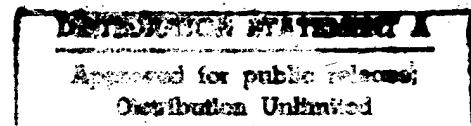




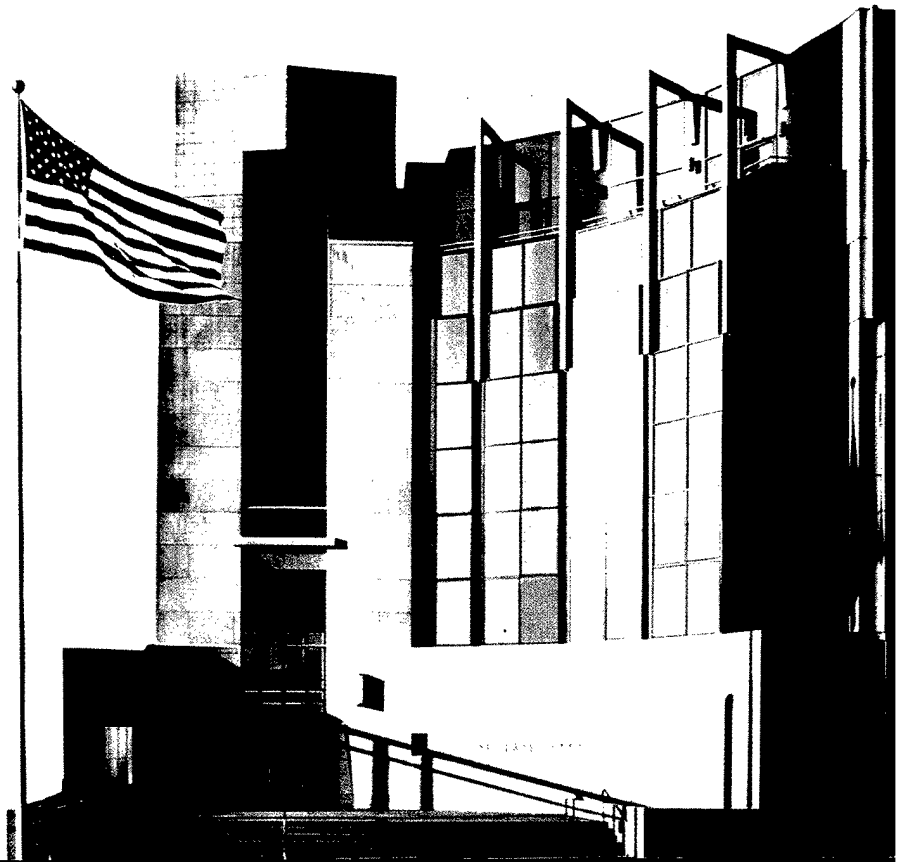
Carnegie Mellon
Software Engineering Institute

An Approach for Selecting and Specifying Tools for Information Survivability

Robert Firth
Barbara Fraser
Suresh Konda
Derek Simmel
July 1998



TECHNICAL REPORT
CMU/SEI-97-TR-009
ESC-TR-97-009



Carnegie Mellon University does not discriminate and Carnegie Mellon University is required not to discriminate in admission, employment, or administration of its programs or activities on the basis of race, color, national origin, sex or handicap in violation of Title VI of the Civil Rights Act of 1964, Title IX of the Educational Amendments of 1972 and Section 504 of the Rehabilitation Act of 1973 or other federal, state, or local laws or executive orders.

In addition, Carnegie Mellon University does not discriminate in admission, employment or administration of its programs on the basis of religion, creed, ancestry, belief, age, veteran status, sexual orientation or in violation of federal, state, or local laws or executive orders. However, in the judgment of the Carnegie Mellon Human Relations Commission, the Department of Defense policy of "Don't ask, don't tell, don't pursue" excludes openly gay, lesbian and bisexual students from receiving ROTC scholarships or serving in the military. Nevertheless, all ROTC classes at Carnegie Mellon University are available to all students.

Inquiries concerning application of these statements should be directed to the Provost, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213, telephone (412) 268-6684 or the Vice President for Enrollment, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213, telephone (412) 268-2056.

Obtain general information about Carnegie Mellon University by calling (412) 268-2000.



Carnegie Mellon
Software Engineering Institute

Pittsburgh, PA 15213-3890

An Approach for Selecting and Specifying Tools for Information Survivability

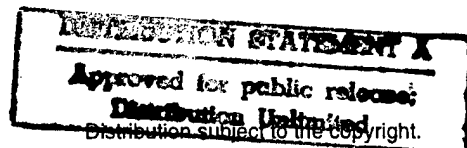
CMU/SEI-97-TR-009
ESC-TR-97-009

Robert Firth
Barbara Fraser
Suresh Konda
Derek Simmel

19980810 154

July 1998

Survivable Systems Initiative



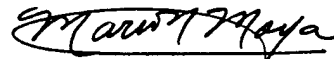
DMO QUALITY INSPECTED 1

This report was prepared for the

SEI Joint Program Office
HQ ESC/DIB
5 Eglin Street
Hanscom AFB, MA 01731-2116

The ideas and findings in this report should not be construed as an official DoD position. It is published in the interest of scientific and technical information exchange.

FOR THE COMMANDER



Mario Moya, Maj, USAF
SEI Joint Program Office

This work is sponsored by the U.S. Department of Defense.

Copyright 1998 by Carnegie Mellon University.

NO WARRANTY

THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

Use of any trademarks in this report is not intended in any way to infringe on the rights of the trademark holder.

Internal use. Permission to reproduce this document and to prepare derivative works from this document for internal use is granted, provided the copyright and "No Warranty" statements are included with all reproductions and derivative works.

External use. Requests for permission to reproduce this document or prepare derivative works of this document for external and commercial use should be addressed to the SEI Licensing Agent.

This work was created in the performance of Federal Government Contract Number F19628-95-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. The Government of the United States has a royalty-free government-purpose license to use, duplicate, or disclose the work, in whole or in part and in any manner, and to have or permit others to do so, for government purposes pursuant to the copyright license under the clause at 52.227-7013.

This document is available through Asset Source for Software Engineering Technology (ASSET): 1350 Earl L. Core Road; PO Box 3305; Morgantown, West Virginia 26505 / Phone: (304) 284-9000 or toll-free in the U.S. 1-800-547-8306 / FAX: (304) 284-9001 World Wide Web: <http://www.asset.com> / e-mail: sei@asset.com

Copies of this document are available through the National Technical Information Service (NTIS). For information on ordering, please contact NTIS directly: National Technical Information Service, U.S. Department of Commerce, Springfield, VA 22161. Phone: (703) 487-4600.

This document is also available through the Defense Technical Information Center (DTIC). DTIC provides access to and transfer of scientific and technical information for DoD personnel, DoD contractors and potential contractors, and other U.S. Government agency personnel and their contractors. To obtain a copy, please contact DTIC directly: Defense Technical Information Center / Attn: BRR / 8725 John J. Kingman Road / Suite 0944 / Ft. Belvoir, VA 22060-6218 / Phone: (703) 767-8274 or toll-free in the U.S.: 1-800 225-3842.

Table of Contents

1	Introduction	1
2	Tool Characterization and Selection Methods	3
2.1	Summary	5
3	Security Tool Identification Approach	7
3.1	Security Functionalities	7
3.2	Tool Identification Procedure	11
3.3	Summary	13
4	Identifying Functionalities to Counter Common Attack Methods	15
4.1	Denial-of-Service	16
4.1.1	Prevention	16
4.1.2	Detection	17
4.1.3	Response	17
4.2	IP Source Address Spoofing	18
4.2.1	Prevention	18
4.2.2	Detection	18
4.2.3	Response	19
4.3	Summary	19
5	Conclusions	21
	References	23

List of Figures

Figure 4-1:	A Simplified Attack Scenario	15
Figure 4-2:	Network Monitors Installed for Inspection and Logging of Network Events	17

List of Tables

Table 3-1:	Security Functionality Lexicon
------------	--------------------------------

8

An Approach for Selecting and Specifying Tools for Information Survivability

Abstract: As today's technology becomes increasingly complex to manage, administrators of survivable systems will need to place increased reliance on tools to assist them. The selection and specification of these tools must be conducted in a reliable, systematic fashion. This paper proposes a lexicon of functionalities to characterize survivable systems activities, and an approach to analyze networked systems environments. Application of this analysis approach will assist organizations in establishing criteria for selecting tools, and to identifying requirements for new tool development to accommodate needs not met by currently available tools.

1 Introduction

Today's technology base is becoming increasingly large and complex. Networks are growing, and applications are being migrated from centralized systems to client-server environments. In addition, organizations are connecting their networks to those of other organizations and to the Internet at a rapid rate [Network 96]. All of this added complexity presents a challenge to administrators who are responsible for managing these systems. The growth in the number of networked systems has accelerated demand for qualified administrators, and the increasing complexity of networked systems has raised the threshold of expertise required of these administrators. At the time of the birth of the Internet, systems administrators were typically well-versed and experienced in the technology they were charged to manage. There is ample evidence that the average level of expertise demonstrated by the fast-growing number of systems administrators today is considerably lower, and insufficient to allow them to configure and manage their complex systems in a survivable manner. This lack of adequate expertise is seen daily at the CERT[®] Coordination Center.¹ The continuing growth in the number of courses, seminars, and conferences directed at managing technology in the Internet environment reflects heavy demand for training and development in network systems administration skills. At the same time, a quick look at popular press publications reveals that many organizations now provide business products and services over the Internet, and that they are becoming increasingly concerned about the security and reliability of their technology. All these observations illustrate a dangerous gap between the need to secure systems and the supply of individuals capable of implementing adequately secure information technology environments.

¹ CERT is registered in the U.S. Patent and Trademark Office.

The CERT Coordination Center (CERT/CC) is located at the Software Engineering Institute. Sponsored in part by the United States Department of Defense, the CERT/CC is chartered to work with the Internet community in detecting and resolving computer security incidents, and taking steps to prevent future incidents.

One way to bridge this gap is to provide hardware and software tools (hereafter simply referred to as tools) to assist administrators in their efforts to provide adequate security. Indeed, there are a number of tools available today to help manage networked systems, aid in protecting systems, monitor network activities, and respond to security events. However, these tools are not documented using consistent terminology, and there is no standard way to evaluate and select tools. It is often left up to individual administrators to sort through them and select, based on their own expertise, tools that may be appropriate for their specific environment.

The purpose of this paper is to provide an approach to support decision making about which tools are appropriate for the survivability needs of a given networked systems environment. On a broader level, we hope that software developers will implement tools targeted at providing the *functionalities* described in this paper, and that commercial vendors will market their products using the vocabulary of functionalities in our lexicon. We believe that this work can serve as a foundation for further developing and formalizing the description, classification, analysis, and selection of tools to support system survivability.

In section 2, we briefly review some methods currently used to characterize and select among existing security tools. Existing techniques tend to be informal and often do not provide much insight into the applicability of tools to meet specific security objectives. In section 3, we present a new lexicon of security functionalities. We then use these functionalities in a procedure to link security policies to tools. The special vocabulary only applies to functionalities that are useful when considering the security and survivability of information resources, and does not cover functionalities that have no direct security bearing. Section 4 examines two common security problems to illustrate how the functionalities may be used to identify tools for addressing specific attack methods. Finally, in section 5, we review our approach, discuss issues raised during its development, and suggest areas for future development.

2 Tool Characterization and Selection Methods

Several methods have been proposed in the literature to characterize and guide selection of security tools. Popular texts on Internet security topics often discuss specific uses of well-known security tools, and include appendices listing tools grouped by similar areas of use. Web sites of commercial software companies portray tools in a variety of ways. Some tool characterizations are evident in the ways archival sites on the Internet have chosen to arrange their collections. In this section, we will briefly discuss a few examples of such sources examined during development of our approach.

In the NIST special publication, *Guide for Selecting Automated Risk Analysis Tools* [Gilbert 89], Gilbert lists *data collection*, *analysis*, and *output of results* as the three modules that should be present in any automated risk analysis tool. This represents a purpose-specific (in this case, for risk analysis) description of the input-processing-output model by which any tool may be characterized. Gilbert also describes site-specific selection criteria. Among these she includes hardware and software compatibility, methodology (whether the tool performs a quantitative or qualitative analysis), reporting requirements, documentation, history and security features, utility and ease of use, training and technical support, and cost. These criteria represent qualitative measures by which selection can be made between otherwise apparently acceptable tools.

Missing from a selection process based on such criteria are answers to two more fundamental questions: (1) "Is this the right tool for the problem" and (2) "Is the problem the correct one to be solved?" With respect to selection of tools for survivable systems, our approach begins to address these more fundamental issues by providing a common vocabulary of *functionalities*. These functionalities may be used to characterize solutions to security objectives, and then to identify sets of tools that serve to meet the objectives. Once the objectives have been met, then selection among the sets identified can be made based on parameters such as cost, efficiency, flexibility, and so forth.

In *A Guide to the Selection of Anti-Virus Tools and Techniques* [Polk 92], Polk and Bassham use the titles *detection*, *identification*, and *removal* to name three primary classes of anti-virus products. Each of these classes is supported by a variety of techniques and categories of tools. The techniques described include signature scanning, algorithmic detection, inoculation, heuristic binary analysis and precise identification. Tool categories include general purpose monitors, access control shells, checksums for change detection, and knowledge-based removal tools. If we break such techniques and tool categories down into their component operations, we begin to see some basic functionalities, such as scanning, monitoring, access-controlling, integrity-checking, inspecting, and eradicating.

Texts on network and information security often include appendices in which the authors place categorized lists of security tools. Their choices of categories suggest functionalities which tools in a category have in common. For example, in Appendix A of *Firewalls and Internet Security* [Cheswick 94], Cheswick and Bellovin provide an annotated list of software available via

the Internet. The list is clustered in four categories: software useful for building firewalls, software useful for network management and monitoring, auditing packages, and cryptographic software. The first category comprises tools that implement a group of functionalities that together serve to build a specific type of protective security, namely firewalls. The second category covers functionalities such as probing, scanning, and monitoring, which represent information gathering and detection activities. As its name implies, packages in the third category serve auditing functionalities. The cryptographic software category lists resources for implementing encrypting¹ functionalities. Chapman and Zwicky categorize many of the same tools in Appendix B of *Building Internet Firewalls* [Chapman 95]. They group tools in the following categories: authentication, analysis, packet filtering, proxy systems, daemons, and utilities. The first four of these may easily be described by the functionalities they serve: authenticating, auditing or integrity-checking, filtering, and proxying. The daemon category lists security-enhanced tools that operate in place of vendor-supplied programs. In this way they provide a *substituting* functionality by providing the same services, but in a more secure manner. The utilities category lists a number of other tools that serve a variety of functionalities. The different categorization methods used in these two books further illustrates the lack of uniformity in characterizations of common security tools in the literature of the field. Several other texts simply list tools alphabetically by name, which may be convenient for looking up information about a specific tool, but does not provide insight regarding functionalities provided by tools.

To explore security tool characterizations and categories currently used within the vendor community, we surveyed a broad range of web sites on the Internet. Among web sites of commercial software producers [e.g., Sun 96, Microsoft 96, TIS 96a, DEC 96, Cisco 96b], almost all vendors list their products alphabetically by name within market categories they define for themselves. Some provide a list of their products ordered by release date. A few, e.g., [Cisco 96a] highlight products oriented specifically toward security purposes. At one web site we visited, the term "security technology" is used in a way that suggests that the listed items are stand-alone software modules when they are in fact different capabilities implemented within a single operating system product. Functionalities provided by commercial products are rarely evident without careful examination of associated documentation, which is often not available on-line. We believe that web sites such as these could be made considerably more useful if products could be listed (and identified in search engines) by the security functionalities they provide.

Popular archives of security tools available via the Internet categorize tools in a variety of ways. For example, the COAST archives at Purdue University [COAST 96a,b] provides indexes of security information and tools by category (sources of tools and papers) and by author. Within the category index is a section where tools are separated by operating system platform. Selecting a platform initiates automatic generation of an annotated, alphabetical list of tools

¹. When we refer to *encrypting* functionalities, we assume that corresponding *decrypting* functionalities exist as well.

for that platform. Functionalities provided by the tools, however, are not evident without further investigation of each tool. The tool archives maintained by the U.S. Department of Energy's Computer Incident Advisory Capability [CIAC 96] also separates tools initially by OS platform. Within each platform, however, tools are grouped into categories that focus on the use of the tool. For example, under the category of tools for use on UNIX systems, groups include authentication, cryptographic checksums, firewalls, network monitoring, network security, system monitoring, and general tools. These groups represent different sets of functionalities, and to some extent, different scopes of similar functionalities (e.g., network vs. system monitoring).

2.1 Summary

Existing methods for selecting tools do not sufficiently examine the issue of whether or not the correct tool is being selected or if the right problem is being solved. Current classifications of tools are diverse, and provide varying degrees of insight into the specific functionalities served by each tool within a given group. This makes it difficult to identify and compare tools to suit the needs of a particular security objective in a specific environment. Nevertheless, we can determine a set of basic functionalities by drawing upon elements from existing methods, characterizations and categorizations, and examining them in light of established security principles and strategies. As we will see in the following section, these functionalities represent the building blocks used to describe solutions to specific security objectives.

3 Security Tool Identification Approach

Before appropriate security tools can be identified and selected, an organization must conduct a comprehensive examination of its security needs. The organization must know, and have documented, all of its current and anticipated information assets, and the infrastructure in which these assets are stored and communicated. Policies must be clearly defined regarding expectations and responsibilities for physical, personnel, and networked systems security, and the relationships between them. The organization must define expectations about how information is communicated between internal and external entities. As the needs and activities of an organization change with time, it is imperative that knowledge about the organization's information assets, infrastructure, personnel, and policies be kept up-to-date and consistent with one another. The organization must also maintain current knowledge about the kinds of security problems to which their information assets, infrastructure and personnel may be susceptible. Once these preparations have been completed, and procedures put in place to maintain the currency and accuracy of policies and knowledge of the organization, then specific security objectives may be defined for each information asset and service.

To establish a common vocabulary between security objectives and the specification of tools selected to meet those objectives, we have generated a lexicon of *security functionalities*. These functionalities represent primitive actions that support information survivability. We intend the lexicon to be small, yet comprehensive in its coverage of the primary security activities that any organization may need to employ over the lifespan of its information technology infrastructure. The functionalities span a variety of purposes within efforts to protect information assets and services, detect security-related activities, and respond to security events.

The approach we propose uses these functionalities in a procedure to link organizational security requirements to tools. These security functionalities are used to map security objectives to tools and in this way provide a direct line from an organization's security policies to the tools that are required by administrators to support those policies. The approach may be used to systematically analyze an organization's information assets and services, or to analyze specific security problems involving particular information assets or services.

3.1 Security Functionalities

The functionalities described in Table 3-1 represent specific activities associated with common security objectives. They cover a broad range of purposes and were derived from a variety of sources. A number of functionalities are suggested by existing selection and characterization methods (see section 2). We also identified functionalities exhibited in currently available security tools. To survey terminology used by authors and vendors of existing security tools, we examined the installation, configuration, and user documentation provided with 120 commercial, shareware, and freeware products. The sample of tools included those for use with Microsoft operating systems (MS-DOS, Windows 3.x/95/NT), Apple MacOS, and a variety of UNIX-based systems. The tools surveyed represent a broad range of solutions, from password-prompting screen-locks to network monitoring agents and robust firewall construction

packages. From the myriad of features, purposes and implementations described in the documentation, we derived a number of common functionalities served among clusters of these tools. In addition, hypothesized security needs and solutions for security issues not yet implemented in available tools were considered. To complete the lexicon, we analyzed a number of current attack methods and extracted functionalities that supported the prevention, detection, or response to the attack methods.

Some functionalities are complementary to one another, some are defined in terms of others, and the remaining functionalities may be considered independently. Tools can exist to provide a single functionality, part of a functionality, or to encompass several functionalities. To prevent bias in establishing contexts for their use, we have deliberately avoided grouping the functionalities into categories. Functionalities that appear to have similarities between them have been collocated in the table to facilitate comparison between their definitions.

The definitions assume no particular implementation, context, or scope of deployment. For example, an organization may determine a need to *hide* an entire subnet, a specific host on a network segment, parts of a storage volume, or specific fields within a data structure in active memory. It may need several such objects hidden in completely different areas of the network environment. Alternatively, all these objects may reside in one area of the network environment, but the users or processes from which the objects are to be hidden may be different.

Functionality	Activity	Security Purpose
Hiding	Placing a data resource where it cannot be discovered	to prevent unauthorized access to that data resource.
Encrypting	Translating data in its original form into an unintelligible (encrypted) form	to protect the data from being read by unauthorized users or processes.
Decrypting	Translating data in encrypted form back into its original form	to allow authorized users or processes to read the data.
Locking	Making a data resource accessible only to the lock holder until it is unlocked	to prevent modification of the data resource by others while it is locked.
Limiting	Setting an upper bound on system resources that may be consumed by an agent or process	to maintain availability of those system resources for other agents and processes.
Reserving	Setting a lower bound on system resources that will be available to an agent or process	to provide a minimum guaranteed quantity of system resources to that agent or service.

Table 3-1: Security Functionality Lexicon

Functionality	Activity	Security Purpose
Filtering	Examining a data stream and removing from it items that are deemed undesirable or inappropriate	to protect downstream resources or processes against the undesirable or inappropriate items and their effects.
Shielding	Creating a barrier	to keep undesirable activity away from specified data resources or processes.
Containing	Confining a data resource or process	(1) to prevent transmission of the data resource or process out of the confinement area. (2) to protect external resources or agents from undesirable effects of the data resource or process.
Authenticating	Proving that an agent is who or what the agent claims to be	to establish confidence in the identity of the agent, often as a prerequisite to subsequent access.
Access Controlling	Granting access to data resources only to authorized agents or processes	to prevent access by unauthorized agents and processes.
Enforcing	Controlling the sequence, direction or route of access or processes	to ensure that agents and processes operate as required by security policy.
Tunneling	Transmitting data formatted for one protocol in a container of a second protocol	to convey the data in a more secure manner by employing the second protocol.
Obliterating	Disposing of data in a manner that assures that it can never be restored from the medium on which it was stored	to ensure that the medium can be reused or disposed of without risk of disclosing the data it previously contained.
Eradicating	Obliterating every instance of an item	to ensure that no further access is possible to any instance of that item.
Replicating	Providing multiple copies of a data resource	to maintain availability of that resource.

Table 3-1: Security Functionality Lexicon

Functionality	Activity	Security Purpose
Mirroring	Replicating, in real time, an identical copy of a data resource	(1) to maintain real-time availability of that resource. (2) to protect the original copy of the data resource.
Preserving	Storing in a secure place, for an indefinite period, a data resource	in order that the data resource may be restored when needed.
Restoring	Returning a resource to previous state known to be correct	to recover from corruption or tampering of that resource.
Probing	Attempting connections or queries	to gain knowledge about a specified system.
Scanning	Iteratively probing a collection of systems or data	to identify those which respond to probes.
Monitoring	Observing a data stream for specified events	to provide data for subsequent action or analysis.
Logging	Systematically recording specified events in the order that they occur	to provide a data trail for subsequent analysis.
Inspecting	Examining a data resource or process	to identify anomalous content or behavior in the data resource or process.
Auditing	Systematically examining system data against documented expectations of form or behavior	to verify conformance with documented expectations.
Integrity Checking	Verifying that the contents of a data resource are exactly as created, stored, or transmitted	to detect modification of the data resource.
Notifying	Alerting a designated recipient to the occurrence of a specific event	to prompt the recipient to take appropriate action.
Reporting	Processing selected data to produce output that has meaningful form	to provide the reader information for subsequent analysis and decision support.
Patching	Modifying code or data	to correct known errors or vulnerabilities in a data resource or program.

Table 3-1: Security Functionality Lexicon

Functionality	Activity	Security Purpose
Substituting	Operating in place of some original resource or service	to provide the same resource or service in a more secure manner.
Proxying	Operating as an agent on behalf of a resource or service	to provide a more secure interface to that resource or service.
Inoculating (Vaccinating)	Installing protective measures	to prevent subsequent vulnerability to known attacks.
Retreating	Restricting access to, or disabling, resources and services in response to specific events	to eliminate exposure to undesirable activity that has been detected.

Table 3-1: Security Functionality Lexicon

Although the functionalities described above identify activities with specific security purposes, they may, depending on deployment scope and purpose, be interdependent and may themselves have security implications. For example, a replicating functionality may induce a requirement to protect all of the copies generated of a data resource. If security objectives are left incompletely covered, intruders may be able to circumvent even the most comprehensive of solutions implemented elsewhere in the network environment. It is therefore imperative that the identification of functionalities associated with a security objective be as precise and comprehensive as possible.

3.2 Tool Identification Procedure

Identifying that a functionality is required will arise from the organization's knowledge of its security goals, security problems it expects to defend against, activities it will observe and record, and actions it intends to take in response to security events. *Where* the functionality is required will be determined as the organization systematically examines all of its information assets and services, and the context of their implementation.

As was mentioned earlier in this chapter, the organization must identify its information assets and services, and their corresponding security requirements. Once the security objectives for each information asset and service have been defined by the organization, tools to support the objectives can be identified using the following procedure:

For each information asset or service X, for which tools are to be identified, do the following:

1. Given the security objectives for X, select functionalities from Table 3-1 that support the objectives for that asset or service.
2. Define the implementation context for X. This will include the system and network environment in which X resides or is used.

3. Identify and characterize tools in terms of the set of functionalities (from Table 3-1) they support and the implementation context in which they operate.
4. Identify candidate tools that support functionalities identified in step 1, and that operate within the implementation context defined in step 2.
5. Select those tools that together provide all the functionalities required by X.

This procedure can be applied to a single information asset or service, or it can be comprehensively applied to all information assets and services. When it is applied to all information assets and services, the set of functionalities identified will be the total set required by the organization as a whole. Since most organizations already have significant installed technology bases, this procedure will often be used to analyze existing information assets and services. However, this procedure is also useful for planning the addition of new information assets and services.

The set of functionalities identified in step 1 will often be those that support the well known requirements of confidentiality, integrity, and availability but may also include those that support the maintenance of those requirements (e.g., monitoring, logging, auditing, etc.). The richness of this set may depend on the comprehensiveness of the organization's policies, and an alert administrator may identify either a weakness or absence of required policies as he scans the table of functionalities and selects those that match stated objectives.

Step 2 is necessary because it provides operational constraints that must be considered when selecting tools. A tool that provides a specific set of functionalities is of no use if the operational context is different from that of the information asset or service for which the tool is being considered.

Upon completing steps 1 and 2 of the procedure, an organization will have documented a collection of required functionalities and a description of the implementation context for each information asset or service for which it is selecting tools. The next task is to map that set of functionalities to one or more tools.

In order to select tools to support functionalities, the tools must be identified and then described in terms of the functionalities they support as well as the context in which they operate (step 3). At present, such standard descriptions don't exist. Until such standard descriptions do exist, it will be the job of the administrator to evaluate tools in terms of the functionalities they support. For example, Tripwire [Kim 93] could be described as providing auditing, integrity checking, and reporting. To define the set of functionalities supported by a specific tool, the administrator will need to review the accompanying documentation *and possibly* install and test the tool (since documentation is often incomplete). Once the tools are described in terms found in Table 3-1 along with their operating context, selecting candidate tools is a straightforward task.

The set of candidate tools (step 4) is comprised of all tools that provide at least one of the required functionalities, and that operate within the same implementation context as the information asset or service. From this set, one or more tools are then selected that, together, will

satisfy all of the functionalities required (step 5). In some cases, a single tool per functionality may suffice. In general, however, it is likely that a variety of permutations of tools may be identified, in which each tool cooperates with others to provide the necessary coverage of functionalities in the required areas. Each permutation that provides the required coverage represents a set of tools that together define a solution. If no arrangement of available tools will provide the required coverage to implement the functionalities, further analysis will be needed to determine how to address the gap. For some organizations, customized solutions may need to be engineered. For other organizations this may mean changing the underlying technology (implementation context) supporting the information asset or service in order to take advantage of existing tools. In the worst case, the organization may have to accept the risks of not providing the missing functionalities.

Selection among multiple solution sets may be made based on criteria such as efficiency, flexibility, ease-of-use, cost, and the availability of technical support.

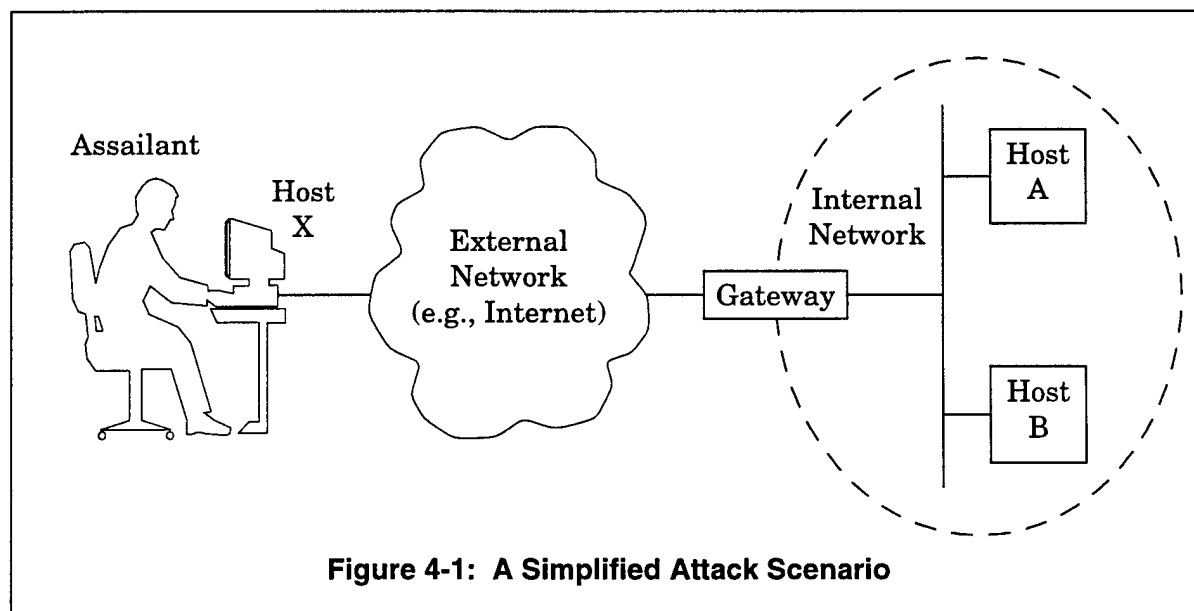
3.3 Summary

In this chapter we have proposed a set of descriptive terms, *functionalities*, and a method for using them to identify tools needed by administrators to support organizational security policies. In this way, we have demonstrated a method to translate security policies to supporting tools. The proposed procedure highlights the need for organizations to comprehensively develop security policies and for tools to be characterized in a uniform, consistent manner. This procedure can support the evaluation of competing solution sets based on functionality of existing tools as determined by the richness of the security functionalities they support. In addition, the analysis performed when mapping organizational security objectives to these functionalities and then to tools may indicate opportunities to improve the environment's existing configuration for long-term survivability, robustness, and manageability.

4 Identifying Functionalities to Counter Common Attack Methods

As a consequence of the technologies and protocols employed in implementation contexts, there exist context-specific vulnerabilities that may require the implementation of additional security functionalities. Intruders often exploit newly-discovered vulnerabilities in an attempt to circumvent existing security measures that organizations have implemented. It is therefore imperative that organizations keep up-to-date with security information relevant to their information infrastructure. As new vulnerabilities are discovered, vendors and incident-response organizations produce reports¹ detailing workarounds and patches to address such vulnerabilities.

To illustrate how an organization may identify additional functionalities to address specific network security problems, we will explore two well known attack methods. Security functionalities that serve to prevent, detect and respond to these problems will be identified, along with contexts in which such functionalities may appear in a typical network environment. For simplicity, we will assume that assailants attempting to disrupt or gain access to an organization's internal network are doing so from some external site. For organizations in which there is concern of attack from within the organization's boundaries, the internal network should be interpreted as the part of the organization's infrastructure (a single department or subdivision) to be protected. It is assumed that precautions implemented to protect against assailants may be placed between the assailant and the target systems.



1. Examples of such reports include CERT advisories and vendor-initiated bulletins.

4.1 Denial-of-Service

As its name suggests, this form of attack is aimed at preventing a targeted system from providing, receiving or responding to network services. Current attacks typically operate by flooding one or more services on the target with connection requests or queries. The services on the target become overloaded, disabling their ability to respond to legitimate service requests, and in some cases, the target system is forced to shut down completely. Alternatively, an assailant may take advantage of a flaw in the implementation of a service to disrupt or disable that service on the target host.

4.1.1 Prevention

Unfortunately, attacks of this sort are difficult to prevent, in part because the purpose of providing services in the first place is to allow legitimate clients to make use of them. Verifying that requests for service are from legitimate clients requires a strong *authenticating* process for each request. This authentication should not rely solely on network data such as host IP addresses or information provided by domain name services, since such information may be spoofed by an assailant, and is therefore unreliable. One solution is to employ cryptographic authentication protocols [Kaufman 95, p.184], which incorporate an *encrypting* functionality to protect the confidentiality and integrity of authentication dialogues.

Flooding attacks are successful because more service requests can be sent to a service host than it can process within a short period of time. Hence a *limiting* functionality is necessary to simply refuse or ignore any more requests than services can handle within a given period. Similarly, the service hosts should be protected against being forced to shut down by *reserving* a minimum of system resources necessary to keep them up and running.

To minimize the effects of denial-of-service attacks, one should configure each host to offer and respond to as few services as possible. All unused and unnecessary services (e.g., echo, chargen, finger, tftp, uucp) should be disabled and removed. If possible, different services should be implemented on separate hosts, in order to reduce the susceptibility of a service to an attack launched upon another service operating on the same host, i.e. *containing* the effects of a denial-of-service attack against one service to protect other services. In addition, one should *shield* services of a host by installing *proxying* agents for each of the services. Such agents may be designed to *inspect* and *authenticate* requests for service, *filter* out those that are malformed or fail authentication, and then pass only the qualified requests to the appropriate service host. If different proxying agents are installed to serve different constituencies of clients, then a denial-of-service attack reaching one proxying agent will not affect service to clients from the other constituencies.

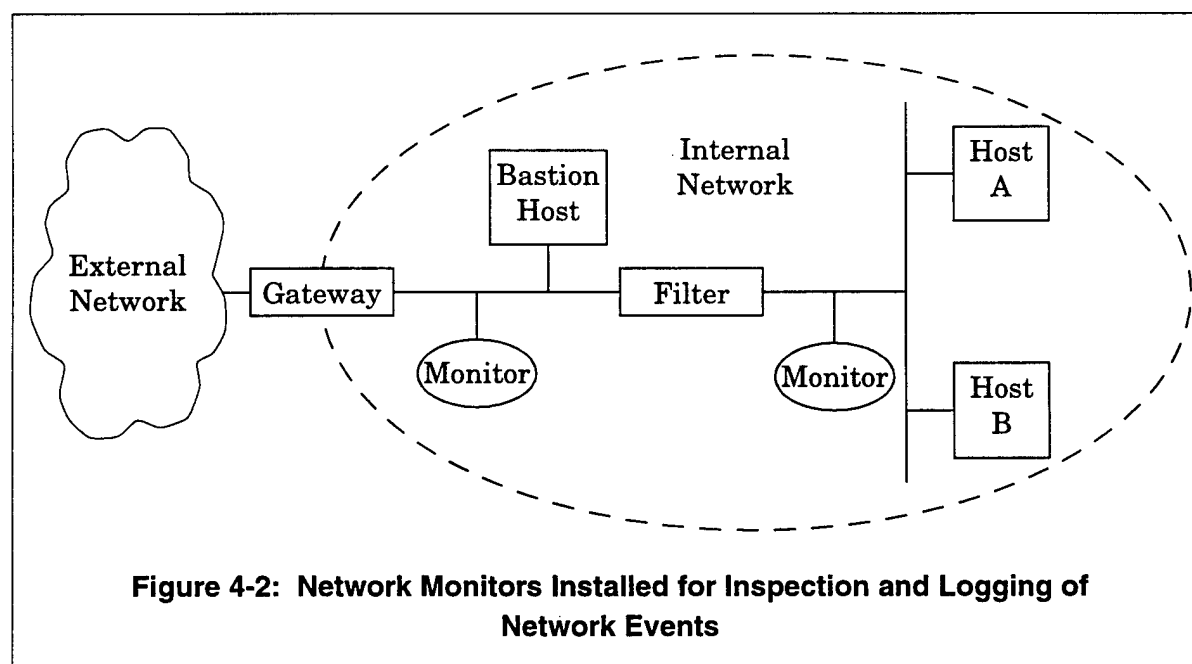
Several tools exist to permit greater security, control and *auditing* of services. For UNIX systems, tools such as Wietse Venema's *TCP wrappers* [Venema 96a] package and the *netac* utility in Trusted Information System's Firewall Toolkit [TIS 96b] may be used to pre- and post-

process transactions with services to *inspect*, *authenticate*, *filter*, *monitor*, *log*, and *report* service activities.

An additional strategy for minimizing the effects of denial-of-service is to maintain a distributed capacity for services. This is achieved by *replicating* a service across several independent systems so that an attack against one system does not wipe out all availability of that service.

4.1.2 Detection

In order to be able to respond to a denial-of-service attack, one must be able to observe and react to suspicious traffic on the network. Network *monitoring* capabilities with secured output channels (to avoid tampering of the output) should be installed to permit *logging* and *notification* about significant network traffic and events. Such monitoring not only serves to allow detection of anomalies, but also facilitates *auditing* of network and host configurations. As depicted in Figure 4-2, network monitors should be placed strategically so that their output may be compared to verify configuration and operational assumptions.



4.1.3 Response

Given a sufficiently robust environment, one may also choose to implement a dynamic, *retreating* facility, by which systems will automatically react to bursts of service requests, or to other suspicious activity, by temporarily disabling affected areas until the burst or other activity subsides. This strategy, however, only serves to protect the target systems themselves. By forcing a system to react by disabling itself, the denial-of-service is achieved, and therefore the attack succeeds. Nevertheless, it is generally a good practice to implement self-defensive mecha-

nisms on systems, especially if such systems are expected to operate in an unattended manner.

4.2 IP Source Address Spoofing

This attack method attempts to gain access to a target host by assuming the identity of a host trusted by the target. The assailant will often inhibit traffic from the genuine trusted host by means of a denial-of-service attack against it, and then attempt to gain access to the target host using the trusted host's identity. In the case of hosts on a TCP/IP network, the identity assumed by the assailant is the trusted host's IP address.

4.2.1 Prevention

Prevention of source address spoofing attacks, as with denial-of-service, comes down to having some way of reliably *authenticating* the hosts making connection attempts. Again, methods for implementing such authentication should not rely solely on information from domain name services or network addresses, which spoofing attacks abuse to masquerade as trusted hosts. Reusable passwords must be transmitted in strongly *encrypted* form. Connections should be periodically reauthenticated during each session, to guard against hijacking due to a compromised session. For added security once a session has been established, all communication between hosts should also be strongly encrypted. An example of a tool which implements such cryptographic authentication and transmission protocols is *Secure Shell* (SSH) [Ylonen 96a,b].

In addition to reliable authentication, it is helpful to *shield* trusted hosts, to whatever extent is possible, against denial-of-service attacks. For organizations that do not require Mobile-IP accessibility across their firewalls, packet *filtering* precautions (see [CA-96.21]) may be implemented at an organization's firewalls to keep out connections from external sources that attempt to masquerade as trusted internal hosts. Similarly, to prevent source address spoofing attacks originating from within an organization's internal network, one must *contain* outgoing packets that have source addresses not belonging to genuine internal hosts. This can be achieved by filtering outgoing packets at the organization's firewalls to ensure that their source addresses are only those from internal hosts. For TCP/IP networks, addresses reserved for private intranet use [Rekhter 96], and loopback addresses in the 127.x.x.x range should also be filtered (incoming and outgoing) at the organization's firewalls, since packets with these source addresses should neither be arriving from external hosts nor leaving the internal network.

4.2.2 Detection

Since IP spoofing attacks are typically preceded by a denial-of-service attack, the *monitoring* functionalities described earlier also apply in this case. In a network environment where the packet filtering precautions described above have been applied, characteristics of network packets on a given segment of the network are predictable by virtue of their source and des-

termination addresses. Monitoring functionalities implemented for each pair of gateways between network segments may be coordinated to *log* suspicious packets, and activate a *notifying* functionality to alert the appropriate response processes and personnel in the organization.

Another way to identify IP spoofing activity is to *audit* the process accounting logs of hosts within the organization's network to verify that connections made to each host are matched by connection attempts from the corresponding hosts. If the process accounting logs show a connection made without a corresponding connection attempt from the alleged source host, then the connection may have been achieved via IP spoofing. This is often the only means to track IP spoofing activity that occurs within the confines of an internal network, since the host from which the spoofing attack was initiated is behind the organization's firewall.

4.2.3 Response

If an IP spoofing attack is detected by a network monitoring and notification mechanism, one could trigger a *retreating* functionality to isolate affected systems by shutting down their external connections. Alternatively, the retreating functionality might be implemented upstream at the network filters themselves, with a dynamic filter adjustment capability to block only the suspicious traffic while allowing other network activity to continue. A dynamic capability such as this would minimize broad-scale disruption of service due to a single suspicious event.

If systems are discovered to have been compromised, *integrity-checking* functionalities will need to be initiated using untainted media to determine the extent of damage. *Restoring* functionalities may then be used to return affected systems to a state known to be correct.

4.3 Summary

The existence of context-specific vulnerabilities requires implementation of additional security functionalities. Despite existing security measures that organizations have implemented, intruders may be able to disrupt or gain access to systems and services by exploiting newly-discovered vulnerabilities. To illustrate how an organization may identify additional functionalities to address specific network security problems, functionalities for preventing, detecting, and responding to denial-of-service and IP spoofing attacks were presented. The increasing sophistication of intruder attack methods, as observed by the CERT/CC's incident handling and vulnerability analysis experts, underscores the importance of maintaining an organization's knowledge of potential security problems to its information infrastructure, and its ability to identify and implement additional security functionalities as needed.

5 Conclusions

In this paper we have proposed a standard lexicon to describe the basic activities that are needed to address the survivability needs of information assets and services. These *functionalities* can be used to map the security requirements of assets and services to specific tools. The functionalities listed in Table 3-1 cover a broad range of purposes and represent a starting point for the further exploration of this area. As people use these terms in the specification of their tools, they may discover gaps, ambiguities or redundancies in the vocabulary. We encourage such exploration and welcome the enrichment of the vocabulary.

The selection of tools based on functionality requirements is currently a time-consuming task primarily because existing tool developers do not use standard terminology to characterize their products. If tools makers in the future will adopt the proposed vocabulary and use it to describe the nature of their products, the evaluation of tools will be made much simpler since the mapping process will be more direct. To provide immediate assistance, we encourage current product vendors and interested parties to describe existing tools in terms of the vocabulary. Not only will this be a test of the vocabulary itself, it will also demonstrate functionalities for which there are few or no tools available. These gaps would represent opportunities for future development efforts.

As tools are characterized using the standard vocabulary of functionalities, we encourage the recording of the procedures used to accomplish the characterizations. In this way, the knowledge of how to characterize the tools can be shared and the activity can become more widespread.

The table of functionalities currently does not include arguments to the terms. For example, the term *filtering* is included but not *what* to filter or *where* to filter. Future research efforts are needed to fully qualify the gerunds by providing the context for their use. However, such qualification based on today's technology might be restricting and would surely need to be revised regularly as newer technologies emerge.

In order to further develop both the lexicon and the tool selection procedure, we recommend that a series of case studies of increasing complexity be initiated. The results of such case studies would support the following:

- a. improvement of the vocabulary
- b. refinement of the procedure
- c. determination of which steps in the procedure would be candidates for automation

The current procedure, if applied to a number of information assets and services simultaneously, may result in a number of solution sets. The task of choosing the optimum solution from this set for a given security context could be difficult. Knowledge-based tools such as an expert system could aid in this challenging task.

Another avenue for future work involves the certification of tools for the functionalities they provide. Two aspects are important: 1) does the tool provide a given functionality, and 2) the

strength and comprehensiveness with which the functionality is implemented in the tool. A reasonable starting point might be a simple yes/no notation (a 1,0 scale) to indicate whether or not the functionality is present. Later, this could be expanded to a scale that would represent both the presence and the strength of the functionality.

Until our technology base is without flaw there will be an ongoing need for tools to support the survivability needs of our information assets and services. This paper represents a starting point in the quest to formalize the description, classification, analysis, and selection of tools to support that information survivability.

References

- [CA-96.21] CERT Coordination Center. *TCP SYN Flooding and IP Spoofing Attacks* (CA-96.21) [online]. Available FTP: <URL: ftp://info.cert.org/pub/cert_advisories/CA-96.21.tcp_syn_flooding>. Pittsburgh, Pa.: CERT/CC, Software Engineering Institute, Carnegie Mellon University (1996).
- [Chapman 95] Chapman, D. B. & Zwicky, E. D. *Building Internet Firewalls*. Sebastopol, Ca.: O'Reilly & Associates, Inc., 1995.
- [Cheswick 94] Cheswick, W. R. & Bellovin, S. M. *Firewalls and Internet Security: Repelling the Wily Hacker*. Reading, Ma.: Addison-Wesley Publishing, 1994.
- [CIAC 96] Computer Incident Advisory Capability. *Security Tools* [online]. Available WWW: <URL: <http://ciac.llnl.gov/ciac/SecurityTools.html>>. Washington, DC: United States Department of Energy (1996).
- [Cisco 96a] Cisco Systems, Inc. *Cisco Security Solutions* [online]. Available WWW: <URL: <http://www.cisco.com/warp/public/778/security.html>>. San Jose, Ca.: Cisco Systems, Inc. (1996).
- [Cisco 96b] Cisco Systems, Inc. *Products & Ordering* [online]. Available WWW: <URL: http://www.cisco.com/public/Product_root.shtml>. San Jose, Ca.: Cisco Systems, Inc. (1996).
- [COAST 96a] Computer Operations, Audit, and Security Technology Project. *Security Archive - Author Index* [online]. Available WWW: <URL: http://www.cs.purdue.edu/coast/archive/data/author_index.html> West Lafayette, In.: Department of Computer Science, Purdue University (1996).
- [COAST 96b] Computer Operations, Audit, and Security Technology Project. *Security Archive - Category Index* [online]. Available WWW: <URL: http://www.cs.purdue.edu/coast/archive/data/category_index.html> West Lafayette, In.: Department of Computer Science, Purdue University (1996).
- [DEC 96] Digital Equipment Corporation. *Products & Services* [online]. Available WWW: <URL: <http://www.digital.com/info/products.html>>. Maynard, Ma.: Digital Equipment Corporation (1996).
- [Firth 87] Firth, R.; Mosley, V.; Pethia, R.; Roberts, L.; & Wood, W. *A Guide to the Classification and Assessment of Software Engineering Tools* (CMU/SEI-87-TR-10, ADA213968). Pittsburgh, Pa.: Software Engineering Institute, Carnegie Mellon University, 1987.

- [Gilbert 89] Gilbert, I. E. *Guide for Selecting Automated Risk Analysis Tools* (SP 500-174). NIST Special Publication. Gaithersburg, Md.: National Institute of Standards and Technology, 1989.
- [Kaufman 95] Kaufman, C.; Perlman, R.; & Speciner, M. *Network Security: Private Communication in a Public World*. Englewood Cliffs, N.J.: PTR Prentice Hall, Inc., 1995.
- [Kim 93] Kim, G. H. & Spafford, E. H. *The Design and Implementation of Tripwire: A File System Integrity Checker* (CSD-TR-93-071). Purdue Technical Report. West Lafayette, In.: Department of Computer Science, Purdue University, 1993.
- [Microsoft 96] Microsoft Corporation. *Microsoft Product Catalog* [online]. Available WWW: <URL: <http://www.microsoft.com/products>>. Redmond, Wa.: Microsoft Corporation (1996).
- [NCC 87] National Computer Centre & Department of Trade and Industry (UK). *The STARTS Guide* (2nd Ed.). Manchester, England: National Computer Centre Publications, 1987.
- [Network 96] Network Wizards, Inc. *Internet Domain Survey, July 1996* [online]. Available FTP: <URL: <http://www.nw.com/zone/WWW/report.html>>. Menlo Park, Ca.: Network Wizards, Inc. (1996).
- [Polk 92] Polk, W. T. & Bassham, L. E. *A Guide to the Selection of Anti-Virus Tools and Techniques* (SP 800-5). NIST Special Publication. Gaithersburg, Md.: National Institute of Standards and Technology, 1992.
- [Rekhter 96] Rekhter, Y.; Moskowitz, B.; Karrenberg, D.; de Groot, G. J.; & Lear, E. *Address Allocation for Private Internets* (RFC 1918) [online]. Available WWW: <URL: <http://nitrous.digex.net/rfc1918.txt>>. Internet Engineering Task Force (1996).
- [Sun 96] Sun Microsystems, Inc. *Products and Solutions* [online]. Available WWW: <URL: <http://www.sun.com/products-n-solutions>>. Mountain View, Ca.: Sun Microsystems, Inc. (1996).
- [TIS 96a] Trusted Information Systems, Inc. *Products & Services* [online]. Available WWW: <URL: <http://www.tis.com/prodserv/index.html>>. Glenwood, Md.: Trusted Information Systems, Inc. (1996).
- [TIS 96b] Trusted Information Systems, Inc. *TIS Internet Firewall Toolkit* [online]. Available WWW: <URL: <http://www.tis.com/docs/products/fwtk/index.html>>. Glenwood, Md.: Trusted Information Systems, Inc. (1996).
- [Venema 96a] Venema, W. *TCP wrappers* (source code version 7.4) [online]. Available FTP: <URL: <ftp://ftp.win.tue.nl/pub/security/>>. Netherlands: Eindhoven University of Technology (1996).

- [Ylonen 96a] Ylonen, T. "SSH - Secure Login Connections over the Internet," 37-42. *Proceedings of the Sixth USENIX UNIX Security Symposium*. San Jose, Ca., July 22-25, 1996. Berkeley, Ca.: USENIX Association (1996).
- [Ylonen 96b] Ylonen, T. *Secure Shell* (source code version 1.2.17) [online]. Available FTP: <URL: <ftp://ftp.cs.hut.fi/pub/ssh/>>. Helsinki, Finland: Helsinki University of Technology (1996).

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (leave blank)	2. REPORT DATE July 1998	3. REPORT TYPE AND DATES COVERED Final
4. TITLE AND SUBTITLE An Approach for Selecting and Specifying Tools for Information Survivability	5. FUNDING NUMBERS C — F19628-95-C-0003	
6. AUTHOR(S) Robert Firth, Barbara Fraser, Suresh Konda, Derek Simmel		
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213	8. PERFORMING ORGANIZATION REPORT NUMBER CMU/SEI-97-TR-009	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) HQ ESC/AXS 5 Eglin Street Hanscom AFB, MA 01731-2116	10. SPONSORING/MONITORING AGENCY REPORT NUMBER ESC-TR-97-009	
11. SUPPLEMENTARY NOTES		
12.a DISTRIBUTION/AVAILABILITY STATEMENT Unclassified/Unlimited, DTIC, NTIS	12.b DISTRIBUTION CODE	
13. ABSTRACT (maximum 200 words) As today's technology becomes increasingly complex to manage, administrators of survivable systems will need to place increased reliance on tools to assist them. The selection and specification of these tools must be conducted in a reliable, systematic fashion. This paper proposes a lexicon of functionalities to characterize survivable systems activities, and an approach to analyze networked systems environments. Application of this analysis approach will assist organizations in establishing criteria for selecting tools, and to identifying requirements for new tool development to accommodate needs not met by currently available tools.		
14. SUBJECT TERMS security tools classification, selection, description		15. NUMBER OF PAGES 26 pp.
		16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED
20. LIMITATION OF ABSTRACT UL		