Challenges in Operating System and Network Security
Literature Survey for CPSC 538F

Tim Wellhausen
Department of Computer Science
University of British Columbia

March 24, 1999

1 Introduction

This literature survey is motivated by the number of issues that still exists in the area of operating system and network security. Although there are many well-known protocols and specifications, almost no system can be considered to be absolutely secure. Maybe the reason is that absolute security is not possible. Still, it is desirable to achieve the best possible security for every system.

Because this area has so many different aspects, I picked three different papers about security. The first paper, "Research Challenges in Operating Systems" gives an broad overview of the topic. The second paper, "The DGSA: Unmet Information Security Challenges for Operating System Designers", describes a specific security architecture. Both papers were written for the American Department of Defense. Thus, the requirements for security are very high.

The third paper, "Secure Public Internet Access Handler", is about an implemented system to protect public ports to a network in an university environment. Therefore, the requirements for security are completely different.

In the following three sections, I explain the content of each paper, and section five concludes my survey.

2 DARPA/NSA/DISA Joint Technology Office: Research Challenges in Operating System Security

Security is still an area of operating systems that is in need of further research. Many "secure" systems and algorithms already exist, but there are still many challenges. This paper gives an overview of the most important areas which have to be addressed. It is split in the following five sections: Technology-Security Interplays, Security Policy, Assurance, Network-OS Interaction, and Risk Reduction.

2.1 Technology-Security Interplays

New concepts in the design of operating systems lead to new challenges for security. Traditionally, security in an operating system was achieved by strictly separating the applications from each other and from the kernel, so that security mechanisms inside the kernel could establish a secure base. But there are several new technologies that interfere with this traditional concept.

Service Co-location, for example, achieves a better performance by co-locating tasks in a single address space. This raises the problem of domain protection. Several proposals for this problem exist, including software fault isolation, type-safe languages, virtual machines, and interpretation of code. Whereas these concepts address the issue of protection, it is still a question how security policies can be enforced efficiently.

Another new technology is User-Level Resource Management. This could lead to applications that run in its own customized operating system environments. Although this gives applications more control over its resources, there are concerns about securely sharing resources. Furthermore, it seems to be quite difficult to formulate and enforce security policies uniformly in such a case.

The authors propose the well-known concept of separating mechanism and policy. Whereas the kernel should provide policy-neutral mechanisms, servers should implement specific security services.

Another issue is the trustworthiness of compiler technology. Especially when protection highly depends on compiler enforcement, it is crucial that these compilers work flawlessly and that they are trusted. Even virtual machines have security holes as shown by the Java virtual machine. It is almost impossible to achieve high security in new technologies. Nevertheless, these problems have to be addressed if the technologies should be used in new security products.

Distributed computing is another concept that becomes more important. Although this doesn't introduce fundamental new security challenges, existing technologies have to be used properly. The key to achieve secure distributed technology lies in building a solid infrastructure of security services. The goal

should be to create a framework of security services that can be changed independently from the applications that use this framework.

Finally, the authors address the problems between security and fault tolerance. Fault tolerance and recovery algorithms are very complex and difficult. Most of the time they rely on global access to information and global management of resources. This is partly contradictory to security principles of isolation and least privilege. In this area, there is still a lot of research work to do.

2.2 Security Policy

As long as security policies have to be only enforced within one domain of protection, this is not very difficult. But now that most computers are connected with networks, it is more important to create rules how data can be securely shared and exchanged between different domains, especially if different security policies apply.

Before data can be shared, the security policies of all domains involved have to be known. A decision must be made whether the policy of the target domain meets the desired standard. If this should be done automatically, the problem arises how to compute the security properties of two systems with known but different security properties.

As a part of every network, there must be an authorization and access control framework. Many systems, like for example Kerberos, are in use and many algorithms are well-kown. Nevertheless, there are only few standards for such systems. There is a need to generalize the application program interfaces, so that it is possible to change the underlying system without changing the applications. Furthermore, it might be desired to use different systems for one domain. One for communication inside the domain and one for communication to and from the domain which has to have stronger protection.

A network also requires access to the user's security attributes. Public key algorithms have brought progress in this area, but they are not the best solution. Public keys are bound to information about a user such as his name and organization. Without a global naming scheme, this creates potential mis-matches both in form and context of the attribute information.

Last but not least, there is need to have a uniform access control framework. Often, many different policies are needed for different purposes. There should be an operating system mechanism to support the widest possible range of policies. Configuration of and changes to these policies must be easy to do.

2.3 Assurance

The new paradigm in software engineering, creating independent components, leads to new challenges in assuring safety, security, timeliness, reliability, and other properties. It is necessary to extend the known methods of assuring these

properties.

The authors talk about the importance of assurance metrics as a standard for the degree to which a system can provide the desired properties. Although the so-called Orange Book defines several classes of requirements to ensure security, those requirements can not be a measure of the security of a resulting operating system. Additional metrics are needed, for example, for the development process or the use of the system.

Different methods exist to assure the properties of a system. These methods include verification by formal reasoning of the program code. Similar techniques can be applied for the information flow in the system or to generate tests from formal specifications.

A particular problem are covert channels. These channels normally use shared resources to transfer data in a way that was not intended by the designers. Therefore, these channels work outside any specification. Such a channel might be, for example, an artificial delay in a computation which is supposed to take a specific time under normal conditions. Finding and removing these channels is a big problem that still has to be addressed.

2.4 Network-Operating System Interaction

Currently, a great number of security protocols for network communication exists. These protocols are well-known and assumed to be correct and secure. Nevertheless, in a typical environment different protocols have to be used. This leads to problems of interaction between the protocols (creating different publics keys for similar protocols, etc.). Furthermore, some of the services may be redundant. The authors state that ultimately, security services should be another parameter in a quality of service negotiation, taking into account the delays for encryption, etc.

Another challenge is mobility. Both mobile users (users working within groups with different security requirements) and mobile computers (computers which might be connected to different groups) raise new issues of security.

Before computers were connected with networks, all necessary security information could be stored locally on the computers. Now, many services are executed either in different places in an operating system or even on different computers. Thus, managing security information is an important issue: A system can only be as secure as secure the security information is stored.

2.5 Risk Reduction

In the last section of the paper, the authors discuss several ways to prevent and detect intruders. A secure system needs to collect data and to offer mechanisms to analyze data about the running system. This includes logging all kind of important information. For such analysis to be trustworthy, it must be impossible

for an intruder to deactivate the mechanisms or to change the collected data. Furthermore, the audit process must be easy to use and efficient.

The authors also talk about the research of operating systems themselves. They argue that it is expensive and takes a long time to research new concepts of operating systems. Therefore, an infrastructure has to be created that allows early, rapid, and accurate analysis of the benefits and consequences of innovative operating system concepts.

3 Feustel, Mayfield: The DGSA: Unmet Information Security Challenges for Operating System Designers

This paper describes important aspects of the Department of Defense (DoD) Goal Security Architecture (DGSA). This architecture provides a framework for secure information access, processing, and distribution.

Different from former approaches, the DoD acknowledges the importance of open systems. Even for the DoD, it is not possible any more to rely only on systems that are made according to their own requirements. Instead, the DGSA recommends to use public standards and implementations of the Internet Engineering Task Force (IETF) Internet Protocol Security Drafts to deal with public networks, encryption, etc.

3.1 Requirements

The new requirements of the DGSA state that information systems for the DoD must (1) support information processing under multiple security policies of any complexity or type, (2) be sufficiently protected to allow distributed information processing among multiple hosts on multiple networks in accordance with open system architectures, (3) support information processing among users with different security attributes, and (4) be sufficiently protected to allow connectivity via common carrier (public) communications systems.

The use of public communication systems is a move away from dedicated circuits. Public networks cannot be controlled by the DoD. Therefore, the only requirement that can be made is availability. To protect the flow of information, the end points of the communication must be under physically and logically control of the DoD. These end systems must be inside a safe and protected environment.

The DoD requires that multiple security policies can be applied to protect information of different sensitivity. It must be possible that information objects can be accessed and/or used simultaneously by one or more users based on levels of clearance and access to compartments of information. The DoD has realized

that its requirements are not very different from those requirements of commercial organizations. Therefore, implementations should support security policies with respect to confidentiality, integrity, and availability.

The words "sufficient protection" are another change in the requirements of the DoD. The DoD used to examine assertions about protection at the time of certification and to assume that the assertions remain true despite changes to critical components. Now, the DoD focuses on risk management and the value of the information to be protected and concentrates on assuring that information remains protected.

The DGSA does not specify all details for an implementation. Instead, it establishes the minimum constraints of acceptable implementations by defining an information framework. The design of the operating system's architecture is left to the implementors.

3.2 Information Domains

Information domains are used to achieve the desired goals of flexibility and multiple security policies. An information domain contains information objects, users (called principles), and a single information security policy.

An information object can be a file or any kind of data which can be accessed. Every information object is a member of exactly one information domain and has security attributes that are the same among all objects within a domain. Thus, it is not necessary to distinguish sensitivity of objects while operating on them inside a domain. All objects are at the same sensitivity level within one information domain.

Only when an object is to be exported into another domain, the policy regarding sensitivity must be consulted. Therefore, the main challenge in implementing such a system is to assure that there are mechanisms to transfer objects between information domains that may be on different platforms and inside different systems.

Each principle also has a set of attributes. These attributes define the rights a principle has to get access to objects. Once more, this is controlled by the security policy of the information domain.

Each information domain defines explicit relationships to other information domains. If these relationships are not provided, domains must be strictly isolated. Only if such a relationship between two domains exist, one domain allows the export to the other domain, and the other one allows the import, then the transfer may be performed.

3.3 Security Policies

The DGSA does not explicitly describe the security policies that may be implemented. But it specifies that no implicit hierarchical trust or sensitivity

relationships can be inferred between information domains. But this makes it still possible to adopt a wide range of common security policies. The authors refer, for example, to the ISO Security Framework.

In the ISO Security Framework several services are defined. These include Authentication and Identification, Access Control, Confidentiality, Integrity, Availability, and Audit and Alarms.

A possible policy for functional access control levels is given in reference to a paper from Saltzer and Schroeder: There, the following levels of access exist: Unprotected Systems with only one public information domain, All-or-nothing Systems that have one private information domain where only authenticated users are accepted, Controlled Sharing which implies that different access rights on each object requires multiple information domains, User-Programmed Sharing Controls that has separate information domains for protected objects with a domain per content type or per security policy, and Labeling Information with one domain per distinct label.

3.4 Implications for Implementations

Depending on the class of security policies to be implemented, the design of the operating system may be simple or hard. For some policies, a virtual machine model may be sufficient. If distribution of information objects is to be allowed, these machines may be linked by a Virtual Security Network. Here, the authors refer again to the IETF Security Protocol suite of specifications and to CORBA. For more sophisticated policies, the authors recommend to use existing software architectures like message-oriented Object Frameworks.

Because of the way the DGSA defines relationships between information domains, the management of the information domains is a key issue for the implementors and users. Furthermore, the are several issues how to create security policies and who is allowed to change them. The authors suggest to design a collection of plugable security modules which implement the policies. Another concern is audit. Implementation must be done so that the audit process and its performance will not lead to turning the audit off.

3.5 Research and Development Areas

One key point of the DGSA is showing that absolute protection is maintained as additional systems are added to the Virtual Secure Network. The authors address several areas which are in need of further research to achieve this goal.

One of the questions they pose is what procedures need to be followed to assure the tester/certifier that the security policy which is supposed to be in force is the one being carried out. Another area of interest is whether it is possible to standardize the expression of policy so that this expression may be used automatically to generate parameters or rules for an implementation of the

policy.

Furthermore, they talk about how communication between virtual machines can be implemented and refer to the Java Virtual Machine as an option.

4 Poger, Baker: Secure Public Internet Access Handler

Different network environments need different approaches for security and access. This paper describes the design and implementation of a system called SPINACH that was created according to requirements of a university department.

At Stanford University, the computer science department provides public ports to access the internal LAN inside their building. Because of security concerns, a system has been developed that establishes a "prisonwall" around the public ports. All packets leaving this subnet are filtered by a specific router. Only traffic from authorized users is allowed to pass the router, other packages are dropped. With minimal requirements both for users and the system software, this design provides a reasonable protection against attacks without unnecessary restrictions.

The SPINACH system consists of a number of public network ports building a public subnet and a router that runs slightly specialized software to grant or refuse access to the department network. A user who wants to connect to the network plugs his laptop into a port and has to authenticate himself. Afterwards, all packets coming from this host are allowed to go through the router.

The current security policy of the system filters only outgoing traffic. Traffic into the public subnet is considered to be not critical because unauthorized users can not initiate a connection. Traffic inside the subnet is also not affected. That means that every user that uses a public port has to know that his environment might be hostile.

SPINACH distinguishes between three different kinds of users: department users, university users, and guests. Whereas department users have full access to all resources as if they would be logged in a terminal, university users are not allowed to get access although authorizing them would be very similar to department users. Guests, who want to log in, need to get a pair of <userid, one-time password> to get access.

The connection works as follows: If a client has DHCP (Dynamic Host Configuration Protocol) installed, all important parameters like his IP number are set automatically. Otherwise, the user has to configure his computer manually. There are two different ways to authorize a user: University users are identified by a unique university identifier and can authenticate themselves when they have a Kerberos client installed. Otherwise, and in the case of guests, the user

has to open a telnet session to connect to the router. A modified telnet server on the router asks for username and password and disconnects the telnet session immediately afterwards. If the user input is accepted access is granted. This new session is then valid for a specific amount of time. After that time, the user has to re-authenticate himself.

On the client, a telnet is sufficient for authorization. On the router, some software has to be installed. A DHCP server makes it possible to easily configure the client. A modified telnet server is needed to authorize users with passwords. Authorization clients administrate the router to enable/disable network access and to generate guest passwords. Furthermore, a so-called prisonguard is needed, that is a process running at user-level that maintains the database about users and modifies the packet filter rules dynamically. Finally, there is a packet filter that it part of a Linux kernel.

The packet filter has to work according to rules that specify which packets are allowed to pass. All packets leaving through the SPINACH router are assumed guilty until proven innocent. That is, either they are destined for the trusted departmental DNS server (may be needed to find the Kerberos server), they are destined for the trusted Kerberos server, or they come from an authorized user.

The authors are well aware that this system does not provide high security. But it may be appropriate for environments where it is not possible (for example because of money constraints) to install specialized hardware or software to provide protection or where this kind of protection is sufficient. It doesn't prevent an authorized user to attack the system, but it denies access to the network for unauthorized users. Furthermore, SPINACH seems to be a system that is easy to install, configure, and maintain.

On the other hand, several attacks against SPINACH are possible. Because traffic inside the subnet is not filtered, in a shared Ethernet environment it is possible to eavesdrop both IP address and hardware address of a machine. With this information, spoofing attacks are possible. Furthermore, SPINACH is vulnerable to denial-of-service attacks against the Kerberos or DNS server.

Nevertheless, is seems to be a good balance between the need of temporary access to a network and protection of the network against malicious use.

5 Conclusion

This survey was meant to give an impression about current problems and research in operating system security. It is not possible in this survey to write about every aspect of this topic. Nevertheless, it became clear that there are many different areas where further research is necessary. Especially, it is important to review new technologies for their impact on security.

On the other hand, security is always relative to the requirements for a specific system. Different environments need different security policies. Therefore, it

is necessary to define the degree of desired security and implement the system according to it.

6 References

[1] DARPA/NSA/DISA Joint Technology Office: *Research Challenges in Operating System Security*, Proceedings of the DARPA/NSA Workshop on Operating System Security, May 22-23, 1996,
URL: http://www.darpa.mil/ito/Proceedings/OS_Security/challenges/challenges_long.html

[2] Feustel, Mayfield: *The DGSA: Unmet Information Security Challenges for Operating System Designers*, Operating Systems Review, ACM SIGOP, January, 1998, p. 3-22

[3] Poger, Baker: *Secure Public Internet Access Handler*, Proceedings of the USENIX Symposium on Internet Technologies and Systems, December 1997, URL: http://mosquitonet.stanford.edu/publications/spinach.ps