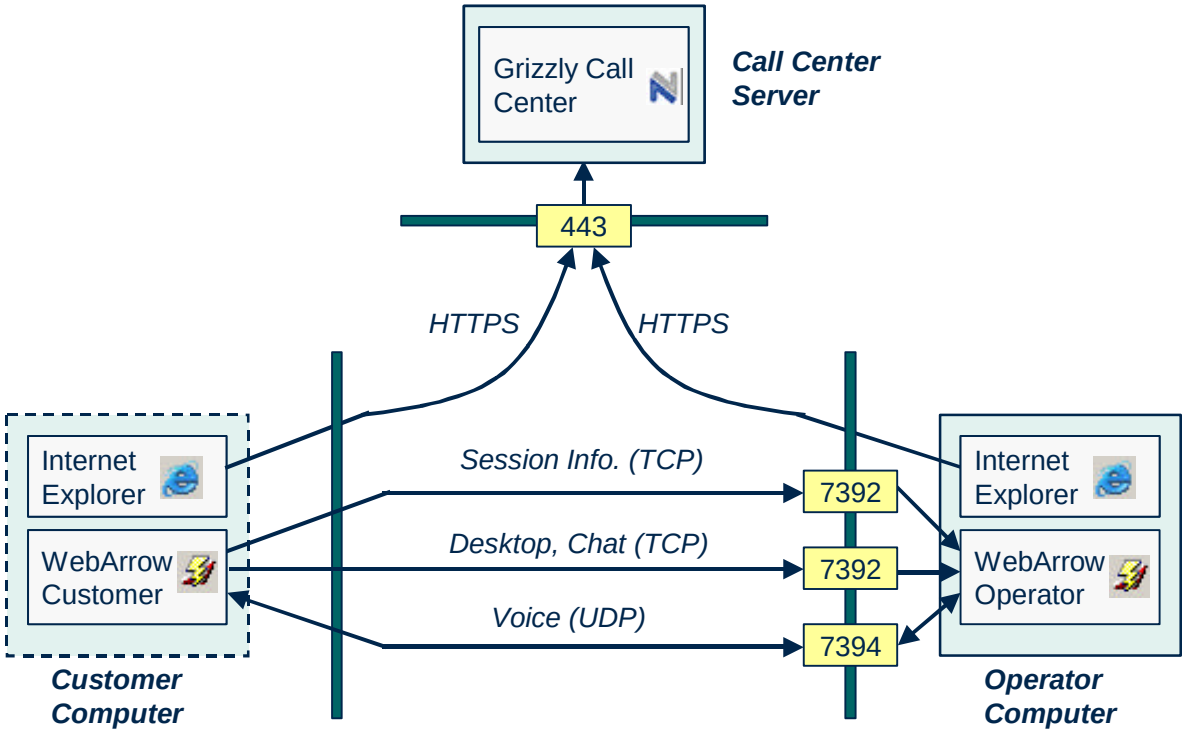


Figure 2: The Typical *WebArrow*/*GrizzlyCC* Deployment

If we look more deeply at how the communication is achieved, the picture looks like Figure 3, which shows some of the technical details of how the communication between customer, Call Center, and operator is achieved. When customers and operators log into the Call Center, HTTPS is used to securely transmit user id's and passwords. Following this, communication is based on the faster HTTP protocol. Therefore, customer-to-Call Center and Call Center-to-operator communication is done via HTTPS, via port 443, and subsequently HTTP, via port 80. Once the operator chooses to communicate with the customer, a direct peer-to-peer connection is established. The Call Center is not involved in this portion of the collaboration. Hence the customer and operator computers communicate directly via ports 7392 and 7394.

Note that the Call Center server may reside behind a corporate firewall. If it does, then it needs to have an external IP address and it needs to have its HTTP and HTTPS ports open. Otherwise it cannot function as a secure web server.

The customer may also reside behind firewall. The customer does not require an external IP address. However, the customer firewall must permit outbound TCP traffic over port 7392 and UDP traffic over port 7394. Typical firewalls permit such transmission, requiring no adjustment to firewall to run WebArrow as a customer. Finally, the operator may also reside behind firewall. If this is the case then the operator also requires an external IP address.

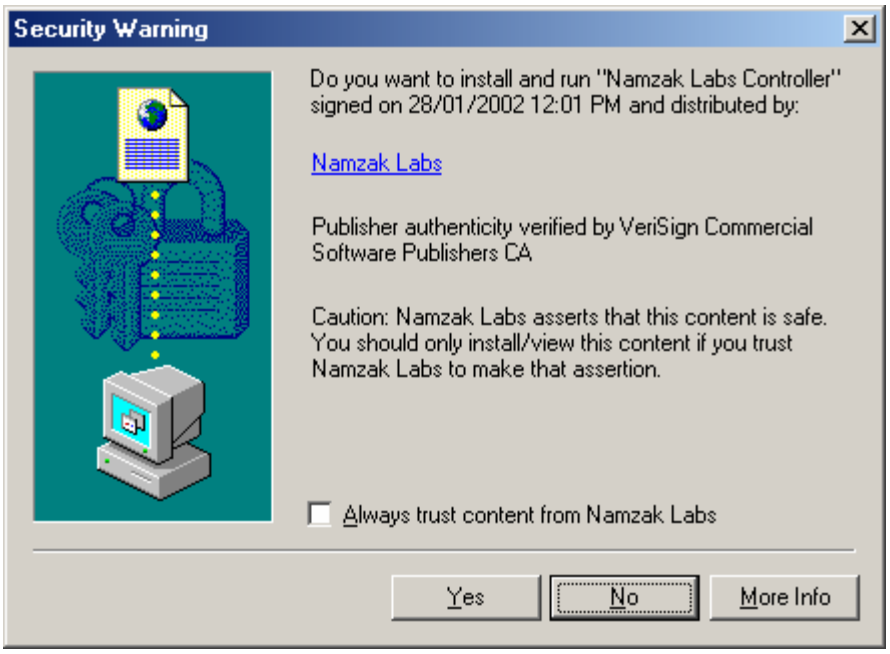


The principle at work here is simple: respect the firewall configurations of customers. If we want to collaborate with customers we cannot expect them to reconfigure their systems. We must accommodate their existing configurations and restrictions. And so any accommodation (such as providing an external IP address) is done on the operator side of the transaction.

7.2 Security Features for Customers

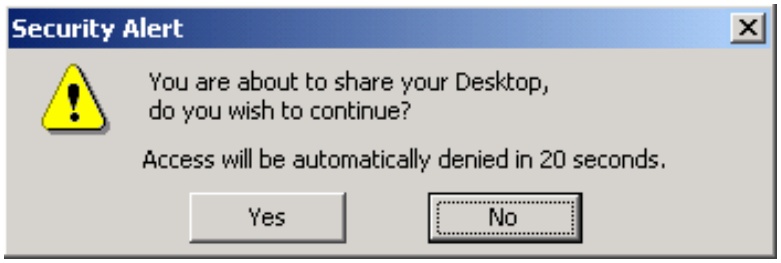
WebArrow was designed, from the ground up, to engender trust in customers. To be able to use ***WebArrow***, customers are asked to permit the download and installation of collaboration software. We must be able to assure them that this does not involve risk to their property or to their privacy. ***WebArrow*** features provide high security for customers.

- ***WebArrow*** *protects access to customer data*: the ***GrizzlyCC*** provides for the ability to issue userids and passwords to customers. Customer private data can be protected by requiring them to log in to the Call Center.
- *Communication with call center via HTTPS*: All confidential communication between the customer and the Call Center is done via HTTPS (which, as you recall, is based on SSL). HTTPS provides high security for the initial information that a customer enters, such as passwords, and problem descriptions
- *Initial download of* ***WebArrow*** *requires explicit confirmation by a customer*: When customers first download ***WebArrow***, they are prompted with a warning dialog. They must click "Yes" to download ***WebArrow***. This ensures that customers are aware of, and approve of, the fact that software is about to be installed on their machine.



? *All data streams are encrypted*: All screen, voice and chat data encrypted using Twofish, a state of the art 128-bit block cipher encryption algorithm. This ensures third parties cannot eavesdrop on the data. Twofish is very low cost in terms of bandwidth and latency.

? *Customer prompting*: Whenever the operator attempts to view or control the customer's desktop, the customer is prompted. Thus it is not possible for the operator to view a customer's sensitive information without their explicit permission.



? *The customer version of* ***WebArrow*** *does not permit inbound connections*: To ensure that ***WebArrow*** could never be used as a Trojan horse, the software has been engineered such that all connections must be initiated *from* the customer. The version of ***WebArrow*** that gets downloaded to a customer does not listen on any ports, and so it can never be externally controlled. Therefore, there is no way for hostile parties to attempt to connect to a customer ***WebArrow***.

In summary, ***WebArrow*** provides a substantial number of security features for customers. Customers need to be shown that any product that they are asked to download onto their computer addresses both the security of their system and the privacy of their data. ***WebArrow*** achieves this by its system architecture, by passwording and state of the art encryption, and by privacy prompts that keep the customer in control.

7.3 SECURITY FEATURES FOR OPERATORS

But what about operators? Operators expose their machines to outsiders by using ***WebArrow.*** We must be able to assure them that this exposure involves minimal risk.

WebArrow features provide high security for operators.

- *Login to call center is required*: Unlike the case with customers, where login to the call center is optional, operators *must* log in to access the Call Center. This avoids inadvertently exposing their computer to the outside world.
- *Communication with call center via HTTPS:* as with the customer, all confidential communication with the call center is done via HTTPS.
- *Operator controls who connects*: Nobody can connect to the operator's machine until the operator *accepts* them. Only the machine with the same IP address as the accepted machine can connect. This ensures that an intruder cannot connect by masquerading as a legitimate user that had just been accepted.
- *All data streams are encrypted*: As mentioned in the discussion of customer security, all screen, voice and chat data are encrypted using Twofish. This ensures third parties cannot eavesdrop on the data.
- *Operator controls service upgrades*: The operator is in control of what service upgrades are offered to the customer (such as viewing the customer's desktop, showing the operator's desktop, or activating VoIP, etc.). No customer and no third party can initiate the view or control of the operator's desktop.
- *Operator can only connect with a* ***WebArrow*** *client*: On connection, the operator ***WebArrow*** issues a challenge to the customer ***WebArrow.*** The customer response is based on the challenge. This ensures customer is indeed running ***WebArrow***, and not a program that is merely spoofing the ***WebArrow*** protocol.

7.4 DEALING WITH SPECIFIC SECURITY THREATS

Now we shall briefly review how ***WebArrow*** deals with specific security threats. These security threats are by far the most common, and most dangerous, that any piece of network software must be concerned with.

Hackers

Hackers represent a wide variety of threats to a networked software system, as was described in Section 1.1 of this paper. ***WebArrow*** has state of the art features to deal with all of these threats.

WebArrow is highly restricted in what connections are possible. A customer cannot accept connections. An operator accepts *only* connections from an accepted, verified customer. So an intrusion via an uninvited connection is extremely unlikely.

In addition, customer screen sharing requires customer confirmation. So a hacker cannot gain control of a customer's screen without their knowledge.

Another form of hacker attack is to impersonate a valid user by cracking the user password file. Using ***WebArrow***, passwords are always transmitted in encrypted form, and are stored in encrypted form on the ***GrizzlyCC*** server. Since passwords are never stored in a plain-text form, a hacker cannot intercept them for a later attack.

Interception

Another common security fear is that confidential information will be intercepted. This is a valid fear, given that this information is flowing over the relatively open and relatively unregulated internet. We ensure confidentiality and privacy by use of strong encryption methods. SSL is used for Web browser communication and Twofish is used for ***WebArrow***-to-***WebArrow*** communication. Both the customer and the operator in a ***WebArrow*** session require a 128 bit encryption key.

Impersonation

To ensure that an apparent ***WebArrow*** client is truly who they claim to be, the ***WebArrow*** operator uses challenge/response authentication to validate that a customer is running a true ***WebArrow*** client. Without this provision, other security features of the system might be compromised. This provision prevents other software from mimicking ***WebArrow***'s protocol.

Another of ***WebArrow***'s provisions that aids in avoiding impersonation attacks is that the operator only accepts connections from the IP address of the customer who entered the Call Center. This helps ensure that the connecting customer has been accepted by the operator.

Finally, when connecting to the Call Center, the customer is presented with a digitally signed SSL certificate. This helps the customer verify that the Call Center is the computer it claims to be (and hence prevents impersonation of the Call Center).

Remotely Log In as Root User

Root user (administrator) login to ***GrizzlyCC*** requires a password, and this password, like all others within ***GrizzlyCC***, is transmitted to the Call Center server via SSL. The weakest link is, therefore, the password itself. This must be securely kept, and it must be chosen to be hard to guess.

Content Threat

To guard against content threats, all ***WebArrow*** software is digitally signed, identifying its origin at Namzak Labs.

8. CONCLUSIONS

This white paper has presented an overview of internet security threats and techniques, and followed that with a detailed look at the security provisions in ***WebArrow*** and ***GrizzlyCC***.

It must be cautioned in parting that no internet-deployed software can guarantee 100% security. The goal is to provide sufficient security mechanisms to:

1) Present a strong case to your customers that their security and privacy is being respected, and
2) Establish a highly secure environment for your own operations

With ***WebArrow*** we have architected numerous features to mitigate risks to security and privacy. Among these detailed in this white paper were encryption (SSL, Twofish), passwording, customer-side prompting, the unique ***WebArrow*** system architecture, and firewall compatibility.

Of course attention to policies and users is still required. But ***WebArrow*** provides a solid infrastructure upon which such policies may be confidently implemented.