The operating system cannot know that you will write

* 15,000 bytes into a dynamic heap space for which you requested only 15 bytes
* eight bytes into a four-byte parameter
* four return parameter values into three parameter spaces

Stack Smashing

* Overwrite the program counter in the stack
* Overwrite part of the code in low memory
* Overwrite the program counter and data in the stack (causing the data overwritten into the stack to be executed)

Vulnerability: Parameter Length and Number

* Too many parameters
* Wrong output type or size

Example: caller may expect a date result as a number of days after January 1, 1970 but the result produced is a string of the form “dd-mmm-yyyy”

* Too-long string

Vulnerability:   
Unsafe Utility Programs

Compare

strcpy(dest, src)

to

strncpy(dest, src, n)

important Overflow Exploitation Attacks

Morris Worm (1988)

Effects:

Resource exhaustion

(host already infected?)

* Disconnection of many systems (6,000) from the ARPANET
* Isolation and inability to perform necessary work

Locating Potential Machines

* find user accounts to invade on the target machine
* exploit a bug in the finger program
* use a trapdoor in the sendmail mail handler

Remain Undiscoverable

* If a transmission error occurred while the rest of the worm was being fetched, the loader zeroed and then deleted all code already transferred and it exited.
* As soon as the worm received its full code, it brought the code into memory, encrypted it, and deleted the original copies from disk.
* The worm periodically changed its name and process identifier.

Code Red Worm (2001)

* On July 19, it infected more than 250,000 systems in just 9 hours
* Overall, 750,000 servers were affected, including 400,000 just in the period from August 1 to 10
* 1/8 servers running code subject to infection were indeed infected

Exploiting the Vulnerabilities

* Web servers running Microsoft’s Internet Information Server (IIS)
* To infect a server, the worm overflows the buffer in the dynamic link library idq.dll
* To propagate, Code Red checks IP addresses on port 80 of the PC to see if that web server is vulnerable

Overflow Details

* Resulted from a change in the representation of a string from ASCII to Unicode notation in the routine DecodeURLEscapes()
* The calling procedure IDQ.DDL is supposed to pass the number of bytes of a Unicode string but instead passes the length in ASCII units

Behavior

* From day 1 to 19 of the month, the worm spawned 99 threads that scanned for other vulnerable computers
* On days 20 to 27, the worm launched a denial-of-service attack at [www.whitehouse.gov](http://www.whitehouse.gov/)
* From day 28 to the end of the month, the worm did nothing

Code Red Variants

* The second variant (end of July 2001) did not deface the web site, but its propagation was randomized and optimized to infect servers more quickly
* The third version (early August) injected a Trojan horse in the target to ensure that a remote attacker could execute any command on the server

More Exotic Features

* The worm also checked the year and month, so that it would automatically stop propagating in October 2002.
* The worm rebooted the server after 24 or 48 hours, wiping itself from memory but leaving the Trojan horse in place

The Trojan Horse

* Modified the system registry to disable certain kinds of file protection and ensure that some directories have read, write, and execute permission
* Continued to run in the background, resetting the registry every 10 minutes

Propagation

* Created 300 or 600 threads (depending on the variant) and tried for 24 or 48 hours to spread.
* Chose a random IP address close to the host computer’s own address (later versions).
* SQL Slammer
* Conficker

Cloud Computing

* Security Challenges
* (Textbook Interlude A)

Cloud Characteristics

* On-demand self-service

Server time or network storage

* Broad network access mobile phones, laptops, desktops, and mainframe computers
* Resource poo ling Multitenant model: a single resource accessed by multiple customers
* Rapid elasticityServices can be quickly and automatically scaled up or down
* Measured service Services and resources can be monitored, controlled, and reported

Service Models

* Software as a service (SaaS)
  + Applications
* Platform as a service (PaaS)
  + Languages and tools
* Infrastructure as a service (IaaS)
  + Processing
  + Storage
  + Networks
  + Operating Systems
  + Some applications
  + Some network components

Deployment Models

* **Private** cloud: infrastructure that is operated exclusively by and for the organization that owns it, but cloud management may be contracted out to a third party.
* **Community** cloud: shared by several organizations, usually intended to accomplish a shared goal.
* **Public** cloud: available to the general public, owned by an organization that sells cloud services.
* **Hybrid** cloud: composed of two or more types of clouds, connected by technology that enables data and applications to be moved around the infrastructure to balance loads among clouds.

Some vendors suggest that

* security is cheaper or better in the cloud and
* customers no longer need to worry about security

Technical Risks

* Resource exhaustion
* How can you isolate data or operations when the resources are shared?
* How can the cloud protect data in transit? Would encryption slow it down?
* Denial-of-service attacks
* The system controlling the cloud itself: Like a giant operating system, it must control infrastructure provision and service provision, map scattered data together and identify their owners, and coordinate security and privacy policies and services.

More Security Concerns

* Customer instructions for deleting data may not be followed may just have been moved to an “undisclosed location
* A malicious insider can attack data integrity changing actual values but manipulating the interface so that customers think the data have not been changed
* Forensic analysis extremely difficult, even with an audit trail due to the dynamic reshuffling
* Many legal jurisdictions something that is illegal in one part of the cloud may in fact be legal in another

Legal Risks

* How do we manage identities in the cloud? How can we authenticate an access, manage huge access control lists, or identify a user?
  + botnets have been discovered operating from inside an IaaS cloud
  + spammers have rented cloud space to launch phishing campaigns
  + clouds can be put to work to break encryption quickly
* If licenses run out, support for applications could disappear.

**Countermeasure: Programmer Bounds Checking**

Programmer, operating system, compiler, and hardware should…

(Reminder)

* Check lengths before writing.
* Confirm that array subscripts are within limits.
* Double-check boundary condition code to catch possible off-by-one errors.
* Monitor input and accept only as many characters as can be handled.
* Use string utilities that transfer only a bounded amount of data.
* Be suspicious of procedures that might overrun their space.

**Countermeasure: Programming Language Support**

* The more programming errors the language prevents, the safer it is.
* Two ways:
  + checking boundaries on memory transfers
  + checking types of data values

*(Look up Sir C. A. R. Hoare, if interested)*

Buffer Overflow “Features”

* Write directly to particular memory addresses
* Inappropriate operations on certain data types

Buffer Overflow “Features”

* Write directly to particular memory addresses

**Language Countermeasure:**

**Memory Safety**

* Inappropriate operations on certain data types

**Language Countermeasure:**

**Type Safety**

Pointers Checking

* using a pointer before assigning it a value
* incrementing a pointer by the wrong amount
* performing incorrect arithmetic on pointers
* using the wrong pointer, such as a pointer to a parameter’s address instead of to the underlying parameter data item

**Countermeasure: Stack Protection/Tamper Detection**

The attack–countermeasure tennis match

* The canary value should be recognizable, for example, 0x0f1e2d3c.
* To add variety the defender picks random patterns that follow some sequence, such as 0x0f1e2d3c, 0x0f1e2d3d
* …
* Cryptography can be used to generate verifiable but effectively unpredictable numbers.

**Countermeasure: Hardware Protection of Executable Space**

What role can the operating system play?

* Separation
  + Physical
  + Temporal
  + Logical
  + Cryptographic

Sharing at various granularities

Advantages for the Operating System

* The OS can place any segment at any location or move any segment to any location, even after the program begins to execute.
* A segment can be removed from main memory, if not used currently.
* Every address reference passes through the OS.

Segmentation Benefits

* Each address reference is checked.
* Many different classes of data items can be assigned different levels of protection.
* Two or more users can share access to a segment, with potentially different access rights.
* A user cannot generate an address or access to an unpermitted segment.

Segmentation Problems

* A program may generate a reference to a valid segment name, but with an offset beyond the end of the segment.
* Efficient implementation presents two problems:
  + Segment names are inconvenient to encode in instructions, and the OS’s lookup of the name in a table can be slow.
  + If names are converted to numbers, complication when two procedures share the same segment.

**Countermeasure: General Access Control**

* Check every access
* Enforce least privilege
* Verify acceptable usage

Access control is a function that must be performed by the operating system.

(Exceptions include database managers)

Problems

* Each list becomes too large if there are many shared objects.
* Revocation of access, especially combined with propagation of access rights.
* Pseudonyms: can lead to multiple permissions that are not necessarily consistent (see next slide)