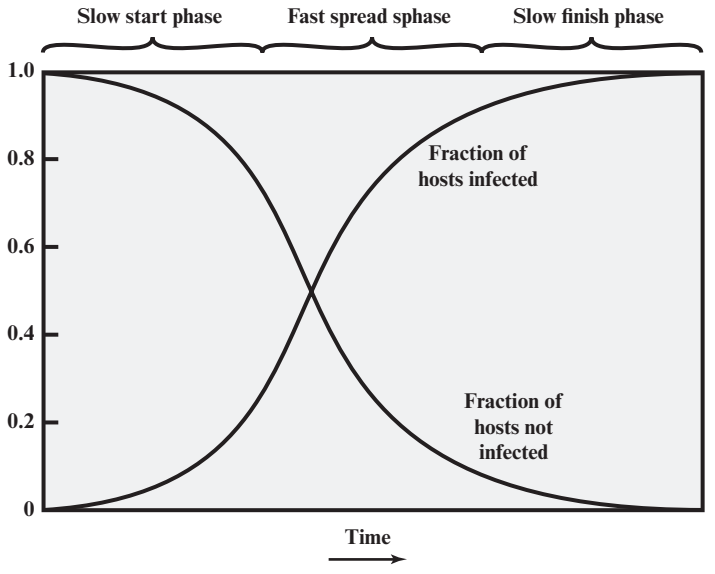

Figure 10.3 Worm Propagation Model

hosts infects two more hosts, and so on. This results in exponential growth. After a time, infecting hosts waste some time attacking already-infected hosts, which reduces the rate of infection. During this middle phase, growth is approximately linear, but the rate of infection is rapid. When most vulnerable computers have been infected, the attack enters a slow finish phase as the worm seeks out those remaining hosts that are difficult to identify.

Clearly, the objective in countering a worm is to catch the worm in its slow start phase, at a time when few hosts have been infected.

Zou and others [ZOU05] describe a model for worm propagation based on an analysis of network worm attacks at that time. The speed of propagation and the total number of hosts infected depend on a number of factors, including the mode of propagation, the vulnerability or vulnerabilities exploited, and the degree of similarity to preceding attacks. For the latter factor, an attack that is a variation of a recent previous attack may be countered more effectively than a more novel attack. Zou's model agrees closely with Figure 10.3.

The Morris Worm

The earliest significant worm infection was released onto the Internet by Robert Morris in 1988 [ORMA03]. The Morris worm was designed to spread on UNIX systems and used a number of different techniques for propagation. When a copy began execution, its first task was to discover other hosts known to this host that would allow entry from this host. The worm performed this task by examining a variety of lists and tables, including system tables that declared which other machines were trusted by this host, users' mail forwarding files, tables by which users gave themselves permission for access to remote accounts, and from a program that

reported the status of network connections. For each discovered host, the worm tried a number of methods for gaining access:

1. It attempted to log on to a remote host as a legitimate user. In this method, the worm first attempted to crack the local password file and then used the discovered passwords and corresponding user IDs. The assumption was that many users would use the same password on different systems. To obtain the passwords, the worm ran a password-cracking program that tried
   a. Each user's account name and simple permutations of it
   b. A list of 432 built-in passwords that Morris thought to be likely candidates[2]
   c. All the words in the local system dictionary
4. It exploited a bug in the UNIX finger protocol, which reports the whereabouts of a remote user.
5. It exploited a trapdoor in the debug option of the remote process that receives and sends mail.

If any of these attacks succeeded, the worm achieved communication with the operating system command interpreter. It then sent this interpreter a short bootstrap program, issued a command to execute that program, and then logged off. The bootstrap program then called back the parent program and downloaded the remainder of the worm. The new worm was then executed.

State of Worm Technology

The state of the art in worm technology includes the following:

- **Multiplatform:** Newer worms are not limited to Windows machines but can attack a variety of platforms, especially the popular varieties of UNIX, or exploit macro or scripting languages supported in popular document types.
- **Multiexploit:** New worms penetrate systems in a variety of ways, using exploits against Web servers, browsers, e-mail, file sharing, and other network-based applications, or via shared media.
- **Ultrafast spreading:** Exploit various techniques to optimize the rate of spread of a worm to maximize its likelihood of locating as many vulnerable machines as possible in a short time period.
- **Polymorphic:** To evade detection, skip past filters, and foil real-time analysis, worms adopt the virus polymorphic technique. Each copy of the worm has new code generated on the fly using functionally equivalent instructions and encryption techniques.
- **Metamorphic:** In addition to changing their appearance, metamorphic worms have a repertoire of behavior patterns that are unleashed at different stages of propagation.
- **Transport vehicles:** Because worms can rapidly compromise a large number of systems, they are ideal for spreading a wide variety of malicious payloads,

[2]The complete list is provided at this book's Premium Content Web site.

such as distributed denial-of-service bots, rootkits, spam e-mail generators, and spyware.

- **Zero-day exploit:** To achieve maximum surprise and distribution, a worm should exploit an unknown vulnerability that is only discovered by the general network community when the worm is launched.

Mobile Code

SP 800-28 (*Guidelines on Active Content and Mobile Code*, March 2008) defines mobile code as programs (e.g., script, macro, or other portable instruction) that can be shipped unchanged to a heterogeneous collection of platforms and execute with identical semantics.

Mobile code is transmitted from a remote system to a local system and then executed on the local system without the user's explicit instruction. Mobile code often acts as a mechanism for a virus, worm, or Trojan horse to be transmitted to the user's workstation. In other cases, mobile code takes advantage of vulnerabilities to perform its own exploits, such as unauthorized data access or root compromise. Popular vehicles for mobile code include Java applets, ActiveX, JavaScript, and VBScript. The most common ways of using mobile code for malicious operations on local system are cross-site scripting, interactive and dynamic Web sites, e-mail attachments, and downloads from untrusted sites or of untrusted software.

Client-Side Vulnerabilities and Drive-by-Downloads

Another approach to exploiting software vulnerabilities involves the exploit of bugs in user applications to install malware. One common approach to this exploits browser vulnerabilities so that when the user views a Web page controlled by the attacker, it contains code that exploits the browser bug to download and install malware on the system without the user's knowledge or consent. This is known as a **drive-by-download** and is a common exploit in recent attack kits. In most cases this malware does not actively propagate as a worm does, but rather waits for unsuspecting users to visit the malicious Web page in order to spread to their systems.

In general, drive-by-download attacks are aimed at anyone who visits a compromised site and is vulnerable to the exploits used. Watering-hole attacks are a variant of this used in highly targeted attacks. The attacker researches their intended victims to identify Web sites they are likely to visit and then scans these sites to identify those with vulnerabilities that allow their compromise with a drive-by-download attack. They then wait for one of their intended victims to visit one of the compromised sites. Their attack code may even be written so that it will only infect systems belonging to the target organization and take no action for other visitors to the site. This greatly increases the likelihood of the site compromise remaining undetected.

Malvertising is another technique used to place malware on Web sites without actually compromising them. The attacker pays for advertisements that are highly likely to be placed on their intended target Web sites, and which incorporate malware in them. Using these malicious adds, attackers can infect visitors to sites displaying them. Again, the malware code may be dynamically generated to either reduce the chance of detection or only infect specific systems.

Related variants can exploit bugs in common e-mail clients, such as the Klez mass-mailing worm seen in October 2001, which targeted a bug in the HTML handling in Microsoft's Outlook and Outlook Express programs to automatically run itself. Or, such malware may target common PDF viewers to also download and install malware without the user's consent, when they view a malicious PDF document [STEV11]. Such documents may be spread by spam e-mail or be part of a targeted phishing attack, as we discuss next.

Clickjacking

Clickjacking, also known as a *user-interface (UI) redress attack*, is a vulnerability used by an attacker to collect an infected user's clicks. The attacker can force the user to do a variety of things from adjusting the user's computer settings to unwittingly sending the user to Web sites that might have malicious code. Also, by taking advantage of Adobe Flash or JavaScript, an attacker could even place a button under or over a legitimate button, making it difficult for users to detect. A typical attack uses multiple transparent or opaque layers to trick a user into clicking on a button or link on another page when they were intending to click on the top level page. Thus, the attacker is hijacking clicks meant for one page and routing them to another page, most likely owned by another application, domain, or both.

Using a similar technique, keystrokes can also be hijacked. With a carefully crafted combination of stylesheets, iframes, and text boxes, a user can be led to believe they are typing in the password to their e-mail or bank account but are instead typing into an invisible frame controlled by the attacker.

There is a wide variety of techniques for accomplishing a clickjacking attack, and new techniques are developed as defenses to older techniques are put in place. [NIEM11] and [STON10] are useful discussions.

10.5 PROPAGATION—SOCIAL ENGINEERING—SPAM E-MAIL, TROJANS

The final category of malware propagation we consider involves social engineering, "tricking" users to assist in the compromise of their own systems or personal information. This can occur when a user views and responds to some SPAM e-mail or permits the installation and execution of some Trojan horse program or scripting code.

Spam (Unsolicited Bulk) E-Mail

Unsolicited bulk e-mail, commonly known as spam, imposes significant costs on both the network infrastructure needed to relay this traffic and on users who need to filter their legitimate e-mails out of this flood. In response to the explosive growth in spam, there has been the equally rapid growth of the antispam industry, which provides products to detect and filter spam e-mails. This has led to an arms race between the spammers devising techniques to sneak their content through and the defenders taking efforts to block them. In recent years, the volume of spam e-mail has started to decline. One reason is the rapid growth of attacks, including spam,

spread via social media networks. This reflects the rapid growth in use of these networks, which form a new arena for attackers to exploit [SYMA13].

While some spam is sent from legitimate mail servers, most recent spam is sent by botnets using compromised user systems, as we discuss in Section 10.6. A significant portion of spam e-mail content is just advertising, trying to convince the recipient to purchase some product online, or used in scams, such as stock scams or money mule job ads. But spam is also a significant carrier of malware. The e-mail may have an attached document, which, if opened, may exploit a software vulnerability to install malware on the user's system, as we discussed in the previous section. Or, it may have an attached Trojan horse program or scripting code that, if run, also installs malware on the user's system. Some trojans avoid the need for user agreement by exploiting a software vulnerability in order to install themselves, as we discuss next. Finally the spam may be used in a phishing attack, typically directing the user either to a fake Web site that mirrors some legitimate service, such as an online banking site, where it attempts to capture the user's login and password details, or to complete some form with sufficient personal details to allow the attacker to impersonate the user in an identity theft. All of these uses make spam e-mails a significant security concern. However, in many cases it requires the user's active choice to view the e-mail and any attached document or to permit the installation of some program, in order for the compromise to occur.

Trojan Horses

A Trojan horse is a useful, or apparently useful, program or utility containing hidden code that, when invoked, performs some unwanted or harmful function.

Trojan horse programs can be used to accomplish functions indirectly that the attacker could not accomplish directly. For example, to gain access to sensitive, personal information stored in the files of a user, an attacker could create a Trojan horse program that, when executed, scans the user's files for the desired sensitive information and sends a copy of it to the attacker via a Web form or e-mail or text message. The author could then entice users to run the program by incorporating it into a game or useful utility program and making it available via a known software distribution site or app store. This approach has been used recently with utilities that "claim" to be the latest antivirus scanner, or security update, for systems, but which are actually malicious trojans, often carrying payloads such as spyware that searches for banking credentials. Hence, users need to take precautions to validate the source of any software they install.

Trojan horses fit into one of three models:

- Continuing to perform the function of the original program and additionally performing a separate malicious activity
- Continuing to perform the function of the original program but modifying the function to perform malicious activity (e.g., a Trojan horse version of a login program that collects passwords) or to disguise other malicious activity (e.g., a Trojan horse version of a process-listing program that does not display certain processes that are malicious)
- Performing a malicious function that completely replaces the function of the original program

Some trojans avoid the requirement for user assistance by exploiting some software vulnerability to enable their automatic installation and execution. In this they share some features of a worm, but unlike it, they do not replicate. A prominent example of such an attack was the Hydraq Trojan used in Operation Aurora in 2009 and early 2010. This exploited a vulnerability in Internet Explorer to install itself and targeted several high-profile companies [SYMA13]. It was typically distributed either by spam e-mail or via a compromised Web site using a "drive-by-download."

10.6 PAYLOAD—SYSTEM CORRUPTION

Once malware is active on the target system, the next concern is what actions it will take on this system, that is, what payload does it carry. Some malware has a nonexistent or nonfunctional payload. Its only purpose, either deliberate or due to accidental early release, is to spread. More commonly, it carries one or more payloads that perform covert actions for the attacker.

An early payload seen in a number of viruses and worms resulted in data destruction on the infected system when certain trigger conditions were met [WEAV03]. A related payload is one that displays unwanted messages or content on the user's system when triggered. More seriously, another variant attempts to inflict real-world damage on the system. All of these actions target the integrity of the computer system's software or hardware, or of the user's data. These changes may not occur immediately, but only when specific trigger conditions are met that satisfy their logic-bomb code.

As an alternative to just destroying data, some malware encrypts the user's data and demands payment in order to access the key needed to recover this information. This is sometimes known as **ransomware**. The PC Cyborg Trojan seen in 1989 was an early example of this. However, around mid-2006 a number of worms and trojans, such as the Gpcode Trojan, that used public-key cryptography with increasingly larger key sizes to encrypt data. The user needed to pay a ransom or to make a purchase from certain sites, in order to receive the key to decrypt this data. While earlier instances used weaker cryptography that could be cracked without paying the ransom, the later versions using public-key cryptography with large key sizes could not be broken this way.

Real-World Damage

A further variant of system corruption payloads aims to cause damage to physical equipment. The infected system is clearly the device most easily targeted. The Chernobyl virus not only corrupts data, it attempts to rewrite the BIOS code used to initially boot the computer. If it is successful, the boot process fails, and the system is unusable until the BIOS chip is either reprogrammed or replaced.

The Stuxnet worm targets some specific industrial control system software as its key payload [CHEN11]. If control systems using certain Siemens industrial control software with a specific configuration of devices are infected, then the worm replaces the original control code with code that deliberately drives the controlled equipment outside its normal operating range, resulting in the failure of the attached equipment. The centrifuges used in the Iranian uranium enrichment program were strongly suspected as the target, with reports of much higher than normal failure rates observed in them over the period when this worm was active. As noted in our

earlier discussion, this has raised concerns over the use of sophisticated targeted malware for industrial sabotage.

Logic Bomb

A key component of data-corrupting malware is the logic bomb. The logic bomb is code embedded in the malware that is set to "explode" when certain conditions are met. Examples of conditions that can be used as triggers for a logic bomb are the presence or absence of certain files or devices on the system, a particular day of the week or date, a particular version or configuration of some software, or a particular user running the application. Once triggered, a bomb may alter or delete data or entire files, cause a machine halt, or do some other damage. All of the examples we describe in this section include such code.

10.7 PAYLOAD—ATTACK AGENT—ZOMBIE, BOTS

The next category of payload we discuss is where the malware subverts the computational and network resources of the infected system for use by the attacker. Such a system is known as a bot (robot), zombie, or drone, and secretly takes over another Internet-attached computer and then uses that computer to launch or manage attacks that are difficult to trace to the bot's creator. The bot is typically planted on hundreds or thousands of computers belonging to unsuspecting third parties. The collection of bots often is capable of acting in a coordinated manner; such a collection is referred to as a **botnet**. This type of payload attacks the integrity and availability of the infected system.

Uses of Bots

[HONE05] lists the following uses of bots:

- **Distributed denial-of-service (DDoS) attacks:** A DDoS attack is an attack on a computer system or network that causes a loss of service to users. We examine DDoS attacks in Section 10.10.
- **Spamming:** With the help of a botnet and thousands of bots, an attacker is able to send massive amounts of bulk e-mail (spam).
- **Sniffing traffic:** Bots can also use a packet sniffer to watch for interesting clear-text data passing by a compromised machine. The sniffers are mostly used to retrieve sensitive information like usernames and passwords.
- **Keylogging:** If the compromised machine uses encrypted communication channels (e.g., HTTPS or POP3S), then just sniffing the network packets on the victim's computer is useless because the appropriate key to decrypt the packets is missing. But by using a keylogger, which captures keystrokes on the infected machine, an attacker can retrieve sensitive information.
- **Spreading new malware:** Botnets are used to spread new bots. This is very easy since all bots implement mechanisms to download and execute a file via HTTP or FTP. A botnet with 10,000 hosts that acts as the start base for a worm or mail virus allows very fast spreading and thus causes more harm.

- **Installing advertisement add-ons and browser helper objects (BHOs):** Botnets can also be used to gain financial advantages. This works by setting up a fake Web site with some advertisements: The operator of this Web site negotiates a deal with some hosting companies that pay for clicks on ads. With the help of a botnet, these clicks can be "automated" so that instantly a few thousand bots click on the pop-ups. This process can be further enhanced if the bot hijacks the start-page of a compromised machine so that the "clicks" are executed each time the victim uses the browser.
- **Attacking IRC chat networks:** Botnets are also used for attacks against Internet Relay Chat (IRC) networks. Popular among attackers is the so-called clone attack: In this kind of attack, the controller orders each bot to connect a large number of clones to the victim IRC network. The victim is flooded by service requests from thousands of bots or thousands of channel-joins by these cloned bots. In this way, the victim IRC network is brought down, similar to a DDoS attack.
- **Manipulating online polls/games:** Online polls/games are getting more and more attention and it is rather easy to manipulate them with botnets. Since every bot has a distinct IP address, every vote will have the same credibility as a vote cast by a real person. Online games can be manipulated in a similar way.

Remote Control Facility

The remote control facility is what distinguishes a bot from a worm. A worm propagates itself and activates itself, whereas a bot is controlled from some central facility, at least initially.

A typical means of implementing the remote control facility is on an IRC server. All bots join a specific channel on this server and treat incoming messages as commands. More recent botnets tend to avoid IRC mechanisms and use covert communication channels via protocols such as HTTP. Distributed control mechanisms, using peer-to-peer protocols, are also used, to avoid a single point of failure.

Once a communications path is established between a control module and the bots, the control module can activate the bots. In its simplest form, the control module simply issues command to the bot that causes the bot to execute routines that are already implemented in the bot. For greater flexibility, the control module can issue update commands that instruct the bots to download a file from some Internet location and execute it. The bot in this latter case becomes a more general-purpose tool that can be used for multiple attacks.

10.8 PAYLOAD—INFORMATION THEFT—KEYLOGGERS, PHISHING, SPYWARE

We now consider payloads where the malware gathers data stored on the infected system for use by the attacker. A common target is the user's login and password credentials to banking, gaming, and related sites, which the attacker then uses to impersonate the user to access these sites for gain. Less commonly, the payload may target documents or system configuration details for the purpose of reconnaissance or espionage. These attacks target the confidentiality of this information.

Credential Theft, Keyloggers, and Spyware

Typically, users send their login and password credentials to banking, gaming, and related sites over encrypted communication channels (e.g., HTTPS or POP3S), which protect them from capture by monitoring network packets. To bypass this, an attacker can install a **keylogger**, which captures keystrokes on the infected machine to allow an attacker to monitor this sensitive information. Since this would result in the attacker receiving a copy of all text entered on the compromised machine, keyloggers typically implement some form of filtering mechanism that only returns information close to desired keywords (e.g., "login" or "password" or "paypal.com").

In response to the use of keyloggers, some banking and other sites switched to using a graphical applet to enter critical information, such as passwords. Since these do not use text entered via the keyboard, traditional keyloggers do not capture this information. In response, attackers developed more general **spyware** payloads, which subvert the compromised machine to allow monitoring of a wide range of activity on the system. This may include monitoring the history and content of browsing activity, redirecting certain Web page requests to fake sites controlled by the attacker, dynamically modifying data exchanged between the browser and certain Web sites of interest. All of which can result in significant compromise of the user's personal information.

Phishing and Identity Theft

Another approach used to capture a user's login and password credentials is to include a URL in a spam e-mail that links to a fake Web site controlled by the attacker, but which mimics the login page of some banking, gaming, or similar site. This is normally included in some message suggesting that urgent action is required by the user to authenticate his or her account, to prevent it being locked. If the user is careless, and doesn't realize that he or she is being conned, then following the link and supplying the requested details will certainly result in the attackers exploiting the user's account using the captured credentials.

More generally, such a spam e-mail may direct a user to a fake Web site controlled by the attacker or to complete some enclosed form and return to an e-mail accessible to the attacker, which is used to gather a range of private, personal information on the user. Given sufficient details, the attacker can then "assume" the user's identity for the purpose of obtaining credit or sensitive access to other resources. This is known as a **phishing** attack, which exploits social engineering to leverage user's trust by masquerading as communications from a trusted source [GOLD10].

Such general spam e-mails are typically widely distributed to very large numbers of users, often via a botnet. While the content will not match appropriate trusted sources for a significant fraction of the recipients, the attackers rely on it reaching sufficient users of the named trusted source, a gullible portion of whom will respond, for it to be profitable.

A more dangerous variant of this is the **spear-phishing** attack. This again is an e-mail claiming to be from a trusted source. However, the recipients are carefully researched by the attacker, and each e-mail is carefully crafted to suit its recipient specifically, often quoting a range of information to convince him or her of its

authenticity. This greatly increases the likelihood of the recipient responding as desired by the attacker.

Reconnaissance and Espionage

Credential theft and identity theft are special cases of a more general reconnaissance payload, which aims to obtain certain types of desired information and return this to the attacker. These special cases are certainly the most common; however other targets are known. Operation Aurora in 2009 used a Trojan to gain access to and potentially modify source code repositories at a range of high-tech, security, and defense contractor companies [SYMA13]. The Stuxnet worm discovered in 2010 included capture of hardware and software configuration details in order to determine whether it had compromised the specific desired target systems. Early versions of this worm returned this same information, which was then used to develop the attacks deployed in later versions [CHEN11].

10.9 PAYLOAD—STEALTHING—BACKDOORS, ROOTKITS

The final category of payload we discuss concerns techniques used by malware to hide its presence on the infected system and to provide covert access to that system. This type of payload also attacks the integrity of the infected system.

Backdoor

A **backdoor**, also known as a **trapdoor**, is a secret entry point into a program that allows someone who is aware of the backdoor to gain access without going through the usual security access procedures. The backdoor is code that recognizes some special sequence of input or is triggered by being run from a certain user ID or by an unlikely sequence of events.

A backdoor is usually implemented as a network service listening on some nonstandard port that the attacker can connect to and issue commands through to be run on the compromised system.

It is difficult to implement operating system controls for backdoors in applications. Security measures must focus on the program development and software update activities, and on programs that wish to offer a network service.

Rootkit

A rootkit is a set of programs installed on a system to maintain covert access to that system with administrator (or root)[3] privileges, while hiding evidence of its presence to the greatest extent possible. This provides access to all the functions and services of the operating system. The rootkit alters the host's standard functionality in a malicious and stealthy way. With root access, an attacker has complete control of the system and can add or change programs and files, monitor processes, send and receive network traffic, and get backdoor access on demand.

[3]On UNIX systems, the administrator, or *superuser*, account is called root; hence the term *root access*.