# How to Systematically Classify Computer Security Intrusions[†]

Ulf Lindqvist and Erland Jonsson
Department of Computer Engineering
Chalmers University of Technology
SE-412 96 Göteborg, Sweden
{ulfl, erland.jonsson}@ce.chalmers.se

## Abstract

*This paper presents a classification of intrusions with respect to technique as well as to result. The taxonomy is intended to be a step on the road to an established taxonomy of intrusions for use in incident reporting, statistics, warning bulletins, intrusion detection systems etc. Unlike previous schemes, it takes the viewpoint of the system owner and should therefore be suitable to a wider community than that of system developers and vendors only. It is based on data from a realistic intrusion experiment, a fact that supports the practical applicability of the scheme. The paper also discusses general aspects of classification, and introduces a concept called dimension. After having made a broad survey of previous work in the field, we decided to base our classification of intrusion techniques on a scheme proposed by Neumann and Parker in 1989 and to further refine relevant parts of their scheme. Our classification of intrusion results is derived from the traditional three aspects of computer security: confidentiality, availability and integrity.*

## 1. Introduction

> *The first step in wisdom is to know the things themselves; this notion consists in having a true idea of the objects; objects are distinguished and known by classifying them methodically and giving them appropriate names. Therefore, classification and name-giving will be the foundation of our science.*
> Carolus Linnæus, *Systema Naturæ*, 1735

The work presented in this paper emanates from intrusion experiments that we conducted [20]. The objective of the experiments was to find operational measures of computer security, that is measurements which reflect the dependence on and uncertainty of the operational environment in a probabilistic way, as opposed to static measures [25, 7] which reflect instead the quality of the system design. The need for a classification scheme arose when we were refining our modelling of the intrusion process [11].

Although several classification schemes focusing on different intrusion-related properties have been proposed, there is still no established taxonomy in general use. When trying to apply these schemes to our data, we found that they either focused on aspects other than those we were able to observe or that they were too superficial to be useful. We decided to develop a scheme that would fit our data, as well as be useful to others.

The motivations for a taxonomy and the objectives of the work are further explained in Section 2, while Section 3 is a note on the terminology used in this paper. The previous work in the field is presented in Section 4, the intrusion experiment is described in Section 5, and Section 6 describes our classification scheme. The advantages and limitations of the scheme are discussed in Section 7 and, finally, Section 8 concludes with a summary of the key points presented in this paper.

## 2. Rationale and objectives

Why would someone want to devise a taxonomy of intrusions? Is there a need for an established taxonomy? What tangible gain, other than the abstract aesthetic value of elegant expression and order, can justify the efforts required? Indeed, these are relevant questions, and we have found several answers.

- In general, categorizing a phenomenon makes systematic studies possible. In particular, a taxonomy of intrusions enables us to compile statistics on intrusions, observe patterns and draw other conclusions from collected intrusion data. We hope that this process will extend our knowledge of the phenomenon, and that it will be possible to strengthen systems against intrusions using this knowledge.

[†]Published in *Proceedings of the 1997 IEEE Symposium on Security & Privacy*, pages 154–163, Oakland, California, USA, May 4-7, 1997. IEEE Computer Society Press.

- An established taxonomy would be useful when reporting incidents to incident response teams, such as the CERT Coordination Center. It could also be used in the bulletins issued by incident response teams in order to warn system owners and administrators of new security flaws that can be exploited in intrusions. (The CERT Coordination Center has produced an "Incident Reporting Form" [6] which lists incident categories, however this does not constitute a proper taxonomy since it mixes intent, technique, vulnerability and result categories in an informal manner.)
- If the taxonomy included a grading of the *severity* or impact of the intrusion, system owners and administrators would be helped in prioritizing their efforts.

What is required by such a taxonomy? We have identified some desired (ideal) properties which are worth focusing upon in the formation of the taxonomy.

- The categories in a taxonomy should be mutually exclusive (every specimen should fit in *at most* one category) and collectively exhaustive (every specimen should fit in *at least* one category).
- Every category should be accompanied by clear and unambiguous classification criteria defining what specimens are to be put in that category.
- The taxonomy should be comprehensible and useful not only to experts in security but also to users and administrators with less knowledge and experience of security.
- The terminology of the taxonomy should comply with the established security terminology (something that is not always easy to define).

Landwehr *et al.* [14] made an important general observation:

> "A taxonomy is not simply a neutral structure for categorizing specimens. It implicitly embodies a theory of the universe from which those specimens are drawn. It defines what data are to be recorded and how like and unlike specimens are to be distinguished."

Amoroso pointed out the following properties to consider when inventing or selecting an attack taxonomy [1].

- *Completeness*. The taxonomy should encompass all possible attacks on the target system.
- *Appropriateness*. The selected taxonomy should appropriately characterize the attacks to the target system, that is any constraints on the taxonomy or on the system should be specified and considered before application.
- *Internal versus External Threats*. An attack taxonomy should differentiate attacks that require insider access to a system from those that can be initiated by external intruders who may not have gained access to the system.

## 3. A note on terminology

The terms intrusion, penetration, attack, breach and compromise are often used interchangeably, which can be a source of misunderstanding. Informally, we consider an *intrusion* (or penetration), which is a successful event from the attacker's point of view, to consist of: 1) an *attack* in which a *security flaw* (or vulnerability) is exploited, and 2) a *breach* (or compromise) which is the resulting violation of the explicit or implicit security policy of the system. An attack that does not lead to a breach is considered unsuccessful, although it may provide the attacker with some information, at least that the attempted attack does not work for some reason. However, the distinction between breach and intrusion is neither strict nor crucially important for the following discussion.

We have adopted a wide view of the *system* concept, according to which users can sometimes be considered part of the system or at least seen as part of the system context or environment. This is common in the field of safety engineering [22] and we also find it necessary to the security perspective. One reason for including users in the system concept is that sometimes an attack will be successful only when there are other users in the system who unknowingly interact with the attacker. For example, if an attacker plants a Trojan horse, it must be run by a credulous user in order to work. Another reason for adopting a holistic view of the system, rather than studying separate components when analysing intrusions, is that it is usually not important to the attacker *how* or *where* the intrusion is made, as long as the result is the desired one.

## 4. Previous work

Through the years, several classifications of intrusions have been presented, some concentrated on the intruders and their methods (that is the *threat* or *intrusion technique*) and others on the characteristics of the computer system that make the intrusion possible (that is the *vulnerability* or *security flaw*). The latter classifications do not usually take into account the exploitation of the categorized flaws, while the former often describe the exploited flaw in conjunction with the exploitation technique. For the sake of completeness, both types of classification are included in this survey of previous work.

### 4.1. Classifications of intrusion techniques and threats

An early work is that of Lackey, in which six categories of penetration techniques were presented [13]. The classification is "based on many examples of actual system penetration", although no references are presented.

Neumann and Parker categorized computer misuse techniques into nine classes on the basis of data from about 3,000 computer abuse cases collected by the two authors over a period of 20 years [19]. The authors emphasize that their classes are not mutually exclusive in the sense that actual computer abuse cases often involve techniques from several classes. The classes are listed in Table 1. The order is roughly from the physical world (Class NP1) to the hardware (Class NP2) to the software (Class NP3 and higher), and from unauthorized use to misuse of authority.

We found the classification suggested by Neumann and Parker interesting since it appears to be well-founded and to cover most of the known techniques. It also has an elegant feature, namely the inherent grading of the classes, from external attacks to authorized users misusing their privileges. It is not perfect, however, and some of its shortcomings are discussed in Section 7 (Neumann presented a revised and extended version of the scheme [18], but we prefer the original version since the new scheme does not clearly separate technique from vulnerability or result).

Brinkley and Schell [5] categorized what they call information-oriented computer misuse (regarding the security aspects *confidentiality* and *integrity*, but not *availability*, which the authors call resource-oriented computer misuse) into six different classes, which are not mutually exclusive. No specific support for the classification scheme is presented, except for a small number of examples from other cited references.

In his Ph.D. thesis, Kumar made a classification of intrusions based on the "signatures" (patterns) they leave in the audit trail of the system [12]. The classification is intended for use in intrusion detection systems based on pattern matching. Consequently, it does not consider intrusions that do not leave tracks in the audit trail, for example passive wiretapping.

### 4.2. Classifications of security flaws

In a general sense, a security flaw in a computer system is a kind of "bug". Beizer presented a taxonomy of bugs that concentrates on where in the software development process the bug is introduced [3].

**Table 1. Computer misuse techniques [19].**

| Class | Description |
|---|---|
| NP1 External misuse | Generally nontechnological and unobserved, physically separate from computer and communication facilities, for example visual spying. |
| NP2 Hardware misuse | a) Passive, with no (immediate) side effects.<br>b) Active, with side effects. |
| NP3 Masquerading | Impersonation; playback and spoofing attacks etc. |
| NP4 Setting up subsequent misuse | Planting and arming malicious software. |
| NP5 Bypassing intended controls | Circumvention of existing controls or improper acquisition of otherwise denied authority. |
| NP6 Active misuse of resources | Misuse of (apparently) conferred authority that alters the system or its data. |
| NP7 Passive misuse of resources | Misuse of (apparently) conferred reading authority. |
| NP8 Misuse resulting from inaction | Failure to avert a potential problem in a timely fashion, or an error of omission, for example. |
| NP9 Use as an indirect aid in committing other misuse | a) As a tool in planning computer misuse etc.<br>b) As a tool in planning criminal/unethical activity. |

Landwehr *et al.* constructed a taxonomy of computer program security flaws, exemplified with 50 documented case studies of security flaws in different computing environments [14]. The flaws are categorized with respect to three characteristics or, as we suggest, in three *dimensions*. The dimensions are genesis (*how* did the flaw enter the system?), time of introduction (*when* did it enter the system?) and location (*where* in the system is it manifested?).

In a classic article, Saltzer and Schroeder present eight design principles for protection mechanisms, one of them being the well-known principle of least privilege [23]. Starting from these principles, and using UNIX as an example of an "unsecure operating system", Hogan categorized security flaws in stand-alone systems and distributed environments [9]. This classification is chiefly concerned with why the flaws are present in the system.

Based on 49 cases in which UNIX security faults have led to intrusions, Aslam devised a taxonomy of security faults, as well as a design of a database for vulnerability data [2]. Aslam provides selection criteria that enable a distinct classification of the 49 cases. Only faults embodied in software are included.

## 5. The intrusion experiment

This section briefly outlines the arrangement of the experiment; for details see Olovsson *et al.* [20]. The target system consisted of a set of 24 SUN ELC diskless workstations connected to one file-server, all running SunOS 4.1.2. The attackers were 24 undergraduate students taking a course in applied computer security. They were all legal users of the system with normal user privileges and with physical access to all workstations except the file-server.

During this time, the system was in operational use for other laboratory courses taken by undergraduate students at the Department of Computer Engineering. The system itself was a 'standard' configuration, and thus not expected to differ significantly from other similar systems in use; it was supervised by an experienced system administrator. All standard monitoring and accounting features were enabled in the system to allow us to monitor the activities of each user account and to measure the resources each attacker spent during the breach process.

Through questionnaires, we know that the attackers did not consider themselves particularly knowledgeable about computer security issues compared with other students of the Computer Science and Engineering program, except for a certain degree of interest which made them choose to take the course in the first place. The attackers worked in groups of two. It was a deliberate choice to let 'normal' users attack the system, as opposed to professional attackers with experience from other systems. The attackers were informed that some specific activities were prohibited, namely doing physical damage to the system, attacking other systems, cooperating between groups or affecting the operation of other users on the system without first consulting the experiment coordinator. All attacking activities were to be carefully documented and reported to the coordinator.

A major motivation for the attackers was that the experiment was a compulsory part of the course they were taking. They were also given a general description of the overall objectives of the experiment so that they had a complete understanding of why certain rules must be obeyed, and why and in what way they should report their actions.

The attackers were told that a breach occurs *whenever the attackers succeed in doing something they are not normally allowed to do*, for example to use another user's account. It is still somewhat difficult to determine objectively whether a given event is a valid breach or not but, after analysis of attacker reports and system logs, we have acknowledged some 60 separate, valid breaches in this experiment.

## 6. Taxonomy

### 6.1. Introduction

When examining specimens for classification, it should be noted that the specimens often have many different attributes, any of which could be chosen as the basis of the classification. We suggest the use of the term *dimension* for such an attribute. Accordingly, it is important to decide exactly what dimension of an intrusion the classification should be based on, because there are indeed several possibilities: the system component that was attacked; the intent of the attacker; the technique used in the attack; the reason why the exploited flaw is present in the system; the outcome of the intrusion etc. Some classification schemes make this point very clear (for example [14]), while others are less specific.

When we tried to categorize the flaws exploited in our recorded intrusions according to the scheme of Landwehr *et al.* [14], we found that the only feasible dimension, based on the information we had, was location. Since neither the details of the system development process nor the source code was available to us, only a minority of the flaws could be categorized with respect to genesis or time of introduction. Furthermore, for many of our recorded intrusions, it is not a trivial task to determine the actual flaw. Consider for example the scenario in which an attacker feeds the password file to a password-guessing program that tries words from various dictionaries. What is the vulnerability that makes this attack possible? Is it the fact that every user can read the encrypted passwords in the password file? Or is it

the fact that some users tend to choose easy-to-guess passwords? Or is the encryption method not sufficiently sophisticated? Or is a single reusable password simply insufficient for the authentication of users?

We would like to be able to make a classification from the system owner's point of view. That is why we focus on the external observations of attacks and breaches which the system owner can make. An owner of a system is usually unable to categorize security flaws in detail. This is because most of the software and hardware is purchased from system vendors; source code and internal design is most often proprietary and not available from the vendor.

We believe that the dimensions of an intrusion that are most interesting to system owners are *intrusion techniques* and *intrusion results*. Details of the *intrusion technique* are needed to gain an understanding of intruders and the threat that system owners face. In addition, with this knowledge, it is often possible for the administrator to apply a quick fix to stop further intrusions of this kind while waiting for a patch from the vendor. This quick fix can be, for example, to clear the set-user-id bit of a flawed program or to remove a service completely (this usually has a negative impact on the service to legal users of the system). Information about the *intrusion result* is needed for the system owner to judge how critical the intrusion is according to the security policy of the system. For example, in some systems, disclosure of confidential information is considered much worse than denial of service while, in other systems, it is exactly the opposite. Another important field of application for data on intrusion results and techniques is the design of intrusion detection systems.

Our classification of intrusion techniques is presented in Table 2 and our classification of intrusion results in Table 3. For each category of the two dimensions, we give the number of intrusions from our experiment that fit in the category. The number is zero in some categories; nevertheless they are included as we believe that such intrusions are possible, although they did not occur in this particular experiment. The dimensions and their categories are explained and illustrated with examples below.

## 6.2. Intrusion techniques

As the scheme of Neumann and Parker [19] appeared to be the most useful of the previous classifications of intrusion techniques, our first step was to try to classify the intrusions made during the experiment in those classes. Since all attackers in our experiment were authorized users of the system, we expected that most of the intrusions would fit into the higher classes. The result was that all of the intrusions could be entered in class NP5, NP6 or NP7 (see Table 1). Our goal was a more fine-grained partitioning, however; thus our next step was to define subclasses below the three classes in the Neumann and Parker scheme.

### 6.2.1. Category NP5: Bypass of intended controls

The category *bypass of intended controls* was divided into three subclasses: *password attacks*, *spoofing privileged programs*, and *utilizing weak authentication*.

*Password attacks*, as already pointed out by Neumann and Parker, is a broad subclass that includes all intrusions in which passwords are in some way involved. We decided to further divide this subclass into the third-level categories *capture* and *guessing*, since different countermeasures apply to the two techniques. *Spoofing privileged programs* is a technique in which programs executing with higher privileges are tricked to perform illicit operations on behalf of the attacker. *Utilizing weak authentication* is the technique of taking advantage of the fact that the system does not perform proper authentication of the originator of certain requests. Examples of this subclass include: obtaining client root privileges by manipulating the boot process, obtaining server root privileges by executing a set-user-id program generated by a client root, sending e-mail with faked headers by manually interacting with the mailer daemon, and other situations in which the system trusts an identification without requiring any authentication token at all.

### 6.2.2. Category NP6: Active misuse of resources

The category *active misuse of resources* was divided into the two subclasses *exploiting inadvertent write permission* and *resource exhaustion*.

*Exploiting inadvertent write permission* includes exploitation of the fact that many system objects are by default world writable. This means that any user on the system can modify these objects, although this is seldom the system (or object) owner's intention; it is the same for group writable objects. These objects are often found by using the techniques of category NP7. *Resource exhaustion* is a technique used to cause denial of service, for example by consuming all available disk space. UNIX is very susceptible to this kind of attack, but it is often easy to track down the source of the problem [21], making the attack only temporarily useful. The participants in the intrusion experiment were explicitly told not to use an attack of this kind, for example the command "`while true fork()`", which would effectively stop other users from starting new processes. If they had more innovative ideas for denial of service attacks that could not be traced, such attacks could be tried after discussion with the experiment coordinator at times when no normal users were present.

**Table 2. Taxonomy of intrusions: Intrusion techniques.**

| Category | | | Number of intrusions |
|---|---|---|---|
| NP5 Bypassing intended controls | Password attacks | Capture | 6 |
| | | Guessing | 12 |
| | Spoofing privileged programs | | 6 |
| | Utilizing weak authentication | | 13 |
| NP6 Active misuse of resources | Exploiting inadvertent write permission | | 12 |
| | Resource exhaustion | | 0 |
| NP7 Passive misuse of resources | Manual browsing | | 1 |
| | Automated searching | Using a personal tool | 0 |
| | | Using a publicly available tool | 8 |

**Table 3. Taxonomy of intrusions: Intrusion results.**

| Category | | | Number of intrusions |
|---|---|---|---|
| Exposure | Disclosure of confidential information | Only user information disclosed | 0 |
| | | System (and user) information disclosed | 10 |
| | Service to unauthorized entities | Access as an ordinary user account | 19 |
| | | Access as a special system account | 0 |
| | | Access as client root | 3 |
| | | Access as server root | 5 |
| Denial of service | Selective | Affects a single user at a time | 2 |
| | | Affects a group of users | 0 |
| | Unselective | Affects all users of the system | 2 |
| | Transmitted | Affects users of other systems | 0 |
| Erroneous output | Selective | Affects a single user at a time | 6 |
| | | Affects a group of users | 0 |
| | Unselective | Affects all users of the system | 8 |
| | Transmitted | Affects users of other systems | 3 |

#### 6.2.3. Category NP7: Passive misuse of resources

The category *passive misuse of resources* is the "read" counterpart of NP6. It is natural to divide the techniques into *manual browsing* and *automated searching*; the latter involves the use of a special tool program designed to find security problems in a system. Such a program can be either constructed by the attacker for the particular attack or a general tool fetched from a public archive. Several such tools are available, for example COPS [8], which was a popular instrument among the participants in our experiment. The formation of third-level categories for distinction between *publicly available tools* and *personal tools* is motivated by detection mechanisms. It is often easy to design an intrusion detection system to recognize the characteristics of a public tool, while this is more difficult for tools that are previously unknown (compare with the problem of virus detection).

### 6.3. Intrusion results

What are the consequences of an intrusion? This question is more difficult to answer than might appear at first glance. Usually, it is meaningful to consider only the immediate result that characterizes a breach, because the total outcome of an intrusion depends on how the attackers move on from the initial breach. For example, if the attackers gain root access on the file-server, they can do virtually anything to the system and the final consequences are impossible to assess completely. In our intrusion experiment, the attackers were told to stop when they had obtained the desired higher privileges, as we did not want them to disturb the work of ordinary system users [20]. In terms of real-time intrusion detection, another reason for concentrating on the immediate result is that it is desirable to detect the intrusion and take preemptive action as early as possible, preferably before any damage is done [10].

However, it is not obvious what should be considered the immediate result. A typical example is password-guessing. The very first result of a successful password-guessing attack is that the attackers gain knowledge of the user's password. A password is not just any piece of information, however, because the immediate implication is access to the user's account on the system. We decided to adopt a practical point of view, whereby we consider the result of a password-guessing attack to be access to the account in question.

Another example is the planting of a Trojan horse. The initial event is a modification or creation of an object in the system but, if the Trojan horse is never activated by a credulous user or system process, there is no detrimental result from the system owner's point of view. Consequently, we consider the result of the *activation* of the Trojan horse to be the result of the intrusion (although it would be desirable to detect the presence of the Trojan horse before it is activated). This example also illustrates that there is no point in considering intent when categorizing results. The creation and insertion of the Trojan horse is most likely done with malicious intent, but the activation can be considered an accident. Although we are concerned primarily with intentional attacks, the same results could in fact be caused by accidents (see [18] for more examples).

We decided to base our classification of intrusion results on the three traditional aspects of computer security: confidentiality, availability and integrity. The aspect of confidentiality is extended as suggested by Meadows [15] to *exclusivity*, to denote not only protection against unauthorized access to confidential information, but also protection against unauthorized use of the system. A breach of exclusivity results in *exposure*, a breach of availability results in *denial of service* and a breach of integrity results in *erroneous output*. Those are the top-level categories of our classification of intrusion results.

#### 6.3.1. Exposure

The exposure category is naturally divided into the subclasses *disclosure of confidential information* and *service to unauthorized entities*.

*Disclosure of confidential information* is further divided into the third-level categories *only user information disclosed* and *system (and user) information disclosed*, since we believe that cases of the former class sometimes (but not always) can be considered less severe than those of the latter. Examples of *disclosure of confidential information* include the following.

***Reading backup tapes*** The tape streamer used for backups of the file-server was world-readable. The attackers in our experiment discovered that tapes were automatically ejected immediately after the backup procedure had finished writing to the tape. However, old tapes were reused and could be read from the time the tape was inserted to the start of the backup procedure. The result was that an older copy of the entire contents of the server's disks could be read by anyone on the system. (Result: *system (and user) information disclosed*; Technique: *manual browsing*).

***Spoofing ARP*** The program `/etc/arp` runs with the effective group id of `kmem` and, when a file which is readable to this group, for example `/dev/kmem` or `/dev/eeprom`, is fed to the program, parts of the file will be displayed as syntax error messages. (Result: *system (and user) information disclosed*; Technique: *spoofing privileged programs*).

*Service to unauthorized entities* is divided into third-level categories reflecting the privileges associated with the service delivered. The category *access as an ordinary user account* concerns either a legal user of the system who gains access to another user's account, or an outsider who gains

access to any user account on the system. *Access as a special system account* means an account with higher privileges than an ordinary user account, but not super-user (root) access. An example from UNIX is `bin` or any other account that owns system files. The reason why we make a distinction between *access as client root* and *access as server root* is that in most client-server environments, the super-user on a client host has no special privileges on the server host. This is because users often have complete physical access to the client workstations, and consequently can manipulate the hosts in many different ways; they can reboot the machines, connect or replace storage devices or network connection cables etc. In fact, workstations to which the users have complete physical access cannot be trusted at all, although this is ignored in many systems (with the exception of the root identity on the server as mentioned above). This was realized in MIT's Project Athena, where the root password for the public workstations was not even kept secret; Kerberos was developed instead and used for user authentication [24]. Examples of *service to unauthorized entities* include the following.

***Automated password-guessing*** The use of an automated tool for password-guessing based on dictionaries of likely passwords, a widely discussed and utilized technique, was also successfully used in our experiment. Many user accounts with simple passwords were compromised, but the root password was never guessed. (Result: *access as an ordinary user account*; Technique: *password attacks—guessing*).

***Manipulating the boot process*** Several attackers tried to reboot a client host in single-user mode. Since this was successfully utilized in an earlier experiment to gain client root access, the system administrator had enabled the PROM password feature of the workstations to prevent this type of attack. However, some attackers found a method by which they could still reboot the host in single-user mode to become client root without being prompted for a password. (Result: *access as client root*; Technique: *utilizing weak authentication*).

#### 6.3.2. Denial of service

The subclasses *selective* and *unselective* for denial of service were suggested by Needham [17]. The third-level categories should be self-explanatory. By *transmitted*, we mean that the intrusion affects the service delivered by other systems to their users, not the service delivered by our system to other systems. In the latter case, other systems can in fact be seen as users of our system. There were no intrusions that caused denial of service on other systems in the experiment, but such intrusions are indeed possible. For example, an attacker can make a host on the system use the same IP address as a host on another system, something which normally causes both hosts to lose contact with the network. The possible range of this particular attack depends on the network configuration [4]. We have not separated transmitted attacks as selective or unselective, because it is difficult to define what unselective would mean for a transmitted attack, especially for the denial of service category. We hope that it is not possible for a computer on the Internet to cause denial of service on *all* connected systems (although the result of the Internet Worm incident in 1988 was too close for comfort). An example of *denial of service* is given here.

***Causing a crash by remote copy to audio device*** There was a bug that caused a machine to crash immediately if the remote copy command `rcp` was invoked with the target `/dev/audio`. If executed on the server, the whole system would go down. This was clearly a system bug, but the audio device should not be readable or writable to any user except the user currently logged in at the console. (Result: *unselective*; Technique: *exploiting inadvertent write permission*).

#### 6.3.3. Erroneous output

In the formation of the erroneous output category, it soon became evident that the same subcategories could be used as in the denial of service category. "Output" is used in a wide sense, and denotes more than what is shown on the user's terminal or sent on a network connection. Modifications of system objects, such as the contents of files on hard disks or data structures in main memory, are also considered as "output", and when that output is the result of an intrusion, the intrusion belongs to this category. Examples of *erroneous output* include the following.

***Spoofing Xterm*** The X Windows terminal program `xterm`, running with the effective user id of `root`, had a flawed logging facility (CERT Advisory CA-93:17) which could be used to create any file or append to any existing file. Although this could be used to gain access as server root, we categorized the result as erroneous output, which was the immediate result. Our decision is supported by the fact that it is not obvious how to move on from the first step, that is to gain root access. (Result: *unselective*; Technique: *spoofing privileged programs*).

***Faking e-mail*** By manually communicating with the mailer daemon, attackers can send e-mail messages with faked headers, particularly false sender identity, to any other system on the Internet. (Result: *transmitted*; Technique: *utilizing weak authentication*).

## 7. Discussion

The classification of intrusion techniques proposed by Neumann and Parker [19] is of course not perfect, nor is our extension of their scheme. It can be discussed for example

whether all kinds of attacks involving passwords in one way or another should actually belong in class NP5, as stated by Neumann and Parker, or whether some belong in class NP7. Another problem, which always accompanies attempts to classify human behaviour, is how to obtain an unambiguous classification. The classification of intrusion techniques indirectly involves the intentions of the system owner and of the attacker, which are not always clear and logical. Therefore, it is sometimes a question of interpretation as to whether a certain intrusion belongs in one class or the other, or in both. Our subclasses are designed to be mutually exclusive with respect to technique but, as noted by Neumann and Parker, an actual case of abuse is often complex and involves several techniques. As observed by Meadows [16], it depends on the level of abstraction whether an action that is part of an attack is considered atomic or complex.

The classification of intrusion results is perhaps easier in the sense that the classes are in all essential respects mutually exclusive. The problem here lies in determining what it is meaningful to consider as the outcome of the intrusion, as discussed in Section 6.3. Although it would probably be desirable to include a grading of the severity of the intrusions, this is often a subjective and system-dependent property; it is therefore left to system owners who can judge how severe a particular result category is in their system, according to their security policy.

A significant question is whether our scheme is applicable to other systems and circumstances besides those of the experiment from which it was derived. Our proposed answer is based on the properties specified by Amoroso [1], as cited in Section 2.

- As to the *result* dimension, we believe that, with our definition of exposure, the top and second levels of our taxonomy satisfy Amoroso's *completeness* and *appropriateness* properties for most systems. The third level is more specialized and may fit only similar systems. We do not find any reason to differentiate between internal and external attacks in the *result* dimension, since the results can be the same regardless of the origin of the attack. For example, an intrusion in which an outsider guesses a user password and logs in as that user is categorized as *exposure – service to unauthorized entities – access as an ordinary user account*.

- The *technique* dimension is less general, as it is an extension of a more general scheme. For the system in our experiment, it is complete and appropriate. Since many systems in industrial and academic environments are very similar to our experimental system, we believe that our scheme is likely to have a wide field of application. Our experiment concerns only internal attacks, however external attacks are intended to fit in the lower classes of the Neumann and Parker scheme.

Although the size of the experiment is too small to draw strong conclusions about distribution in general, it is still interesting to examine the number of intrusions in the classes of the two dimensions we have studied. Figure 1 shows this distribution and is also a clear illustration of why the term dimension is appropriate. The figure shows that some techniques have a one-to-one correspondence to the result, while other techniques can be used to reach many different kinds of results.

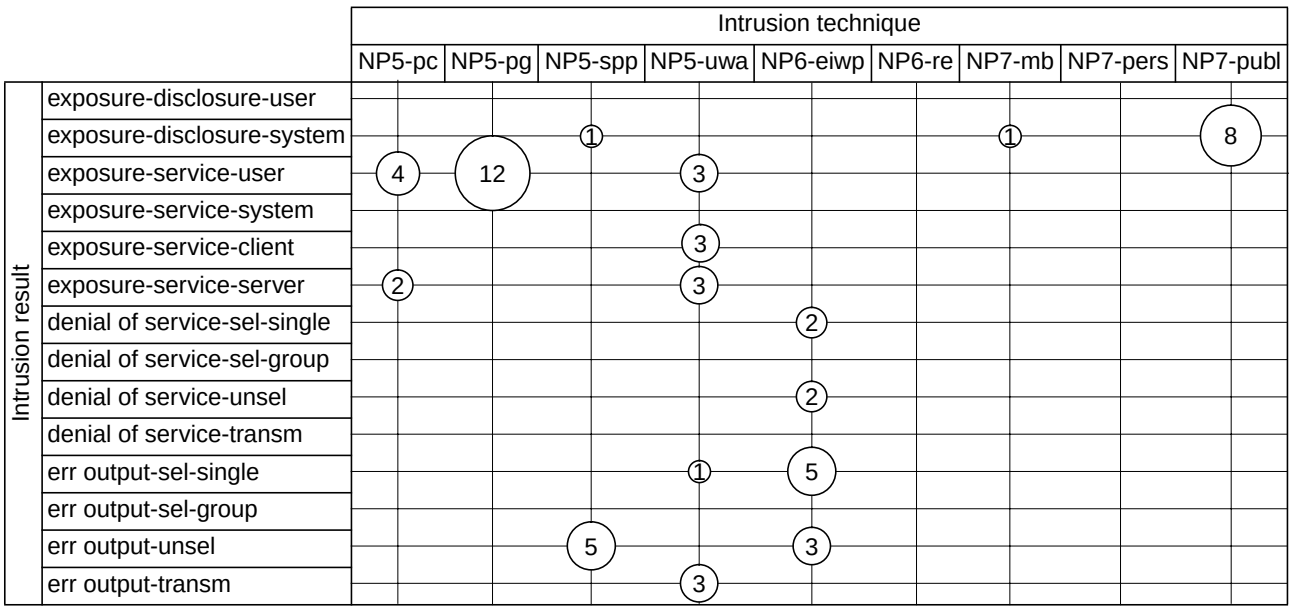


**Figure 1. Distribution of intrusions in the two dimensions.**

## 8. Conclusions

We have presented a classification scheme for computer security intrusions, in which the classification is made with respect to the intrusion technique and the intrusion result, with the needs of system owners and administrators in mind. By using data from a realistic intrusion experiment, we have shown that the scheme is likely to be generally applicable. We believe that the proposed scheme will, with further application, evaluation and refinement, be a good candidate for a generally accepted taxonomy of intrusions.

## Acknowledgments

The authors are grateful to several people who have read earlier versions of this paper and made valuable comments and suggestions. We thank especially Tomas Olovsson who also coordinated the intrusion experiment, and Per Kaijser for his support and suggested improvements.

The introductory quotation from Systema Naturæ was translated from the Latin original into English by M. S. J. Engel-Ledeboer and H. Engel, Nieuwkoop, Holland, 1964.

## References

[1] E. Amoroso. *Fundamentals of Computer Security Technology*. Prentice-Hall, 1994.

[2] T. Aslam. A taxonomy of security faults in the Unix operating system. Master's thesis, Purdue University, West Lafayette, Indiana, USA, Aug. 1995.

[3] B. Beizer. *Software Testing Techniques*. Van Nostrand Reinhold, second edition, 1990.

[4] S. O. Bradner. A practical perspective on routers. In D. C. Lynch and M. T. Rose, editors, *Internet System Handbook*, chapter 7. Addison-Wesley, 1993.

[5] D. L. Brinkley and R. R. Schell. What is there to worry about? An introduction to the computer security problem. In M. D. Abrams, S. Jajodia, and H. J. Podell, editors, *Information Security: An Integrated Collection of Essays*, pages 11–39. IEEE Computer Society Press, 1995.

[6] CERT Coordination Center, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA 15213-3890, USA. *Incident Reporting Form*, Feb. 28, 1996. Version 3.0.

[7] Commission of the European Communities. *Information Technology Security Evaluation Criteria*, June 1991. Version 1.2.

[8] D. Farmer and E. H. Spafford. The COPS security checker system. In *Proceedings of the Summer USENIX Conference*, pages 165–170, Anaheim, California, USA, June 1990. USENIX Association.

[9] C. B. Hogan. Protection imperfect: The security of some computing environments. *Operating Systems Review*, 22(3):7–27, July 1988.

[10] K. Ilgun, R. A. Kemmerer, and P. A. Porras. State transition analysis: A rule-based intrusion detection approach. *IEEE Transactions on Software Engineering*, 21(3):181–199, Mar. 1995.

[11] E. Jonsson and T. Olovsson. An empirical model of the security intrusion process. In *Proceedings of the Eleventh Annual Conference on Computer Assurance (COMPASS '96)*, pages 176–186, Gaithersburg, Maryland, USA, June 17–21, 1996. IEEE.

[12] S. Kumar. *Classification and Detection of Computer Intrusions*. PhD thesis, Purdue University, West Lafayette, Indiana, USA, Aug. 1995.

[13] R. D. Lackey. Penetration of computer systems an overview. *Honeywell Computer Journal*, 8(2):81–85, 1974.

[14] C. E. Landwehr, A. R. Bull, J. P. McDermott, and W. S. Choi. A taxonomy of computer program security flaws. *ACM Computing Surveys*, 26(3):211–254, Sept. 1994.

[15] C. A. Meadows. An outline of a taxonomy of computer security research and development. In *Proceedings of the 1992–1993 ACM SIGSAC New Security Paradigms Workshop*, Little Compton, Rhode Island, USA, Sept. 22–24, 1992 and Aug. 3–5, 1993. IEEE Computer Society Press.

[16] C. A. Meadows. A representation of protocol attacks for risk assessment. In *Proceedings of DIMACS Workshop on Network Threats*, Piscataway, New Jersey, USA, Dec. 4–6, 1996. To appear.

[17] R. M. Needham. Denial of service: An example. *Communications of the ACM*, 37(11):42–46, Nov. 1994.

[18] P. G. Neumann. *Computer-Related Risks*. ACM Press and Addison-Wesley, 1995.

[19] P. G. Neumann and D. B. Parker. A summary of computer misuse techniques. In *Proceedings of the 12th National Computer Security Conference*, pages 396–407, Baltimore, Maryland, USA, Oct. 10–13, 1989.

[20] T. Olovsson, E. Jonsson, S. Brocklehurst, and B. Littlewood. Towards operational measures of computer security: Experimentation and modelling. In B. Randell *et al.*, editors, *Predictably Dependable Computing Systems*, ESPRIT Basic Research Series, chapter VIII. Springer-Verlag, 1995.

[21] D. M. Ritchie. On the security of UNIX, May 1975. Reprinted in *UNIX System Manager's Manual*, 4.3 Berkeley Software Distribution. University of California, Berkeley, USA, Apr. 1986.

[22] J. Rushby. Critical system properties: Survey and taxonomy. Technical Report CSL-93-01, Computer Science Laboratory, SRI International, Menlo Park, CA 94025, USA, May 1993. Revised Feb. 1994.

[23] J. H. Saltzer and M. D. Schroeder. The protection of information in computer systems. *Proceedings of the IEEE*, 63(9):1278–1308, Sept. 1975.

[24] J. G. Steiner, C. Neuman, and J. I. Schiller. Kerberos: An authentication service for open network systems. In *Proceedings of the USENIX Winter Conference*, pages 191–202, Dallas, Texas, USA, Feb. 9–12, 1988. USENIX Association.

[25] U.S. Department of Defense. *Trusted Computer System Evaluation Criteria*, Dec. 1985. DoD 5200.28-STD.