The flawed assumption of security in modern computing environments essay sample

Law, Security

Security in operating systems has been an important matter within military organisations for decades. Nowadays commercial companies are more concerned about their IT security, including secure operating systems. Operating systems are the prime providers of security in computing systems. Because they have such power they are sometimes targets for attacks. Breaking throw the defence of an operating system gives access to the secrets of computing.

Today’s operating systems are more sophisticated and feature-rich than ever before, which makes them substantially more useful to the enterprise but also adds to security vulnerability—unless the operating systems are configured, administered and monitored correctly. Contrary to popular belief, this can be accomplished with a minimum of fuss and bother. The key is to centralize and automate operating system security across the enterprise, rather than do it manually for each box.

In fact, the costs and risks of not centralizing and automating operating system security are enormous. Over half of the security break-ins we read about daily are the result not of inherent weaknesses in operating system technology but of operating systems not being configured properly or not being verified and monitored regularly. The operating systems were provisioned out of the box at the default security settings, which made them highly vulnerable to attack.

Today, roughly 20% of user identifications and passwords have never been changed. The word “ password” is still a common password in many organizations. The reason administrators neglect to configure these settings

properly is simple: It would take approximately 20, 000 hours to provision and verify a 1, 000-server network manually, as it must be done in many organizations, and few organizations can afford the necessary time and money.

The main idea of the “ The Inevitability of Failure: The Flawed Assumption of Security in Modern Computing Environments” issued by National Security Agency is to attract public awareness to the necessity of creation of a secure operating system and to that fact that single security mechanisms out of entire secure operating system isn’t able to provide the appropriate protection.

To that purpose the authors give the description of the differences between mandatory and discretionary security policy and the necessity of using of mandatory security policy which is necessary to (National Security Agency : n. p.): “ ensure that security mechanisms are applied as required and can protect the user against inadvertent execution of untrustworthy applications.”

A trusted path as one of the most efficient security points is also discussed. This mechanism can’ be imitated by other software and is a guarantee of user’s software safeness. Each operating system is to provide its own trusted path which would be closer to this operating systems needs precisely. Though, it is said also that trusted path mechanisms must be more extensible in order to support the system policy administrator’s trusted applications.

The enforcer component and decider component as the integral parts of an application-space access control are also presented as the foundation of any security system. The necessity of the mandatory security mechanisms is also emphasized here as the later can ensure that all the accesses are controlled by the enforcer component. The main thing here is to eliminate any kind of possibilities for malicious agent to impersonate the decider component to the enforcer one what can lead to the subverting of the system.

Regarding Cryptography the two main vulnerabilities are given. The first is that legitimate attempt to use a the token may have no results as the application by means of which the invocation is realized may be bypassed by malicious activities which are able to impersonate the cryptographic token to the invoking application. The second vulnerability is that the misuse of the cryptographic token may cause the use of a service by an unauthorized application. The direct physical interface for user activation can’t result not to be efficient in this case as malicious software is able to spoof the token to the user and get the information that is necessary for authentication. The resolving of this problem is suggested to be in supporting by the operating system of the mandatory security and trusted path and protected path. These facilities help to provide a separate physical interface for activation and identification of the callers. Besides, the fine-grained control over the use of services, algorithms and keys can be strongly implemented.

A lack of operating system support for security makes vulnerable such a security mechanism as “ mobile code”. Java is taken as an example of Security Model. The security facilities as to this model are based on the type-

safety of the language. But the Java Virtual Machine can accept byte code which is able to violate the language semantics. Besides, a considerable part of the Java system is presented in the form of native methods which aren’t checked by the JVM.

Another mode of securing mobile code is the requiring of digitally signed applets. The main weakness of this kind of protection is an all-or-nothing proposition which results in user’s inability to conform a native ActiveX control, which is based on digital signatures, to a limited security domain. To do that the mandatory security mechanisms are also required. Nevertheless, digital signatures are considered to be necessary even on secure operating systems. Despite their limited application they reduce the risk of the entering the system by malicious code and provide another piece of information for an access control decision. The necessity in use of the trusted path mechanism is discussed here again. J. Tyger and A. Whitten in their “ WWW Electronic Commerce and Java Trojan Horses.” speak about a Trojan horse that can easily capture various kinds of data. The proposed ad hoc solution wasn’t successful as it increased the sophistication which is greeted by Trojan horse. Therefore, trusted path is suggested to be efficient in this case.

The conclusion made in this chapter by National Security Agency is that a secure system will not only increase the above mentioned security mechanisms’ efficiency but considerably simplify them providing in the course of that a better overall system.

Kerberos, Network Security Protocols (IPSEC) and SSL are suggested to provide authentication and confidentiality services. But all these are non-less vulnerable than the previous secure services. Kerberos' server application can't establish a trusted path to a user. Therefore, malicious software executed by a user can freely modify user's information. IPSEC and SSL in its turn are suggested to create secure channels what requires secure end points as well. So, if the malicious user penetrates one of the end points he gets access to all the data.

Firewalls have been designed to provide a separation between insiders and outsiders for an organization's computing software. But firewalls do not provide any protection against malicious insiders. These can easily make use of the information and built up channels for outsiders' calls.

In conclusion the authors speak about the necessity of creation of a secure operating system whose security mechanisms would provide specific security services and would complement each other in order to (National Security Agency 2005: n. p.): " provide a complete security package." " No single security mechanism is likely to provide complete protection."

The ideas described in the research above can be interpreted and evaluated in different ways. On the one hand, the question of operating systems security has been actual all along the last 20 years at least and it's hard to define this research as some purely new investigation in this point. On the other hand, this kind of research becomes more and more necessary and actual in the view of computing technology fast progress and obvious

dependence of the principal services on interconnected systems. Furthermore, a great value of this research consists in that fact that it has made a focus of its effort the most important vulnerabilities of the computing systems of present days. The authors of the research have managed to compile the most vital points which are to be highlighted today with reference to the operating systems works.

Speaking about the main idea of the research there is a question which would be useful to answer in order to evaluate it. Has the secure operating system to be created or may be the single secure mechanisms can be just implemented?

The Operating System itself is characterized by a range of particularities. An Operating System supports multiprogramming that means more than one person using a system at a time. That’s why system designers have to protect one user from malicious interference with one other. When more than one person uses a system does it contain more risks and complexity. Multiprogramming means that several objects have to be protected. These objects are: memory, sharable I/O devices (such as disks), serially reusable I/O devices (such as printers and tape drives), sharable programs and sub-procedures, networks and sharable data.

The most important thing to think about is to separate one user from others. This can occur in different ways such as physical, temporal, logical and cryptographic separation. But we also want users to share some objects. We

would like users with different security levels to be able to invoke the same algorithm.

One problem is to prevent one program from affecting the memory of other programs. Fence is the simplest form of memory protection. It is a method to confine subjects to one side of a boundary. The fence is a predefined memory address. A fence can protect the operating system from one single user but cannot protect one user from another user. In a multi-user environment is it especially important to be able to relocate. You can’t know where a program will be loaded for execution. The relocation register solves the problem by providing a base or starting address. When execution changes from one user’s programs to another’s the operating system must change the contents of the base and bounds registers to reflect the true address space for that user. This is a part of the context switch that the operating system performs when changing from one user to another.

In the original IBM OS operating systems files were public. Any user could read, modify or delete a file belonging to another user. Designers thought that users could trust and respect each other’s files. Sensitive files were protected by passwords. You could either do whatever you wanted (read, write, delete etc.) to a file or nothing at all. There are many reasons why this didn’t work out, for example lack of trust, all or nothing access is not flexible and complexity. A way to solve this problem was to identify groups with the same access to a file. This solution is easy to implement but has it drawbacks. A user can only belong to one group and can’t access other group’s files.

The enumerated particularities above must be taken into consideration while using a secure operating system which must contain mandatory security policy and trusted path.

Brockmeier , 2003(2): n. p.):“ Trusted operating systems must be used to implement multi-level security systems and to build security guards that allow systems of different security levels to be connected to exchange data. Use of a trusted operating system may be the only way that a system can be networked with other high security systems.” Professionals prefer to say trusted operating systems instead of secure. Because something is either secure or not secure. It can’t be “ half secure”. There aren’t any operating systems that can withstand all attacks, today, tomorrow, and in ten years. A trusted operating system meets the intended security requirements. Trust is perceived by the system’s users and not by the developers or manufactures. As a user you may not experience that trust at once. There can be different degrees of trust.

In a trusted operating system is the objects surrounded by access control, offering more protection and separation than an ordinary one. In addition, memory is separated by user, and data and program libraries have controlled sharing and separation. User identification and authentication is one of the key features in computer security, in operating systems as well. Most access control is based on accurate identification. Trusted operating systems require secure identification of individuals, and each individual must be uniquely identified.

Mandatory security is a concept often associated with multi-level security policy used by the Department of Defence. A more general definition is any security policy where the definition of the policy logic and the assignment of security attributes are tightly controlled by a system security policy administrator. In the operating systems, mandatory security can be divided into several kinds of policies. Examples of such policies are an access control policy, an authentication usage policy and a cryptographic usage policy. Mandatory access control specifies how subjects may access objects under the control of the operating system. Mandatory authentication describes what authentication mechanisms must be used to authenticate usage to the system. Protecting of data can be specified by a mandatory cryptographic usage policy.

Concerning difficulties that can appear while creating a secure operating system one must remember that for using the mandatory security policies the secure system must provide a framework for translating polices to underlying security mechanism. Without such a framework, there can be no real confidence that the mandatory security mechanisms will provide the desired security properties.

Some applications require special privileges in order to perform some security-relevant function. Such applications are frequently called trusted applications because they are trusted to perform correctly and not misuse privileges. One way to reduce the dependency on trusted applications is to use mandatory security mechanisms designed to only support minimal privileges for functionality. Type enforcement is an example of such a security mechanism which may be used to limit trusted applications to the

minimal set of privileges for functionality and harm they can do by misuse of these privileges.

The creation of the above mentioned framework is the principal point in this area. It’s necessary to create a system which would unite all the features of previous single secure mechanisms. But that’s not a subject of this research. That’s the task of computing engineers. In case of creating of same the answer to the question is obvious – the Secure Operating System will protect much better then any implemented single securing mechanisms.

Another important question regarding the research of the National Security Agency is whether it can be adequately realized in practice. (Brockmeier 2003(1): n. p.): “ There are certainly no formal guidelines that you can follow to obtain the level of security that we’ve obtained. It’s safe to say we are more secure than some of the standards-based guidelines that some follow.” Of course, the research in case doesn’t supply users with exact instructions how to create and use this or that kind of secure operating system. Actually, this research is more of theoretical kind rather than of a practical one. But it hasn’t as its purpose to be a precise instruction. Its aim was to justify the necessity in secure operating systems. And this aim has been completely reached.

The idea submitted in this research is already in use. More and more calls are received all over the world referring creation of secure operating system. A number of projects have already been created. It’s hard to say whether they completely correspond to the requirements of the National Security

Agency because of its not too wide use yet but these first steps are necessary for further development of this area. For example, (Carnegie Institute 2004: n. p.): “ the project of the U. S. Defence Advanced Research Projects Agency (DARPA) that has dedicated US$2. 3 million in funding to Open BSD over the next two years.” Doubtlessly, in the nearest future the creating of secure operating systems will become an ordinary proceeding. The research “ The Inevitability of Failure: The Flawed Assumption of Security in Modern Computing Environments” has become a considerable event in the world of secure computing. Many thought and comments has been issued regarding this paper. Of course, as it was already said it is not a practical instruction but rather a theoretical foundation with empiric elements. But the creation and use of the Secure Operating System is due in considerable part to the researches like this.

Bibliography

1. National Security Agency. *The Inevitability of Failure: The Flawed Assumption of Security in Modern Computing Environments.* http://www. nsa. gov/selinux/papers/inevitability (6 Oct. 2005)
2. Brockmeier, J. 2003(1). *Inside the World of Secure.* http://www. newsfactor. com/perl/story/21212. html (11 Oct. 2005)
3. Brockmeier, J. 2003(2). *Inside the world of Secure Operating. From: News Factor Magazine.* http://www. newsfactor. com/perl/story/21212. html (10 Oct. 2005)

4. Carnegie Mellon Software Engineering Institute. 2004. *Trusted Operating Systems.* http://www. sei. cmu. edu/str/descriptions/trusted_body. html (10 Oct. 2005)