Security of Computer Systems

[1] Krishna kumar,
[1]Computer Science &Engineering,
Gitam, Kablana, Jhajjar,
Haryana, India

[2] Ms. SoniaChaudhary
[2]Assistant Professor
Department of Computer Science & Engineering,
Gitam, Kablana, Haryana, India

After thirty years of work on computer security, why are almost all the systems in service today extremely vulnerable to attack? The main reason is that security is expensive to set up and a nuisance to run, so people judge from experience how little of it they can get away with. Since there's been little damage, people decide that they don't need much security. In addition, setting it up is so complicated that it's hardly ever done right. While we await a catastrophe, simpler setup is the most important step toward better security. In a distributed system with no central management like the Internet, security requires a clear story about who is trusted for each step in establishing it, and why. The basic tool for telling this story is the "speaks for" relation between principals that describes how authority is delegated, that is, whotrusts whom. The idea is simple, and it explains what's going on in any system I know. The many different ways of encoding this relation often make it hard to see the underlying order.

I INTRODUCTION

Computer security , also known as cyber security or IT security, is the protection of information systems from theft or damage to the hardware, the software, and to the information on them. It involves preventing physical access to the hardware, along with protecting against harm that may come via network access, data and code injection, and due to malpractice by operators, whether intentional, accidental, or due to them being tricked into deviating from secure procedures.

The field is of growing importance due to reliance on computer systems and the Internet in most societies, wireless networks such as Bluetooth and Wi-Fi - and the growth of "smart" devices, including smartphones, televisions and tiny devices as part of the Internet of Things.

II THE BASIC COMPONENTS

Computer security rests on availability confidentiality, integrity. The interpretations of these three aspects vary, as do the contexts in which they arise. The interpretation of an aspect in a given environment is dictated by the needs of the individuals, customs, and laws of the particular organization.

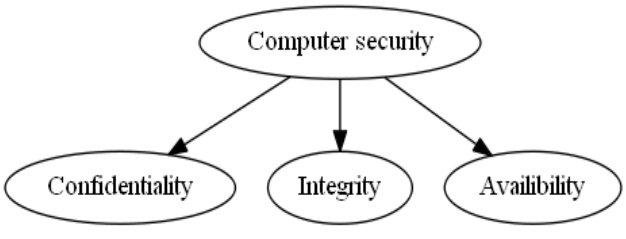


Fig. 1 – Component of computer security

2.1 Confidentiality:- Confidentiality is the hiding of information or resources. The need for keeping information secret arises from the use of computers in sensitive fields such as government and industry.

For example, civilian institutions in the government often restrict access to information to those who need that information. The first formal work in computer security was motivated by the military's attempt to implement controls to enforce a "need to know" principle.As a further example, all types of institutions keep personnel records secret.

Access control mechanisms support confidentiality .A *cryptographic key*controls access to the unscrambled data, but then the cryptographic key itself becomes another datum to be protected.

2.2 Integrity:- Integrity refers to the trust worthiness of data or resources, and it is usually phrased in terms of preventing improper or unauthorized change. Integrity includes data integrity (the content of the information) and origin integrity (the source of the data, often called *authentication*). The source of the information may bear on its accuracy and credibility and on the trust that people place in the information. This dichotomy illustrates the principle that the aspect of integrity known as credibility is central to the proper functioning of a system. We will return to this issue when discussing malicious logic.

Integrity mechanisms fall into two categories: *prevention*m echanisms and *detection*mechanisms.

Prevention mechanisms seek to maintain the integrity of the data by blocking any unauthorized attempts to change the data or any attempts to change the data in unauthorized ways.The former occurs when a user tries to change data which she has no authority to change. The latter occurs when a user authorized to make certain changes in the data tries to change the data in other ways.

Detection mechanisms may analyze system events (user or system actions) to detect problems or (more commonly) may analyze the data itself to see if required or expected constraints still hold. The mechanisms may report the actual cause of the integrity violation (a specific part of a file was altered), or they may simply report that the file is now corrupt.

*2.3 Availability :-*Availability refers to the ability to use the information or resource desired. Availability is an important aspect of reliability as well as of system design because an unavailable system is at least as bad as no system at all. The aspect of availability that is relevant to security is that someone may deliberately arrange to deny access to data or to a service by making it unavailable. System designs usually assume a statistical model to analyze expected patterns of use, and mechanisms ensure availability when that statistical model holds. Someone may be able to manipulate use (or parameters that control use, such as network traffic) so that the assumptions of the statistical model are no longer valid. This means that the mechanisms for keeping the resource or data available are working in an environment for which they were not designed. As a result, they will often fail. Attempts to block availability, called *denial of service attacks*, can be the most difficult to detect, because the analyst must determine if the unusual access patterns are attributable to deliberate manipulation of resources or of environment. Complicating this determination is the nature of statistical models. Even if the model accurately describes the environment, atypical events simply contribute to the nature of the statistics. A deliberate attempt to make a resource unavailable may simply look like, or be, an atypical event. In some environments, it may not even appear a typical.

III OVERVIEW OF COMPUTER SECURITY

Like any computer system, a secure system can be studied under three headings: Specification: What is it supposed to do? Implementation: How does it do it? Correctness: Does it really work?

In security they are called policy, mechanism, and assurance, since it's customary to give new names to familiar concepts. Thus we have the correspondence:

Specification, Assurance, Correctness, Policy,Mechanism, Implementation.

Assurance is especially important for security because the system must withstand malicious attacks, not just ordinary use. Deployed systems with many happy users often have thousands of bugs. This is possible because the system enters very few of its possible states during normal use. Attackers, of course, try to drive the system into states that they can exploit, and since there are so many bugs, this is usually quite easy.

3.1 Policy: Specifying Security

Organizations and people that use computers can describe their needs for information security under four major headings:

•Secrecy: controlling who gets to read information.

•Integrity: controlling how information changes or resources are used.

•Availability: providing prompt access to information and resources.

•*Accountability:* knowing who has had access to information or resources.

They are usually trying to protect some resource against danger from an attacker. The resource is usually either information or money.

Each user of computers must decide what security means to them. A description of the user's needs for security is called a security policy.

Most policies include elements from all four categories, but the emphasis varies widely. Policies for computer systems are usually derived from policies for security of systems that don't involve computers. The military is most concerned with secrecy, ordinary businesses with 4 integrity and accountability, telephone companies with availability. Obviously integrity is also important for national security: an intruder should not be able to change the sailing orders for a carrier, and certainly not to cause the firing of a missile or the arming of a nuclear weapon. And secrecy is important in commercial applications: financial and personnel information must not be disclosed to outsiders. Nonetheless, the difference in emphasis remains.

A security policy has both a positive and negative aspect. It might say, "Company confidential information should be accessible only to properly authorized employees". This means two things: properly authorized employees should have access to the information, and other people should not have access. When people talk about security, the emphasis is usually on the negative aspect: keeping out the bad guy. In practice, however, the positive aspect gets more attention, since too little access keeps people from getting their work done, which draws attention immediately, but too much access goes undetected until there's a security audit or an obvious attack,2 which hardly ever happens. This distinction between talk and practice is pervasive in security.

3.2 Mechanism: Implementing

Of course, one man's policy is another man's mechanism. The informal access policy in the previous paragraph must be elaborated considerably before it can be enforced by a computer system. Both the set of confidential information and the set of properly authorized employees must be described precisely. We can view these descriptions as more detailed policy, or as implementation of the informal policy.

In fact, the implementation of security has two parts: the code and the setup or configuration. The code is the programs in the trusted computing base. The setup is all the data that controls the operations of these programs: access control lists, group memberships, user passwords or encryption keys, etc.

The job of a security implementation is to defend against vulnerabilities. These take three main forms:

1) Bad (buggy or hostile) programs.

2) Bad (careless or hostile) agents, either programs or people, giving bad instructions to good but gullible programs.

3)Bad agents tapping or spoofing communications.

Broadly speaking, there are four defensive strategies:

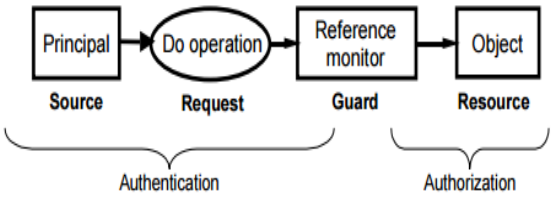
1) Keep everybody out.

2) Keep the bad guys out.

3) Let the bad guys in, but keep them from doing damage.

4) Catch the bad guys and prosecute them.

The well-known access control model as shown in fig. 2 provides the framework for these strategies. In this model, a guard controls the access of requests for service to valued resources, which are usually encapsulated in objects.



The guard's job is to decide whether the source of the request, called a principal, is allowed to do the operation on the object. To decide, it uses two kinds of information: authentication information from the left, which identifies the principal who made the request, and authorization information from the right, which says who is allowed to do what to the object.

For instance, if you want a file system to enforce quotas only for novice users, there are only two ways to do it within this model:

1) Have separate methods for writing with quotas and without, and don't authorize novice users to write without quotas.

2) Have a separate quota object that the file system calls on the user's behalf.

Of course security still depends on the object to implement its methods correctly. For instance, if a file's read method changes its data, or the write method fails to debit the quota, or either one touches data in other files, the system is insecure in spite of the guard.

Another model is sometimes used when secrecy in the face of bad programs is a primary concern: the *information flow control model*as shown in fig3 [6,14]. This is roughly a dual of the access control model, in which the guard decides whether information can flow to a principal.

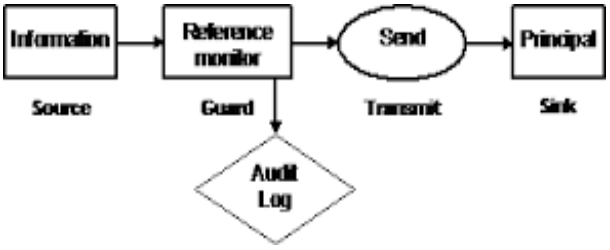


Fig. 2 – The Information Flow Control Model

In either model, there are three basic mechanisms for implementing security. Together, they form the gold standard for security:

• Authenticating principals, answering the question "Who said that?" or "Who is getting that information?". Usually principals are people, but they may also be groups, machines, or programs.

• Authorizing access, answering the question "Who can do which operations on this object?". • Auditing the decisions of the guard, so that later it's possible to figure out what happened and why.

3.3 Assurance: Making security work
The unavoidable price of reliability is simplicity.

What does it mean to make security work? The answer is based on the idea of a trusted computing base (TCB), the collection of hardware, software, and setup information on which the security of a system depends.

The idea of a TCB is closely related to the end-to-end principle — just as reliability depends only on the ends, security depends only on the TCB. In both cases, performance and availability isn't guaranteed.

In general, it's not easy to figure out what is in the TCB for a given security policy. Even writing the specs for the components is hard, as the examples may suggest.

For security to work perfectly, the specs for all the TCB components must be strong enough to enforce the policy, and each component has to satisfy its spec. This level of assurance has seldom been attempted. Essentially always, people settle for something much weaker and accept that both the specs and the implementation will be flawed. Either way, it should be clear that a smaller TCB is better.

A good way to make defects in the TCB less harmful is to use defense in depth, redundant mechanisms for security. For example, a system might include:

• Network level security, using a firewall.

• Operating system security, using sandboxing to isolate programs. This can be done by a base OS like Windows 2000 or Unix, or by a higher-level OS like a Java VM.

• Application level security that checks authorization directly.

The idea is that it will be hard for an attacker to simultaneously exploit flaws in all the levels. Defense in depth offers no guarantees, but it does seem to help in practice.

Most discussions of assurance focus on the software (and occasionally the hardware), as I have done so far. But the other important component of the TCB is all the setup or configuration information, the knobs and switches that tell the software what to do. In most systems deployed today there is a lot of this information, as anyone who has run one will know. It includes:

1) What software is installed with system privileges, and perhaps what software is installed that will run with the user's privileges. "Software" includes not just binaries, but anything executable, such as shell scripts or macros.

2) The database of users, passwords (or other authentication data), privileges, and group memberships. Often services like SQL servers have their own user database. 3) Network information such as lists of trusted machines.

4) The access controls on all the system resources: files, services (especially those that respond to requests from the network), devices, etc.

5) Doubtless many other things that I haven't thought of.

IV TERMINOLOGY

- Access authorization restricts access to a computer to group of users through the use of authentication systems. These systems can protect either the whole computer such as through an interactive login screen or individual services, such as an FTP server. There are many methods for identifying and authenticating users, such as passwords, identification cards, and, more recently, smart cards and biometric systems.
- Anti-virus software consists of computer programs that attempt to identify, thwart and eliminate computer viruses and other malicious software (malware).
- Applications with known security flaws should not be run. Either leave it turned off until it can be patched or otherwise fixed, or delete it and replace it with some other application. Publicly known flaws are the main entry used by worms to automatically break into a system and then spread to other systems connected to it. The security website Secunia provides a search tool for unpatched known flaws in popular products.
- Authentication techniques can be used to ensure that communication end-points are who they say they are.
- Automated theorem proving and other verification tools can enable critical algorithms and code used in secure systems to be mathematically proven to meet their specifications.
- Capability and access control list techniques can be used to ensure privilege separation and mandatory access control. This section discusses their use.
- Chain of trust techniques can be used to attempt to ensure that all software loaded has been certified as authentic by the system's designers.
- Confidentiality is the nondisclosure of information except to another authorized person.
- Cryptographic techniques can be used to defend data in transit between systems, reducing the probability that data exchanged between systems can be intercepted or modified.
- Cyber warfare is an internet-based conflict that involves politically motivated attacks on information and information systems. Such attacks can, for example, disable official websites and networks, disrupt or disable essential services, steal or alter classified data, and cripple financial systems.
- Data integrity is the accuracy and consistency of stored data, indicated by an absence of any alteration in data between two updates of a data record.

V CONCLUSION

Computer security attempts to ensure the confidentiality, integrity, and availability of computing systems and their components. Three principal parts of a computing system are subject to attacks: hardware, software, and data. These three, and the communications among them, are susceptible to computer security vulnerabilities. In turn, those people and systems interested in compromising a system can devise attacks that exploit the vulnerabilities.

In this we have explained the following computer security concepts:

- Security situations arise in many everyday activities, although sometimes it can be difficult to distinguish between a security attack and an ordinary human or technological breakdown. Alas, clever attackers realize this confusion, so they may make their attack seem like a simple, random failure.
- A threat is an incident that could cause harm. A vulnerability is a weakness through which harm could occur. These two problems combine: Either without the other causes no harm, but a threat exercising a vulnerability means damage. To control such a situation, we can either block or diminish the threat, or close the vulnerability (or both).
- Seldom can we achieve perfect security: no viable threats and no exercisable vulnerabilities. Sometimes we fail to recognize a threat, or other times we may be unable or unwilling to close a vulnerability. Incomplete security is not a bad situation; rather, it demonstrates a balancing act: Control certain threats and vulnerabilities, apply countermeasures that are reasonable, and accept the risk of harm from un countered cases.
- An attacker needs three things: method—the skill and knowledge to perform a successful attack; opportunity—time and access by which to attack; and motive—a reason to want to attack. Alas, none of these three is in short supply, which means attacks are inevitable.

In this chapter we have introduced the notions of threats and harm, vulnerabilities, attacks and attackers, and countermeasures. Attackers leverage threats that exploit vulnerabilities against valuable assets to cause harm, and we hope to devise countermeasures to eliminate means, opportunity, and motive. These concepts are the basis we need to study, understand, and master computer security.

Countermeasures and controls can be applied to the data, the programs, the system, the physical devices, the communications links, the environment, and the personnel. Sometimes several controls are needed to cover a single vulnerability, but sometimes one control addresses many problems at once.

REFERENCES

[1] https://en.wikipedia.org/wiki/Computer_security
[2] https://www.digitaldemocracy.org/about
[3] http://www.informit.com/articles/article.aspx?p=30710
[4] http://www.justasc.net/portfolio-view/security-architecture-and-design/
[5] http://research.microsoft.com/en-us/um/people/blampson/64-SecurityInRealWorld/Acrobat.pdf
[6] http://www.informit.com/articles/article.aspx?p=2301451&seqNum=6