



Vector Defense
infectionvectors.com
October 2004

Overview

When a worm infects a system, it must exploit an available vector; a path to the resource must be open for the malicious code to enter. Vectors take multiple forms in a network, but always have a system flaw at their core. Whether this flaw is technical or social and how it can be resolved are all discovered through Vector Defense procedures. This paper examines the infection paths network worms take in an environment absent anti-virus software. This in no way implies that anti-virus software is unneeded or redundant. However, it does point out that this software is the last line of defense against infection (often the only line of defense/auditing relied upon by an organization), and that it should be supplemented with a comprehensive set of filters throughout the network.

Vector Defense refers to how an organization protects its network entry points from malicious code, both known and unknown applications. The primary components of solid Vector Defense are identification and mitigation. Identification is the process of monitoring the malicious code population, cataloging attack methods, and understanding how the code propagates. It involves active monitoring of vulnerability reports as they are released into public circulation. Mitigation is the closing of entry points, either by blocking all access to the resource via a specific path or by scanning electronic transfers for potentially malicious code. Mitigation may be completely non-technical, restricting the access to a resource purely through policy and awareness training falls into this category as well.

Identifying the Paths

Vectors take the form of any flaw in a system that allows malicious code to enter. The term flaw is appropriate here as it is a preventable weakness in the network; some will be allowed to remain open, some will be closed (again through barriers, filters, or training). There are 3 types of flaws that can be used to describe the whole of malicious code:

- Flaws in Program Code
- Flaws in System Configuration
- Flaws in the Social Environment

Flaws in Program Code

Most everyone that works with computers is familiar with the necessity of patching machines with security fixes routinely. Flaws in program code create unintended system

actions/reactions that cannot be fixed without changing the actual code of the application, generally completed by the manufacturer and passed down to clients. Program flaws can certainly be mitigated in ways beyond patching such as turning off affected services, using access control lists (ACL) to restrict access, or filtering attacks from reaching vulnerable machines.

Examples of program code flaws are the Microsoft Windows LSASS or DCOM RPC buffer overflows, the OpenSSL unknown message types infinite loop bug, and the heap overflow associated with Apache 1.3.25's proxy_util.c.

Worms that take advantage of system bugs are Sasser (which exploits the LSASS flaw) and Blaster (which spread by exploiting the RPC DCOM buffer overrun).

Flaws in System Configuration

System Configuration flaws are defined as holes left when each piece is working as expected. They do not involve exploiting a bug in a piece of software, but rather an intended setting in the system. Weakening the network perimeter may be deemed necessary for a particular business function, after a cost/benefit analysis is completed on remediation of the hole.

These types of vectors for infection involve leaving a network file share or database open with no password (or very weak passwords), anonymous FTP servers that allow write access to servers, and unsecured/unauthenticated access to web data stores.

Worms that use configuration flaws are SQLSpida, which attempts to log into MS SQL servers without passwords, and Nebiwo (Deborm), which spreads to administrative shares of Windows boxes that require no password.

Social Flaws

Social flaws exist when users are improperly trained or are unprepared to operate their systems in a secure manner. This involves access to data and to the machines that house the data. Exploits of this nature are often referred to as "social engineering" and can take many forms.

Social vectors include email attachment execution, divulging sensitive data through web forms ("phishing" attacks), and downloading untrusted code from the Internet (via browser or peer-to-peer (P2P) client).

Worms that fall into this category are mass mailers such as Anna Kournikova (VBSWG.J) and ILOVEYOU.

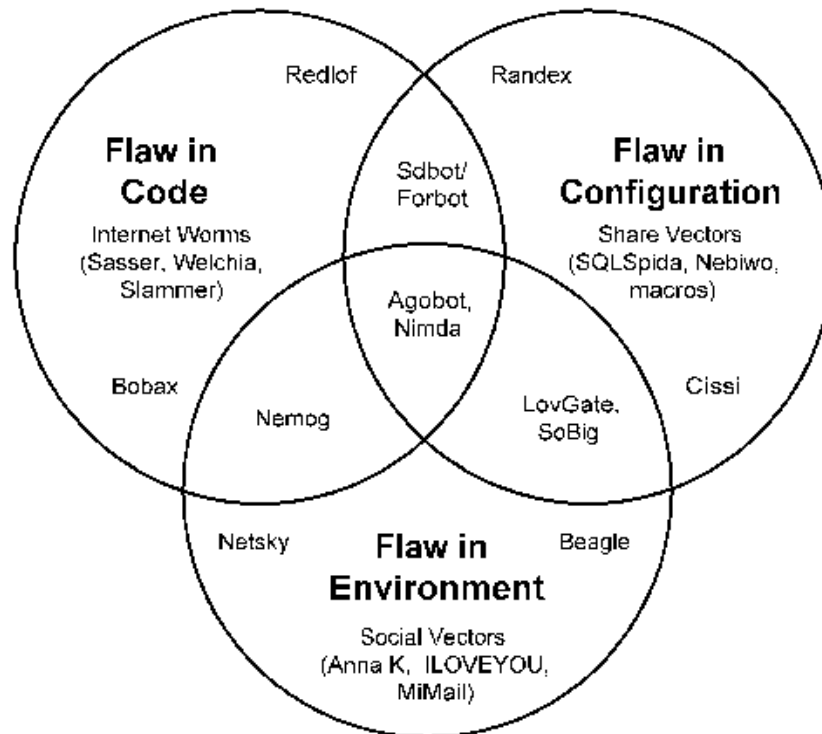
Vector Mix

Of course, many worms don't fall into a single category; they use multiple vectors to attack hosts. These so-called "blended threats" may arrive in an email and then attempt to spread via unprotected file share replication, infect an executable, initiate an Internet worm-like scan, or any number of other routines. The specific vectors fall into two, sometimes all three of the categories described above.

Worms are categorized in this list by how they spread rather than by their payload, or what they do. For example, although the Bugbear worm arrives by email, it will also attempt to inject itself into a list of programs, thereby infecting a user's applications with the virus. For Bugbear to spread to other machines via this mechanism, however, requires a user to share executables, an unwise practice.

The traditional infection mechanism for a virus is to insert itself into an executable. Code that spreads this way takes advantage of hosts that do not have any type of external check on system files and executables, which may be deemed a configuration flaw. The only way that code leaves the machine in these cases, however, is when users share files with other users. Employing untrusted code (anything not from a write-protected source, tested with a file checking application) on official systems is a flaw in the policy and procedures, making it a social problem.

The diagram below identifies a few worms based on the flaws in virus defense they exploit:



The diagram attempts to place worms into a category based on their primary infection vectors. Overlapping vectors contain worms that also propagate by multiple primary means. In addition, the edges of each area are populated with sample worms that may be considered to fall near the “edge” of the category, blurring the line between the various vectors, but still fit within the respective section.

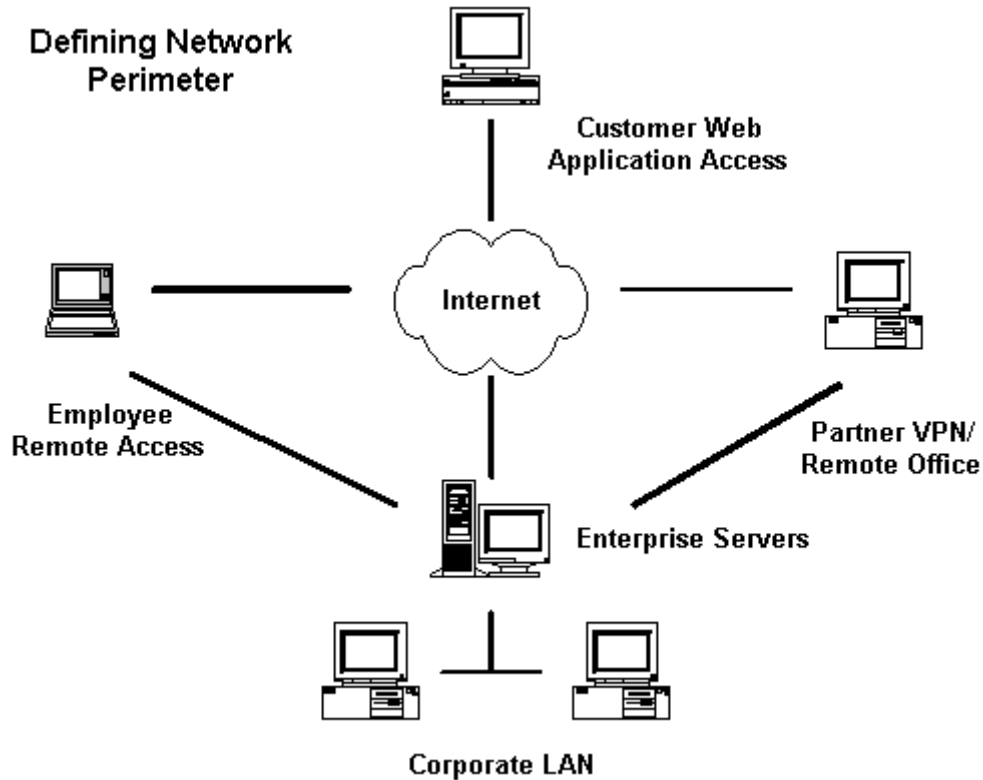
Monitoring the vectors selected by successful worms is important to trend identification. The use of email as the means to penetrate network perimeters (and then launch additional attacks, deliver remote control capability to an author, etc.) during the last two years has prompted many security teams to deploy new mail scanning tools and restrict the attachment types that are allowed into the system.

Making the Connection

Every organization is different; there are exploits that will bounce right off of one network and completely shutdown another. There are two reasons that an enterprise may not feel the effects of a specific worm: the company is not vulnerable to the exploit (there is no path in) or the impact of the worm is not great enough to interrupt operations even if multiple boxes are infected. If a virus infiltrates the network, it is possible that the network traffic it creates is not sufficient to hinder normal operations. Further it is possible that a worm that touches multiple machines (such as one that is able to copy a file to many computers) does not execute any damaging payload (beyond propagation traffic). No infection should ever be taken lightly, however, it is possible to have an infection and not have an emergency worthy of “solutions at any cost.”

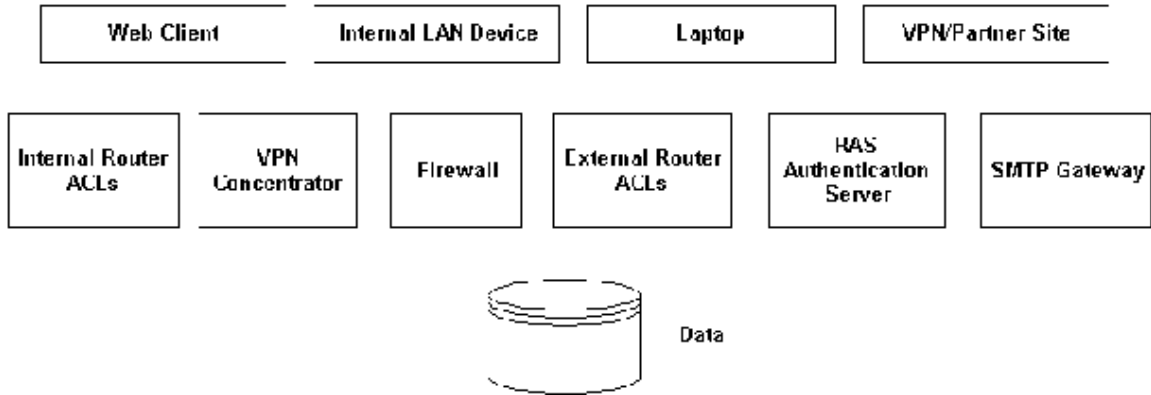
Reactive measures to infections will certainly help reduce the time it takes to answer the first question, “is there a path into the network?” If a machine is found with a virus, a path exists. However, there are many active steps that can help prevent outbreaks. These include defining the network assets, mapping all network entry points, and listing all possible measures to secure those entryways.

Defining network assets in many ways is a simple inventory of components. The traditional network boundary that once existed between the Internet and the corporate LAN has been blurred to include remote access terminals (which are often unmanaged home computers), wireless devices, VPN tunnels, etc. The network perimeter is now difficult to define for many enterprises. The first step is to know exactly how each device retrieves information from the secured machines (organization servers) in the network.



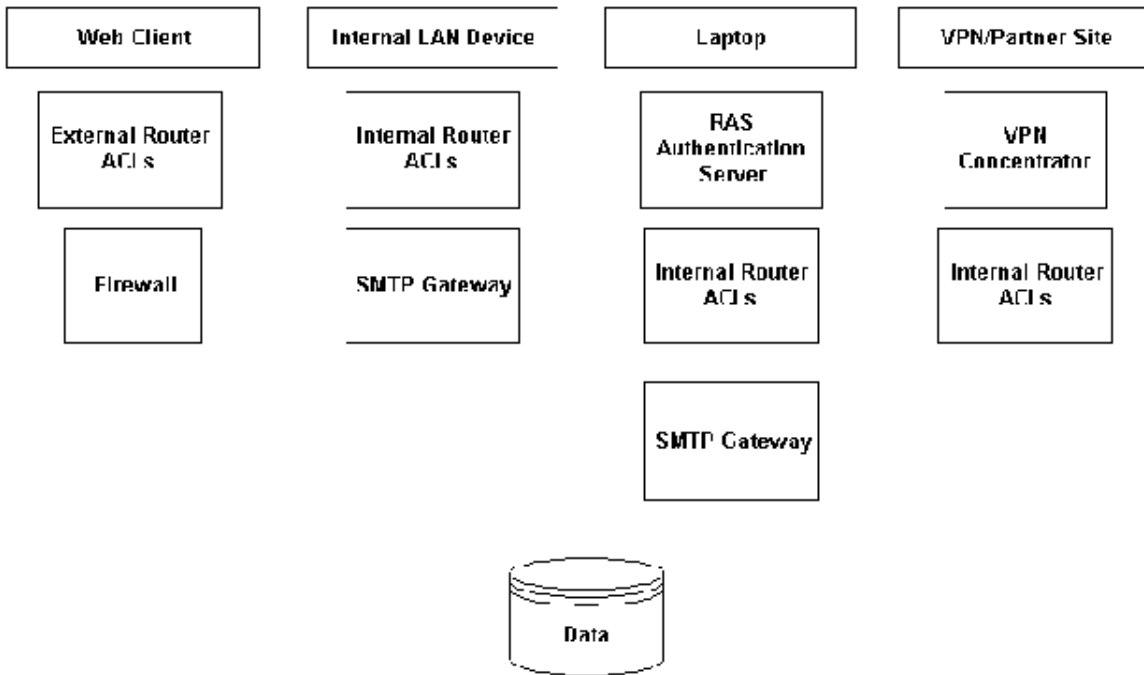
The actual number of enterprise-owned assets is important, as it becomes the baseline for patching and auditing. This group of boxes is also the most trusted of all perimeter devices in virtually all organizations. Each asset utilizes at least one path to access the data within the network.

Mapping access points is the logical extension of the previous exercise. Using the list of assets is the best starting point for determining all open access paths as it will yield a comprehensive hardware inventory and should lend itself to being broken down into classes of access. For example, company desktops probably don't do much traveling, so they can be grouped as "LAN" and by location. Laptops probably straddle many areas, being LAN devices, remote access (dialup), VPN, and possibly external web application (such as web-based database/order entry) all at the same time. The critical information to have gathered is the path each device takes to access data. It is possible that a LAN device takes the same path to an external application that a visiting customer takes. This list should evolve into path enumeration coupled with the filters/defenses in place for each. Visually, it can be represented in the following diagrams. The first shows the device/path and all the filters available within an example network:



Each nominal path can be associated with the logical path it takes to reach the data. This listing needs to incorporate each service/data store within the network. That includes email servers, file shares, and network services (such as DNS, DHCP, etc). The use of generic graphics is often helpful in enumeration and planning for auditing systems. The diagram above can evolve with the data collected in the next phase.

Listing measures to block infection vectors begins by ordering the paths to and from the network data with the existing tools. The diagram below shows one possible outcome, again, shown in graphical form for the sake of simplicity:



In many cases, this will point out areas of weakness, where the network path is currently uncontrollable. The most important result of this part of the exercise is to actually determine what response tools are available to mitigate worm threats. In the case above an administrator is able to see the possible entry points and possible filters should a new threat present itself. So far, this has been primarily a technical inventory, ignoring the strengths of an awareness program and a well-trained user base. That can be viewed as:

firewall to close TCP 445, apply patch KB835732 to the machine, employ AV software that detects and stops the LSASS exploit/worms, or use external firewall appliances/ACLs to prevent anyone from communicating with the machine on TCP 445. Any of these would protect the box; most organizations would choose at least two as part of their network defense plan.

Mitigation efforts should match the existing security policies in most cases (only unique zero-day attacks or where existing policies are underdeveloped are exceptions). This means that groups that have exceptional mechanisms for deploying software should focus on patching, when that is an option. Networks that use strong perimeter controls (firewalls, ACLs, etc.) effectively should choose those as the first tool to block new threats. Mitigating the threat and providing IA teams “breathing room” on an issue will likely produce much better solutions (in terms of cost efficiency and effectiveness) than following the wide range of plans offered by pundits.

The strengths and weaknesses of any organization should be discovered through network assessments that identify the critical assets and where security tools exist. That knowledge will shape the prioritization of what to harden and what to use to harden it. Planning a sound virus defense can be initiated with the guides found on infectionvectors.com.

A Good Offense

There are additional efforts that an enterprise can take to slow the spread of network worms beyond hardening individual machines or the network as a whole. These steps prevent an Internet worm from finding and infecting organization assets.

One method of combating worms is to slow down their scanning abilities. Many security professionals are familiar with tools like LaBrea (<http://labrea.sourceforge.net/labrea-info.html>) that do just that. LaBrea answers for any unused IP addresses with a virtual machine that slows an automated scan by sending only an initial SYN ACK to SYNs and carefully answering additional requests (while setting the RECV window to zero).

Another mechanism that can be employed to reduce the number of machines attacked during an outbreak is to space populated networks widely across a private address range. Using disparate sections of the 10.x.y.z/8 space would make worms like Welchia scan quite a few dead areas (or areas protected with LaBrea) while searching for live hosts.

Crafting the Strategy

Virus coders select the vector that is most likely to succeed on the highest number of machines. When selecting the order of mitigation efforts, the choice should be made in a similar fashion. In addition, it should match the type of flaw that exists within the system: if it is a social problem such as opening an email attachment, a social solution such as awareness training is probably the best place to begin. Patching systems is often the best

mechanism to combat a flaw in code; similarly, configuration flaws are best remedied via changes to the network design.

The process of identifying how malicious code would work its way into the enterprise can appear to be a time consuming process for large networks. However, the number of unique paths into a system is often very comparable across networks of all sizes. It may, however, require much more configuration work to harden a large enterprise versus a small home office.

The overarching task for information defenders is to identify what needs to be protected, how malicious code could penetrate those systems, and what tools are available to defend the flaws that the code would exploit. Rationally selecting the mitigation efforts that best fit the problem and protect the highest number of devices is made much simpler through this process. infectionvectors.com provides a number of resources that can help this exercise and others for enterprises of all sizes.

References

MS03-039 (DCOM RPC Heap Overflow)

<http://www.microsoft.com/technet/security/bulletin/MS03-039.asp>

MS04-011 (includes LSASS)

<http://www.microsoft.com/technet/security/bulletin/MS04-011.msp>

OpenSSL Bug Ref

<http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2004-0081>

Apache flaw

<http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2004-0492>

Sasser Worm Information

<http://securityresponse.symantec.com/avcenter/venc/data/w32.sasser.worm.html>

<http://www.sophos.com/virusinfo/analyses/w32sassera.html>

Stop Port Scans with LaBrea (SANS Paper)

<http://www.sans.org/rr/papers/61/405.pdf>

Virus Defense Process Model

<http://www.infectionvectors.com/emergencyprep.htm>

Copyright © 2004 infectionvectors.com. All rights reserved.