A new taxonomy of web attacks suitable for efficient encoding

Gonzalo Álvarez *, Slobodan Petrović

Instituto de Física Aplicada, Consejo Superior de Investigaciones Científicas, Serrano 144—28006 Madrid, Spain

Abstract

Web attacks, i.e. attacks exclusively using the HTTP/HTTPS protocol, are rapidly becoming one of the fundamental threats for information systems connected to the Internet. When the attacks suffered by web servers through the years are analyzed, it is observed that most of them are very similar, using a reduced number of attacking techniques. It is generally agreed that classification can help designers and programmers to better understand attacks and build more secure applications. As an effort in this direction, a new taxonomy of web attacks is proposed in this paper, with the objective of obtaining a useful reference framework for security applications. The use of the taxonomy is illustrated by means of multiplatform real world web attack examples. Along with this taxonomy, important features of each attack category are discussed. A semantic-dependent web attack encoding scheme is also defined that, together with the taxonomy, can be used to process the attack information with low time and memory consumption. Applications of the taxonomy and the encoding scheme are described, such as intrusion detection systems and application firewalls.

Key words: Web attacks; Taxonomy; Source coding; Intrusion detection; Application firewalls

1 Introduction

With the increasing use of Internet as a commercial channel, there is a growing number of web sites deployed to share information, offer on-line services, sell all sorts of goods, digital or physical, distribute news and articles, etc. On the other hand, the number of attacks is increasing in parallel: theft of private

* Corresponding author: Email: gonzalo@iec.csic.es

information, defacing of homepages, denial of service, worm spreading, and fraud, are but a few of the most common and frequent attacks in cyberspace [21,28].

By *web attacks*, we understand network attacks exclusively using the HTTP/HTTPS protocol. When the web attacks recorded through the years are analyzed, it is observed that most of them are recurrent attacks. In fact, they correspond to a limited number of types of attacks. Thus, it is generally agreed that classification can help designers, programmers, and security analysts to better understand attacks and build more secure applications.

In an effort to create a common reference language for security analysts, a number of taxonomies of computer attacks and vulnerabilities have appeared in recent years [7,14,18,20,27,29,32]. The shortcoming of such taxonomies is that they often include categories unsuitable for the classification of web attacks. Even when their categories can be used in a web context, they fail to cover all the subtleties of web attacks. For example, entry point, target, HTTP Verb, and HTTP Header are web-specific categories that we consider important for a more accurate classification of web attacks, and these are not covered by general taxonomies. In addition, some categories that can also be met in general taxonomies, such as vulnerability, need to take web-specific values (e.g. Code injection, HTML manipulation).

In this paper we propose a taxonomy of web attacks, taking into account a number of important features of each attack category. The role of these features as well as their importance for each of the attack categories is discussed thoroughly. The proposed taxonomy represents an effort to cover known attacks and also some attacks that might appear in the future.

Once the taxonomy has been defined and the role and importance of its categories have been explained, we define a semantic-dependent encoding scheme to encode all relevant information contained in web attacks. The encoding scheme removes local redundancy in the description of the attacks, thus enabling time and memory savings in the processing of the attack information. The vectors (generally of different lengths) obtained in the encoding process can be used in a number of applications, such as intrusion detection systems or application firewalls, where the classification of attacks is needed. With such an encoding scheme, the classification techniques that employ special distance measures (such as, for example, edit distance [25]) can be used, which reduces both memory consumption (by omitting redundancy) and computational effort (by simplifying the compression/decompression process).

The paper is organized as follows. In Section 2, the web attack properties are thoroughly discussed and the taxonomy of web attacks is defined. In Section 3 the encoding scheme of the attack descriptions is given and the advantages of

such scheme over other possible encoding schemes is explained. Section 4 gives some examples of attacks encodings using the proposed taxonomy and the encoding scheme. In Section 5, we estimate the coverage of real attack space by the proposed taxonomy. Section 6 provides some ideas about which applications might benefit from this taxonomy and encoding. Section 7 concludes the paper.

2 Web attack properties

A *taxonomy* is a classification scheme that partitions a body of knowledge and defines the relationship of the objects. *Classification* is the process of using a taxonomy for separating and ordering [14]. According to [3], satisfactory taxonomies have classification categories with the following characteristics:

(1) Mutually exclusive: the categories do not overlap.
(2) Exhaustive: taken together, the categories include all the possibilities.
(3) Unambiguous: clear and precise so that classification is not uncertain, regardless of who is classifying.
(4) Repeatable: repeated applications result in the same classification, regardless of who is classifying.
(5) Accepted: logical and intuitive so that categories could become generally approved.
(6) Useful: could be used to gain insight into the field of inquiry.

First, we introduce a novel model of web attacks based on the concept of *attack life cycle*. By attack life cycle we understand a succession of steps followed by an attacker to carry out some malicious activity on the web server, as depicted in Figure 1. The attacker gets through an entry point, searching for a vulnerability in the web server or web application, which might be exploited to defeat some security service. The vulnerability is realized by an action, using some HTTP verb and headers of certain length, directed against a given target and with a given scope. The attacker might obtain some privileges that depend on the type of attack. Our taxonomy of web attacks is based on the attack life cycle defined in this way.

At every stage of the life cycle we define the following classification criteria or classifiers:

(1) Entry point: where the attack gets through.
(2) Vulnerability: a weakness in a system allowing unauthorized action.
(3) Service (under threat): security service threatened by the attack.
(4) Action: actual attack against the web server exploiting the vulnerability.
(5) Length: the length of the arguments passed to the HTTP request.

(6) HTTP element: Verbs and headers needed to perform the attack.
(7) Target: the aim of the attack.
(8) Scope: impact of the attack on the web server.
(9) Privileges: privileges obtained by the attacker after the successful completion of the attack.

In the next subsections each of these criteria are covered in detail and their relevance explained.

2.1 Entry point

The fact that a web application is successfully attacked usually means that there is a vulnerability that is exploited by the attacker. This vulnerability might be found in the web server software or in the web application code itself. Thus, according to the *entry point* of the attack, we distinguish between *web server software* attacks and *web application* attacks.

2.1.1 Web server software attacks

All web server software, regardless of platform or manufacturer, unintentionally hides a number of vulnerabilities, which allow the application to be used in a different way than intended. Many of these vulnerabilities are disclosed to the public, for example published in security forums and bulletins. Upon notification of the vulnerability, the manufacturer usually releases a patch or service pack which should correct the error. In the meantime, since the patch is released until all servers are correctly patched, many vulnerable servers exist.

2.1.2 Web application attacks

Web application-level attacks refer to the vulnerabilities inherent in the code of a web application itself, regardless of the technology in which it is implemented or the security of the web server/back end database on which it is built [30]. Attacks against web-based mail are also included in this category.

The origin of these vulnerabilities may be errors in HTML forms, client-side scripts, server-side scripts (.asp, .jsp, .php, .pl, etc.), business logic objects (COM, COM+, CORBA, etc.), SQL sentences processing, etc.

2.2 Vulnerability

In order to reach the desired result, an attacker must take advantage of a computer or network vulnerability. A *vulnerability* is a weakness in a system allowing unauthorized action. We define the following vulnerabilities in web applications.

2.2.1 Code Injection

Code injection vulnerabilities allow for injecting arbitrary user-chosen code into a page. These vulnerabilities arise from nonexistent or poorly designed input validation routines on the server-side. The main categories of code injection are:

- Script injection: The attack involves web servers that dynamically generate HTML pages. If these servers embed browser input in the dynamic pages that they send back to the browser, these servers can be manipulated to include content in the dynamic pages that will allow malicious scripts to be executed. This attack does not modify web site content; rather, it inserts new, malicious script that can execute at the victim's browser in the information context associated with a trusted server. Cross-Site Scripting is the most common form of script injection [22,38].
- SQL injection: An attacker creates or alters existing SQL commands to gain access to unintended data or even the ability to execute system level commands on the host. See for example [4–6].
- XPath injection: XPath is a language for addressing parts of an XML document. An attacker can modify search strings to access unauthorized data in XML documents.

2.2.2 Canonicalization

Canonicalization vulnerabilities occur when an application makes a security decision based on a name (a filename, a folder name, a web address), without having in mind the fact that the name may be expressed in more than one way [15].

The most common way of exploiting canonicalization issues is in the form of *path traversal* attacks, which allow a malicious user to execute commands or view data outside of the intended target path. Path traversal vulnerabilities arise normally from unchecked URL input parameters, cookies, or HTTP request headers. In most cases, a path traversal attack inherits the permissions of the application being executed which may then access any file allowable using those permissions.

2.2.3 HTML manipulation

HTML manipulation is a vulnerability, which allows a malicious user to modify data sent between the client (web browser) and server (web application), to which the user was not intended to have direct access. Parameter manipulation is often accomplished through:

- URL Query Strings
- Form Fields
- Cookies

Although it is neglected too often, parameter manipulation can be prevented with good input validation techniques on the server side.

2.2.4 Overflows

Buffer overflows have been causing serious security problems for decades [8]. When a program writes past the bounds of a fixed-length buffer previously allocated in memory, this is called a buffer overflow. Reading or writing past the end of a buffer can cause a number of diverse behaviors: programs can act in strange ways or fail completely.

At best, a buffer overflow can stop a server or service where it happens. At worst, it can allow for arbitrary code execution with the same privileges as the program where it is present.

2.2.5 Misconfiguration

If the platform and web server are not correctly configured (carefully assigning file permissions, using non-default paths, etc.) many vulnerabilities can arise.

Moreover, there are web servers, whose default configuration exposes a number of known directories, sample applications, user accounts, etc. Many vulnerabilities have been discovered and exploited over time in these elements too.

2.3 Service under threat

There is a widely accepted set of services or requisites that all systems must satisfy in order to be considered secure [12,26,32]. These services include authentication, confidentiality, integrity, availability, access control, and auditing.

According to the security property under *threat*, we distinguish between the following attacks.

2.3.1 Authentication

The goal of authentication is to determine whether someone or something is, in fact, who or what it is declared to be. Authentication is commonly performed through the use of passwords. Knowledge of the password is assumed to guarantee that the user is authentic.

In a web context, authentication attacks bypass identification controls by using a number of different techniques, such as session hijacking, session replay, session fixation [17], identity spoofing, valid credentials theft, authentication module subversion, and brute-forcing. As a result of a successful attack, an illegitimate user will be identified as a legitimate one.

2.3.2 Authorization

The goal of authorization is to permit authenticated users to do or have something. In multi-user computer systems, a system administrator defines for the system which users are allowed access to the system and what privileges of use (such as access to which file directories, hours of access, amount of allocated storage space, and so forth).

The most common authorization attacks bypass access control by means of buffer overflows (see Sec. 2.2.4), server platform errors, and SQL injection techniques (see Sec. 2.2.1). Their goal is to execute database or operating system commands and access resources not allowed to the unprivileged user. Eventually, a well-implemented privilege escalation attack will result in the unauthorized increase in the domain of access, even ensuring the attacker administrator or root privileges.

2.3.3 Confidentiality

The goal of confidentiality is to protect the information from being disclosed or revealed to entities not authorized to have that information.

Confidentiality attacks show private information contained in files (source code, other users' information, etc.), or in database tables (credit card numbers, personal information, etc.). Depending on the nature of the information disclosed, the attacker can penetrate further into the attacked system or get private information about the targeted company or its customers.

2.3.4 Integrity

Data integrity services act as safeguards against accidental or malicious tampering or modification of data. Changing the value of a data item includes inserting additional data or deleting, modifying, or reordering parts of the existing data.

Common attacks against data integrity include database records manipulation, cookie poisoning, web page alteration (defacing), and forms field modification.

2.3.5 Availability

The goal of service availability is to ensure that users are not unduly denied access to information and resources.

Denial of service (DoS) is the most frequent attack against availability (see Sec. 2.4.8). DoS attacks can be conducted against the machine hosting the web server or against the web server itself. In the first case, the attack will use techniques equally valid for all machines connected to the Internet. In the second case, the attack will exploit vulnerabilities in the web server or the web application to stop normal service.

2.3.6 Auditing

The auditing services provide the system administrator with the means to record security-relevant information, which can be analyzed to detect potential and actual violations of the system security policy. Auditing, or accountability, has three functions: event detection, information collection, and information processing.

Some attacks manage to pass undetected by preventing their being logged by the auditing system.

2.4 Action

Most real-world attacks suffered by web servers are variants or concatenations of a few basic actions or *attack classes*. In this subsection we try to reference those primary attack classes which account for almost all everyday attacks.

We distinguish among actions aimed at three different objectives: server data, user authentication, and web server. Actions directed against data include

read, *modify*, *delete*, and *fabricate*. Actions directed against authentication include *impersonate*, *bypass*, and *search*. Actions directed against the web server include *interrupt*, *probe*, and *other*.

2.4.1 Read

Read is an action to obtain the content of the data contained within a file, database record, or other data medium stored in the web server. Reading does not alter the integrity of the data read. Examples include viewing source code files, and illicitly copying database tables.

2.4.2 Modify

Modify is an action to alter data. We limit our definition to tampering with data on the server side. Examples include changing a database record, or changing the contents of a file. The data continue to be available, but in altered form. Modifying data on the client side, such as the value of the URL, a cookie, or a hidden field in a form, does not fall under this category because the vast majority of web attacks imply this manipulation in some way or another.

2.4.3 Delete

Delete is an action to remove or render an asset in the server irretrievable by other legitimate users. Examples include deleting database objects and server files.

2.4.4 Fabricate

Fabricate is an action to insert counterfeit objects into the system. Examples include adding hacker toolkits to the server file system, creating new user accounts, or inserting records into a database table.

2.4.5 Impersonate

Impersonate is an action to masquerade an illegal user as a legitimate one. Examples include authentication tickets reuse or theft.

2.4.6 Bypass

Bypass is an action to avoid a control mechanism by using an alternative method to access a target. Examples include defeating a forms based web authorization mechanism in order to access protected web resources, such as multimedia files, by simply following a link.

2.4.7 Search

Search is an action to find valid user authentication information. Examples include brute-force attacks, which try different combinations of login/password pairs, or repeatedly forging possible authentication tickets or session ID's to simulate an already validated user.

2.4.8 Interrupt

Interrupt is an action to cause a server to stop operating or offering a service. The most common form of interruption of service attacks are denial of service (DoS) attacks. Their primary goal is to deny the legitimate users access to a particular resource or service.

2.4.9 Probe

Probe is an attempt to determine the characteristics or vulnerabilities of a specific target. Attackers usually begin by gathering information about the target, before stepping into more invasive activities. For instance, the use of uncommon HTTP headers, such as HEAD, or web site mirroring tools, such as Teleport Pro [39] or wget [40], are indicators that an attack might be under way. More aggressive tools, such as Whisker [41] or Nikto [37] are more than probing tools, since they scan the web server in search of vulnerabilities.

2.4.10 Other

When an attacker executes code, but it is impossible to ascertain what the command executed is, we consider the action carried as *other*. Examples include a buffer overflow with a command execution payload or .bat or .cmd files executed via URL name. Usually, these actions are considered a threat against authorization, because although the outcome of such execution cannot be known, we assume it is unauthorized.

2.5 Length

Buffer overflows are the most common form of security attack today, not only in web applications but also in all Internet applications and services. They are the easiest to exploit with the most devastating consequences, usually resulting in complete takeover of the attacked host.

Buffer overflows often need a very long data input to work. Hence, based on the *length* of the attack string, we distinguish between *common-length* and *unusually long* attacks.

2.5.1 Common length

Attacks which are not based on exploiting a buffer overflow seldom submit arguments of a length greater than a certain threshold that can be experimentally defined for a particular web server. Thus, most attacks fall under this category.

2.5.2 Unusually long

Buffer overflows require passing arguments of sufficient length to fill a memory buffer allocated to a particular input, along with some additional data that are written into memory outside the buffer. When these additional data are missing, the buffer overflow usually results in a Denial of Service (DoS) attack (see Sec. 2.4.8), whereas when it is present and accurately crafted, might result in a system command execution.

Unexpectedly long input arguments very likely imply a buffer overflow attack attempt.

2.6 HTTP Verb

In order to be described using our taxonomy, an attack must use the HTTP/HTTPS protocol. An HTTP request consists of a verb and a group of headers. The verb can be any amongst GET, POST, HEAD, SEARCH, PROPFIND, PUT, DELETE, etc.

2.7 HTTP Header

Different attacks use different headers. It is obvious that this category is not mutually exclusive, but actually is unambiguous and we consider it valid to be included in our global taxonomy, since it provides useful information. Most common headers seen in attacks are Host, Cookie, Referer, Translate, etc. This value is included in the vector only when it is relevant to the attack. Otherwise, when irrelevant,it is listed as X.

2.8 Target

Not all web attacks aim at the same target. Some attackers might be interested in taking control over the web server machine and extending their attack further into the server's network. Others might be interested only in obtaining database records, modifying some pages or misleading other users. Based on the *target* of the attack, we distinguish between *application* and *platform* attacks.

2.8.1 Web application

If the attack succeeds, only the application data and functionality will be affected, but not the operating system resources. These attacks are typically aimed at web pages (e.g., obtaining and/or modifying source code), web users (e.g., stealing cookies or passwords using Cross-Site Scripting attacks, web-based mail attacks), and web data (e.g., viewing, changing and/or deleting information in database records).

2.8.2 Platform

Under this attack, the target is beyond the web application, aimed at the platform. The attacker usually seeks after arbitrary command execution, manipulation of machine accounts, tampering with the host's services, obtaining network information, etc. The web server is used as a mere portal to gain access to the internal network.

2.9 Scope

Different attacks affect the web server in different ways. Many attacks only affect one user or a group of users, whereas others have an impact on all users

of the service.

Regarding the *scope* of the attack, we can distinguish between *local* and *universal* attacks.

2.9.1 Local

In these attacks, only one user or a small group of users is affected. An example is stealing one user's personal data by means of a cross-site scripting attack.

2.9.2 Universal

All users will be affected by the attack. Typically, universal attacks include web defacing, database record manipulation, and DoS. If a local attack can be automated and extended to any other user, then it pertains to the category of *universal* attacks. An example are data harvesting attacks.

2.10 Privileges

When an attacker compromises a web server, the ultimate goal is to gain administrator/root privileges. Most web attacks do not allow the attacker to escalate privileges. They run under a restricted account in the server or database. However, misconfiguration of the operating system access control lists, web server permissions, and database users could enable an attacker to reach administrative privileges. There are three categories of users involved in an attack:

- The web application user.
- The database user.
- The operating system level user.

Hence, with regard to the *privilege* obtained by the attack, we can distinguish between *unprivileged* and *administrative* attacks. This category is only applicable when the objective of the attack is obtaining access as a certain user, i.e., attacks against the authentication service (see Sec. 2.3.1).

2.10.1 Unprivileged user

Attacks are run under the identity of a web user, a database user, and/or a system-level user. The attacker can access resources only accessible to that user and thus the impact of the attack is limited.

2.10.2 Administrator/root

If the attacker succeeds in gaining administrative access, the machine is completely compromised. This is the highest level of attack realization.

3 Encoding of the attacks

Now that we have characterized attacks by their fundamental properties, we face the problem of how to encode the attack information in a compact and useful way. Provided that the number of attacks suffered by a web site with medium traffic can grow very quickly, it is not advisable to record unnecessary information in order to prevent running out of storage space prematurely. By using the taxonomy defined in this paper, it is possible to capture the relevant information about attacks, allowing to perform an analysis in order to decide on their severity.

To reduce the amount of the information recorded on the media, the data can be compressed, using some of the source coding techniques. But the use of general compression techniques and algorithms in this case has serious drawbacks. These drawbacks depend on the class of the method. Here we enumerate some of them.

For static defined word schemes to be implemented (e.g. [1,9,11,16,31]) the knowledge of the probabilities of message classes is needed in advance. The problem is that if we treat the descriptions of the attacks as messages, the probability of their appearance varies with time, as new attacks are invented and the remedies are published. Other definitions of messages in this case would be too general and would not lead to a sufficient compression ratio.

To adapt to the changes of the message probabilities, the adaptive Huffman coding can be used (e.g. the FGK algorithm [10] or the Vitter algorithm [33]). But there is no guarantee that the compression ratio achieved by these methods is satisfactory, since these encodings are often outperformed by the static methods [33].

The main drawback of the free-parse methods, such as Ziv-Lempel [34] is that they perform very badly when short messages are encoded. As in the case of the static word schemes, other definitions of messages would not lead to an efficient compression either.

Having in mind the specific type of local redundancy in the descriptions of the attacks (i.e. the descriptions of some of the attacks include some of the attack properties, whereas the descriptions of the others do not), we propose

a semantic dependent data compression method that makes use of different-length vectors. The vectors have different lengths because only the information significant for the particular attack is retained. Besides its advantage over the general data compression schemes considering the efficiency of the short messages encoding, the use of this type of semantic dependent encoding makes possible the direct use of classification and clustering techniques that are often needed in the implementations, since the classification and clustering can be performed without decompressing [25].

To encode the descriptions of the attacks using the semantic dependent method introduced above, a range of positive integers is assigned to each of the attack properties discussed, in the following way.

(1) Entry point (1 bit of information)
    0 - Web server software (ISAPI filters, Perl modules, etc.)
    1 - Web application (HTML, server-side and client-side scripts, server components, SQL sentences, etc.)
(2) Vulnerability (3 bits of information)
    0 - Code Injection (SQL, JavaScript, cross-site scripting, etc.)
    1 - Canonicalization
    2 - HTML manipulation
    3 - Overflows
    4 - Misconfiguration (default directories, sample applications, guest accounts, etc.)
    X - Not applicable
(3) Service (under threat) (3 bits of information)
    0 - Authentication
    1 - Authorization
    2 - Confidentiality
    3 - Integrity
    4 - Availability
    5 - Auditing
(4) Action (4 bits of information)
    0 - Read
    1 - Modify
    2 - Delete
    3 - Fabricate
    4 - Impersonate
    5 - Bypass
    6 - Search
    7 - Interrupt
    8 - Probe
    9 - Other
(5) Length (1 bit of information)
    0 - Expected

1 - Unexpected (unusually long)

(6) HTTP Verb (4 bits of information)
- 0 - GET
- 1 - POST
- 2 - SEARCH
- 3 - PUT
- 4 - DELETE
- 5 - PROPFIND
- 6 - TRACE
- 7 - HEAD
- 8 - OPTIONS

(7) HTTP Header (4 bits of information)
- 1 - Host
- 2 - Cookie
- 4 - Referer
- 8 - Translate
- X - Irrelevant

(8) Target (1 bit of information)
- 0 - Web application (source files, customers' data, etc.)
- 1 - Platform (OS command execution, system accounts, network, etc.)

(9) Scope (1 bit of information)
- 0 - Local (one user affected)
- 1 - Universal (all users affected)
- X - Not applicable

(10) Privileges (1 bit of information)
- 0 - Unprivileged user
- 1 - Administrator/root
- X - Not applicable

Each property requires a certain number of bits to encode its information. When only one bit is required, it means that the property can take one of two possible values, but not both. However, when the property can take some different values simultaneously, as many bits as the number of possible values are required. This only happens with the property (7).

4 Encoding examples

In this section, we give some examples that illustrate how the methodology works. Let us consider the following common attacks directed against different types of web servers and platforms. When applicable, problem descriptions are taken from the Common Vulnerabilities and Exposures (CVE), a list of standardized names for vulnerabilities and other information security exposures [36].

As the first example, let us consider the following attack:

```
GET /scripts/..%255c../winnt/system32/cmd.exe?/c+delete+/Q+.
```

According to CVE-2000-0884, IIS 4.0 and 5.0 allows remote attackers to read documents outside of the web root, and possibly execute arbitrary commands, via malformed URLs that contain UNICODE encoded characters, also known as the "Web Server Folder Traversal" vulnerability.

(1) The entry point is the web server's software (0), not the web application, since this vulnerability was present in all IIS machines, and can be exploited regardless of the web application on top.
(2) The attack exploits a canonicalization vulnerability in the web server software (1).
(3) The attack is deleting all files in the current directory in silence mode (`delete /Q .`), thus attacking the integrity of the file system (3).
(4) The attack deletes information from the server, thus the action is delete (2).
(5) The HTTP request has normal length (0).
(6) The HTTP Verb used is GET (0).
(7) The HTTP Header used is irrelevant for the attack (X).
(8) The target of attack is the web application (0) because the attacker is deleting the web pages stored on the server.
(9) The scope of the attack is global (1), because every user of the web service will be affected by it.
(10) The attacker gains access to the server's file system under the *IUSR_MachineName* identity, which corresponds to an unprivileged though dangerous user account in Windows machines. Thus, it is marked as 0.

As a result of the encoding process, the following vector is obtained:

Vector: {0, 1, 3, 4, 0, 0, X, 0, 1, 0}

As the second example, let us consider the following attack:

```
GET /product.jsp?id=10&title=<script>w=window.open(
'http://www.attacker.com/read.cgi?PAN='+document.forms[0].PAN.value);
w.close();</script>
```

The page blindly displays as its title the argument of a parameter passed in the URL. Hence, JavaScript code can be injected, opening the opportunity for a Cross-Site Scripting attack. In this case, the credit card number entered by the victim is sent to the attacker's server.

(1) The entry point is the web application (1) because the attack exploits a

lack of input validation to inject a script. This is a defect in the application, not in the web server itself.
(2) It is a cross-site scripting vulnerability, one of the possible code injection vulnerabilities (0).
(3) It is an attack against confidentiality (2) because the attacker is stealing some other user's credit card number.
(4) The attacker is reading information (0).
(5) The HTTP request has normal length (0).
(6) The HTTP Verb used is GET (0).
(7) The HTTP Header used is irrelevant for the attack (X).
(8) The target is the web application (0) because the attacker is obtaining data at the application level, not at the platform level. In this example, he is reading another user's personal information.
(9) The scope of this attack is local (0), because it affects only a user at a time.
(10) The attacker is not gaining access to the application, database or server, and, as a consequence, this category does not apply (X).

As a result of the encoding process, the following vector is obtained:

Vector: {1, 0, 2, 0, 0, 0, X, 0, 0, X}

As the third example, let us consider the following attack:

```
GET /prod.asp?id=1;exec xp_cmdshell 'net user bob h6q2 /add'--
```

The web application reads the input (the value of the parameter `id`) and passes it to the database engine, allowing for SQL injection attacks. The attacker exploits this vulnerability by cheating the application into executing a SQL Server extended procedure which executes any command passed as argument. In this example, the attacker adds himself to the OS users.

(1) The entry point is the web application (1) because the attack exploits poor input validation to inject SQL commands. This is a defect in the application, not in the web server itself.
(2) It is a SQL injection vulnerability, one of the possible code injection vulnerabilities (0).
(3) It is an attack against authorization (1) because the attacker is bypassing the administrator's configuration to add himself to the system's users list (`net user bob h6q2 /add`).
(4) The attacker is fabricating information because he is creating a new user account (3).
(5) The HTTP request has normal length (0).
(6) The HTTP Verb used is GET (0).
(7) The HTTP Header used is irrelevant for the attack (X).
(8) The target is the server platform (1) because the attacker has added

himself to the operating system's users list. The attack is not directed against the web application. Instead, it exploits a vulnerability in the application to gain access to the underlying operating system.

(9) This attack has no direct effect on the web users. As a consequence it is marked as X.

(10) The extended stored procedure `xp_cmdshell` is running under the administrator account's identity, and as a consequence the attacker gains administrative access (1).

As a result of the encoding process, the following vector is obtained:

Vector: {1, 0, 1, 3, 0, 0, X, 1, X, 1}

As the last example, let us consider the following attack:

```
GET /dir/[../](repeated approx 1344 times)
```

According to CVE-2001-0252, iPlanet Enterprise Server 4.1 allows remote attackers to cause a denial of service via a long HTTP GET request that contains many "/../" (dot dot) sequences.

(1) The entry point is the web server software (0) because the attack exploits a problem in the web server itself. This problem is shared by all servers of the same version, regardless of the web application running on top.

(2) It is an overflow attack, because a very long sequence is used, much longer than expected by the server software (3).

(3) The attack is directed against availability (4) because the server stops functioning after the attack.

(4) The attacker is interrupting the normal operation of the service (7).

(5) The HTTP request is unusually long (1).

(6) The HTTP Verb used is GET (0).

(7) The HTTP Header used is irrelevant for the attack (X).

(8) The target is the web application (0) because the attacker is not obtaining any access over the underlying server platform, but limits to disrupt the normal operation of the web application.

(9) This attack affects all users (1), since they will not be able to access the service.

(10) The attacker is not gaining access to the application, database or server, and, as a consequence, this category does not apply (X).

As a result of the encoding process, the following vector is obtained:

Vector: {0, 3, 4, 7, 1, 0, X, 0, 1, X}

In the ensuing process of encoding, the letters X are omitted, leaving different-length vectors. In such a way, memory is saved.

5 Coverage of real attacks space

The space of real attacks is unlimited. On the one hand, new vulnerabilities are discovered every day. On the other hand, there exist infinite variations of some attacks such as SQL injection, buffer overflow or cookie manipulation.

In order to quantitatively test how large a fraction of new vulnerabilities the taxonomy covers, all CVE reported web attacks against Microsoft's Internet Information Server (one of the most popular web servers nowadays) have been successfully encoded and classified. For the sake of completeness, some other vulnerabilities affecting other web servers (iPlanet, Apache, Oracle, BEA) have been randomly chosen from the CVE database, encoded and classified.

Let us now consider the type of attacks with infinite number of instances. We explain in detail the SQL injection attack, but similar reasoning can be applied to other types of attacks with infinite number of instances, such as buffer overflow, cross-site scripting, cookie manipulation, etc. When the SQL sentence varies, some categories in the encoding of the SQL injection attack remain the same (Entry point, Vulnerability, Length, HTTP Verb, HTTP Headers). The categories that are changed can sustain the variability of the SQL language (Service under threat, Action, Target, Scope, Privileges), since it is possible to semantically recognize SQL sentences, and the category Action of our taxonomy is exhaustive. Thus, any different SQL sentence embedded in the SQL injection attack would be comprised by our taxonomy affecting the already mentioned categories, which means that the proposed taxonomy is exhaustive in this case.

6 Possible applications

This taxonomy and the corresponding attack encoding vectors are useful in a number of applications, especially in intrusion detection systems and in application-level firewalls.

6.1 Intrusion Detection Systems (IDS)

An intrusion detection system (IDS) detects and reports attempts to misuse or break into networked computer systems in real time [23]. A traditional, non-heuristically based IDS consists of three functional components:

- A monitoring component, such as a packet capturer, which collects traffic data.

- An inference component, which analyzes the captured data to determine whether it corresponds to normal activity or malicious activity.
- An alerting component, which generates a response when an attack has been detected. This response can be passive (such as writing an entry in an event log) or active (such as changing configuration rules in the firewall to block the attacker's IP address).

One of the biggest problems faced by these systems is the huge amount of alerts that might be generated in a heavily-attacked environment in a matter of hours. It is impossible for a human operator to analyze so many reports and decide on the severity of the detected attacks to determine the action to take. This taxonomy can be used in the following way: first, the attacks are encoded by means of the proposed encoding scheme. Next, the vectors originated from the encoder are processed using pattern recognition or information extraction techniques (clustering algorithms [24], supervised learning [13], etc.) in order to pinpoint the most dangerous attacks, and analyze attack trends throughout time.

6.2 Application-level firewall

Another approach to detect and prevent web attacks consists of using an application-level firewall or, more specifically, web application firewalls [2,19]. A traditional firewall provides protection only at the network level, with minimal or no application awareness [35]. On the other hand, application-level firewalls are capable of processing data at the application level as well as decrypting SSL connections. An application-layer solution works within the application that it is protecting, inspecting requests as they come in from the network level. If at any point a possible attack is detected, it can take over and prevent unauthorized access and/or damage to the web server by simply blocking and logging the offensive request.

This approach might work better than the IDS one, blocking many more attacks. But again the problem of how to postprocess the alerts log file arises. Classifying the attacks once they have been blocked by the firewall and deciding on their severity is crucial for a prompt and effective response. This taxonomy helps in this task, providing an exhaustive group of mutually exclusive categories under which the attacks can be unambiguously classified using methods that employ the encoding scheme proposed in this paper [24]. Using encoder data as input, a decision system can determine the priority of various attacks.

7 Conclusion

In this paper, a taxonomy of web attacks is proposed that intends to represent a forward step towards a more precise reference framework. A web attack life cycle is defined as its base, to make it structured and logical. The properties of the most common web attacks are described. A new semantic-dependent encoding scheme for these attacks is defined that can be exploited by the known pattern recognition algorithms employing special distance measures, which reduces the time and space complexity of attacks data processing. Real world examples of attacks against different platforms, web servers, and applications are given to illustrate how this taxonomy and the encoding scheme can be applied. Finally, possible applications are described, such as intrusion detection systems and application firewalls.

Acknowledgements

This research was supported by Ministerio de Ciencia y Tecnología, Proyecto TIC2001-0586.

References

[1] N. Abramson. *Information Theory and Coding*. McGraw-Hill, 1963.

[2] Gonzalo Álvarez and Slobodan Petrović. Anomaly-based web attack detection system. Submitted to 6th Information Security Conference (ISC'03), 2003.

[3] Edward G. Amoroso. *Fundamentals of Computer Security Technology*. Prentice-Hall PTR, 1994.

[4] Chris Anley. Advanced sql injection in sql server applications. Technical report, Next Generation Security Software, January 2002.

[5] Chris Anley. (more) advanced sql injection. Technical report, Next Generation Security Software, June 2002.

[6] Cesar Cerrudo. Manipulating microsoft sql server using sql injection. Technical report, Application Security, Inc., 2002.

[7] Frederick B. Cohen. Information system attacks: A preliminary classification scheme. *Computers and Security*, 16(1):29–46, 1997.

[8] Crispin Cowan, Perry Wagle, Calton Pu, Steve Beattie, and Jonathan Walpole. Buffer overflows: Attacks and defenses for the vulnerability of the decade. *DARPA Information Survivability Conference and Exposition*, 2:1119–1129, January 2000.

[9] P. Elias. Universal codeword sets and representation of the integers. *IEEE Trans. Inform. Theory 21,2*, March 1975.

[10] N. Faller. An adaptive system for data compression. In *Record of the 7th Asilomar Conf. on Circuits, Systems and Computers*, 1973.

[11] R. M. Fano. *Transmission of Information*. MIT Press, 1949.

[12] Warwick Ford. *Computer Communications Security*, chapter 2. Prentice Hall PTR, 1994.

[13] Jeremy Frank. Artificial intelligence and intrusion detection: Current and future directions. In *Proceedings of the 17th National Computer Security Conference*, Baltimore, MD, 1994.

[14] John D. Howard and Thomas A. Longstaff. A common language for computer security incidents. Technical Report SAND98-8667, Sandia National Laboratories, October 1998.

[15] Michael Howard and David LeBlanc. *Writing Secure Code*, chapter 12. Microsoft Press, 2001.

[16] D. A. Huffman. A method for the construction of minimum-redundancy codes. *Proc. IRE 40,9*, September 1952.

[17] Mitja Kolsek. Session fixation vulnerability in web-based applications. Technical report, Acros Security, 2002.

[18] Ulf Lindqvist and Erland Jonsson. How to systematically classify computer security intrusions. In *Proceedings of the 1997 IEEE Symposium on Security & Privacy*, pages 154–163, May 1997.

[19] Pete Lindstrom. Guide to intrusion prevention. *Information Security*, October 2002.

[20] Daniel Lowry Lough. *A Taxonomy of Computer Attacks with Applications to Wireless Networks*. PhD thesis, Virginia Polytechnic Institute and State University, April 2001.

[21] Stuart McClure, Saumil Shah, and Shreeraj Shah. *Web Hacking*. Addison Wesley Professional, 2002.

[22] Microsoft. Cross-site scripting security exposure executive summary. http://www.microsoft.com/technet/security/topics/ExSumCS.asp, 2000.

[23] Stephen Northcutt. *Network Intrusion Detection, Third Edition*. New Riders Publishing, 2002.

[24] Slobodan Petrović and Gonzalo Álvarez. A method for clustering different length vectors using edit distance. http://arXiv.org/abs/cs.IR/0304007, 2003.

[25] Slobodan Petrović. Clustering unequal length binary data using graph-theoretic techniques. In *Proc. 4th Balkan Conference on Operational Research*, 1997.

[26] Michael Purser. *Secure Data Networking*, chapter 1. Artech House, 1993.

[27] Thomas Winfred Richardson. *The Development of a Database Taxonomy of Vulnerabilities to Support the Study of Denial of Service Attacks*. PhD thesis, Iowa State University, 2001.

[28] Joel Scambray and Mike Shema. *Hacking Exposed Web Applications*. McGraw-Hill Osborne Media, 2002.

[29] Bruce Schneier. Attack trees: Modeling security threats. *Dr. Dobb's Journal*, 12(24):21–29, December 1999.

[30] David Scott and Richard Sharp. Abstracting application-level web security. In *WWW2002*, May 2002.

[31] C. E. Shannon and W. Weaver. *The Mathematical Theory of Communication*. University of Illinois Press, 1949.

[32] William Stallings. *Network and Internetwork Security*, chapter 1. Prentice Hall, IEEE Press, 1995.

[33] J. S. Vitter. Design and analysis of dynamic huffman codes. *J. ACM 34,4*, October 1987.

[34] J. Ziv and A. Lempel. A universal algorithm for sequential data compression. *IEEE Trans. Inform. Theory 23,3*, May 1977.

[35] Elizabeth D. Zwicky, Simon Cooper, D. Brent Chapman, and Deborah Russell. *Building Internet Firewalls (2nd Edition)*. O'Reilly & Associates, 2000.

[36] Common vulnerabilities and exposures. http://cve.mitre.org.

[37] Nikto. http://www.cirt.net/code/nikto.shtml.

[38] Owasp. http://www.owasp.org.

[39] Teleport pro. http://www.tenmax.com/teleport/.

[40] wget. http://www.gnu.org/software/wget/wget.html.

[41] Whisker. http://www.wiretrip.net/rfp/p/doc.asp/i2/d21.htm.

Figures

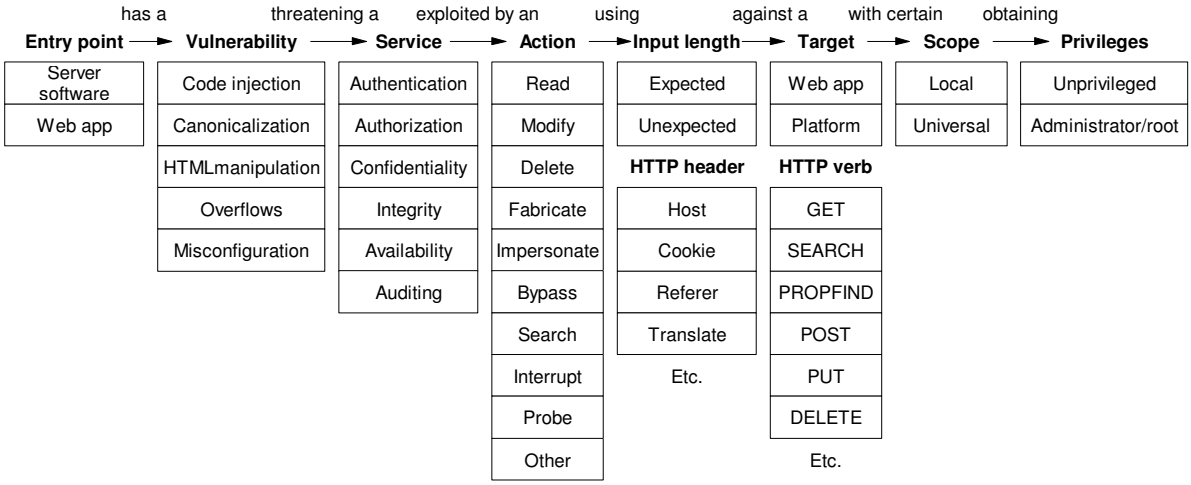

Fig. 1. Taxonomy of web attacks.