Analysis of web-related threats in ten years of logs from a scientific portal

Rafael D. C. Santos[a,b], André R. A. Grégio[c], Jordan Raddick[b], Vamsi Vattki[b] and Alex Szalay[b]

[a]National Institute for Space Research, Avenida dos Astronautas, 1758
São José dos Campos, São Paulo, Brazil;
[b]Johns Hopkins University, 3400 N. Charles Street, Baltimore, MD, USA;
[c]Renato Archer Information Technology Research Center, Rodovia Dom Pedro I, km 143.6
Bairro Amarais, Campinas, São Paulo, Brazil;

ABSTRACT

SkyServer is an Internet portal to data from the Sloan Digital Sky Survey, the largest online archive of astronomy data in the world. provides free access to hundreds of millions of celestial objects for science, education and outreach purposes. Logs of accesses to SkyServer comprise around 930 million hits, 140 million web services accesses and 170 million SQL submitted queries, collected over the past 10 years. These logs also contain indications of compromise attempts on the servers. In this paper, we show some threats that were detected in ten years of stored logs, and compare them with known threats in those years. Also, we present an analysis of the evolution of those threats over these years.

Keywords: Web Log Analysis, Security, Threat Analysis

1. INTRODUCTION

The World Wide Web has been used for more than twenty years to disseminate and collect information for many purposes: social interaction, scientific data, electronic commerce, etc. Because of their inherent openness, sites and portals on the WWW may be subject to abuse in several different ways, from unintentional misuse to deliberate attempts of intrusion to steal data, gain control, change site contents, or simply hinder its working.

Sites and portals collect, through logs or records, all user activities and system interactions. These logs may be used for resource monitoring and optimization, user profiling and security analysis.[1,2] While monitoring the collected logs may not protect the web sites per se (since they reflect actions already performed), analyses of these logs can help identify security problems on the server infrastructure (usually software-related), and potentially suspicious user activity.

The SkyServer portal is an open Internet portal to astronomy information, which contains data from a survey that maps one-quarter of the entire sky, containing detailed information on hundreds of millions of celestial objects. SkyServer provides free access to all SDSS data in several different ways, supporting astronomy research, public outreach and education, and allowing access through web forms, services and SQL queries. The SkyServer portal logs contains information about its more than ten years of operation (since April 2001).

As expected, these logs contain several entries that indicate some intrusion or compromise attempts on the servers. Since Skyserver is a scientific and education portal, the analysis of the threats it is exposed to may yield interesting information on general and directed attack attempts on this type of portal.

In this paper, we describe which security-related events were detected in ten years of logs on the SkyServer portal, comparing events found on the logs with threats expected to be found on the Web in different years. We use the Open Web Application Security Project (OWASP) list of Top Ten risks associated with web applications for those years, and presenting analysis on the evolution of the threats on those ten years.

Send correspondence to R. Santos (rafael.santos@inpe.br)

Cyber Sensing 2012, edited by Igor V. Ternovskiy, Peter Chin, Proc. of SPIE Vol. 8408,
84080H · © 2012 SPIE · CCC code: 0277-786X/12/$18 · doi: 10.1117/12.919545

The paper is organized as follows: Section 2 describes the SkyServer portal, its functions, and which types of logs are collected from the portal. Section 3 describes and comments on the most relevant Web application security risks as described by the OWASP (Open Web Application Security Project) Top Ten project for the years 2004, 2007 and 2010. In Section 4, we select some of the most relevant threats described in Section 3 and analyze the SkyServer portal logs searching for evidence of these threats; and whenever it is applicable we describe our findings and some comments. Finally, Section 5 presents the conclusions for this paper and possible future enhancements.

2. THE SKYSERVER PORTAL

The Sloan Digital Sky Survey (SDSS) is an effort to make a map of the universe. The project has released all of its data, free of charge to the pubilc - images and catalog data for nearly half a billion stars and galaxies. The multi-Terabyte SDSS dataset is the largest archive of digital astronomy data ever released.

SDSS data is accessible through two web portals: the Science Archive Server (SAS; `http://data.sdss3.org/`) provides access to binary image and spectra data as flat files, while the Catalog Archive Server (CAS) provides distilled science parameters through advanced query interfaces. The CAS is a collection of SQL server databases, each storing a particular release of SDSS data. This paper is concerned with catalog data and the CAS.

SDSS offers a number of HTTP, SQL, and SOAP interfaces to its catalog data, through the mechanism of the CAS. These interfaces are organized into several sites that offer access tailored to serve various audiences. It should be emphasized, however, that all interfaces provide access to the same SDSS data. The web sites are:

SkyServer (`skyserver.sdss3.org`; prior datasets at `cas.sdss.org`): a public Website offering access to the SDSS data, with documentation. SkyServer offers several visual interfaces for browsing SDSS data, and several form-based queries to search for sky objects by position, brightness, and a variety of other criteria. It also offers a raw SQL interface and tutorials for learning SQL, giving novices a gentle introduction to SQL in the context of astronomy through the SDSS.

Astronomer portal to SkyServer: formerly, SkyServer offered separate websites, hosted on separate servers, for astronomers. These sites had more generous query limits than the main SkyServer site. But by Data Release 7, CasJobs (see below) had become the primary interface for professional astronomers, so the astronomer portal was dropped for Data Release 8. Hits to the astronomer portal are in the weblogs, and are included in this analysis.

Collaboration portal to SkyServer: SkyServer also offered a separate access point for members of the SDSS collaboration, which allowed collaboration members access to data releases before they are available to the public.

CasJobs (batch jobs for the CAS; `http://skyservice.pha.jhu.edu/casjobs`): an asynchronous job interface to the CAS. CasJobs is a public Web service that allows users to create a personal database (MyDB), into which they can select SDSS data, upload their own, and cross-match either with a variety of datasets. CasJobs has become the default interface to SDSS data for the research community.

Virtual Observatory (`www.usvao.org`): a collection of web services being developed by the astronomy community as part of their efforts to create seamless access to astronomy data in all wavelengths of the electromagnetic spectrum.

Galaxy Zoo (`www.galaxyzoo.org`): a "citizen science" website in which online volunteers look at images of SDSS galaxies and classify them by shape. Galaxy Zoo has been actively collecting classifications since July 2007; those classifications up to September 2008 are included in the final Galaxy Zoo data release. The final data release includes classifications of about 900,000 galaxies – an average of 74 classifications per galaxy. The result was a classification by shape of each galaxy; these classifications are included in SDSS Data Release 8, where they can be searched and matched with other SDSS data using SkyServer or CasJobs.

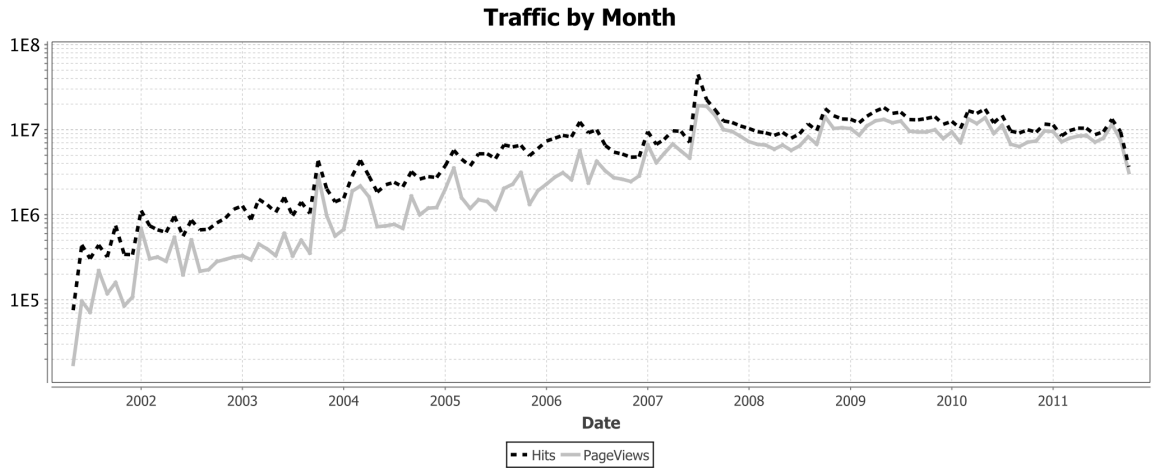

Figure 1. Aggregate traffic: number of hits and pageviews per month.

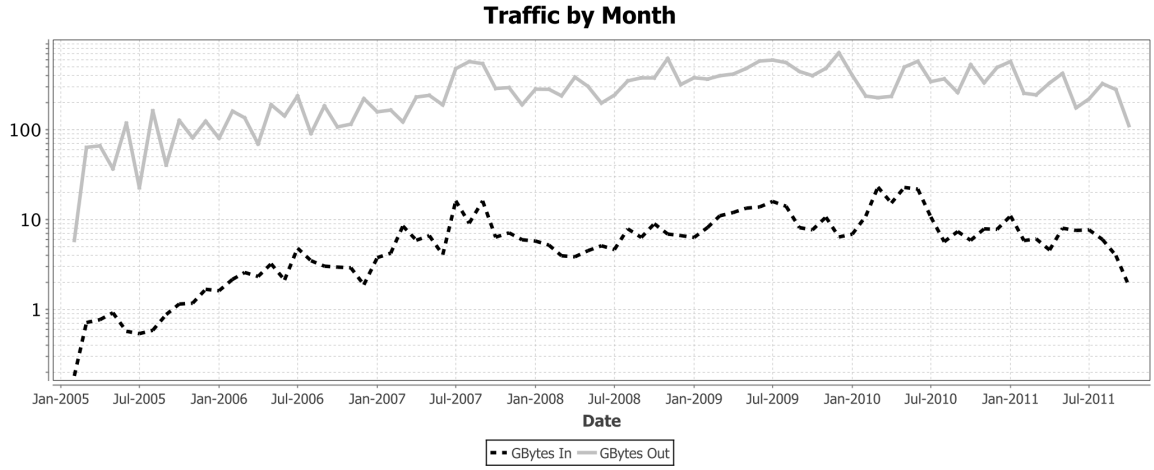

Figure 2. Aggregate traffic: number of gigabytes in and out of the portal.

Logs of access to SkyServer and related tools have been collected continuously for more than ten years, and contain entries on more than 930 million web hits, 140 million web services accesses and 170 million SQL submitted queries. To facilitate the creation of queries and tools that analyze these logs, they were preprocessed and stored in a SQL Server database at Johns Hopkins University.

Presently, these logs are being analyzed in order to characterize the Skyserver users (human and bots) and which tools and services they use most. Some publications and reports already deal with different aspects of user characterization[3–6] using the first five years of logs.

In this work, we will use a larger collection of logs that encompasses more than ten years of web activity, from April 2001 to October 2010 (with some gaps in specific logs, mentioned where applicable). To give an idea of how much data are in these logs, Figure 1 shows the number of hits (requests) and page views (requests for content pages, determined primarily by page extensions such as .asp or .html) that were served by the portal in these ten years. Figure 2 shows the portal's traffic in Gigabytes (however, note that traffic volume in bytes was stored only from February 2005 on). It must be pointed out that most of this traffic comes from web bots and applications, not from human users.

3. THE OWASP TOP TEN PROJECT

Web application security issues can be basically divided into server-side and client-side issues.[7] Each type, or side, has its own vulnerabilities that depend on the application being used, its settings and policies, the level of hardening applied to the host system and the external access provided. Moreover, there are flaws that are common to both server- and client-sides. To address the current most critical web application vulnerabilities, The Open Web Application Security Project (OWASP), produced a document, *OWASP Top 10 Application Security Risks*[8] that aims on rising awareness about web security flaws and their potential risks. This document is reevaluated from time to time, and there are three versions of it so far, covering the study up to 2004, 2007 and 2010.

The risk estimative, which defines the Top 10 ranking for each one of the vulnerabilities, is calculated based on the OWASP Risk Rating Methodology,[9] which states that the risk is proportional to the likelihood of a threat multiplied by its impact, and considers several factors to calculate the likelihood and impact, such as the attack vector, the weakness prevalence and detectability, and the technical impact. This leads to an overall business impact and, consequently, to an overall risk severity that can be low, medium or high.

Table 1 shows which vulnerabilities were assessed on each OWASP Top 10 list (2004, 2007 and 2010). In this study we consider the entries on Table 1 – we describe the more recent Top 10's threats (2010), select the most potentially interesting vulnerabilities (based on the possibility of their occurrence on our servers and of their search in our log database) and explain below the associated risks for the portal being analyzed.

Table 1. OWASP Top 10 threats up to 2010, 2007 and 2004. The identifier between parenthesis (e.g., #1) indicates the order of the threat related to the Top 10's publication year. A "-" means that the threat was not recorded or considered in that year.

2010	**2007**	**2004**
Injection (#1)	Injection Flaws (#2)	Injection Flaws (#6)
Cross-Site Scripting - XSS (#2)	Cross-Site Scripting - XSS (#1)	Cross-Site Scripting - XSS (#4)
Broken Authentication and Session Management (#3)	Broken Authentication and Session Management (#7)	Broken Authentication and Session Management (#3)
Insecure Direct Object References (#4)	Insecure Direct Object References (#4)	-
Cross-Site Request Forgery - CSRF (#5)	Cross-Site Request Forgery - CSRF (#5)	-
Security Misconfiguration (#6)	-	Insecure Configuration Management (#10)
Insecure Cryptographic Storage (#7)	Insecure Cryptographic Storage (#8)	Insecure Storage (#8)
Failure to Restrict URL Access (#8)	Failure to Restrict URL Access (#10)	Broken Access Control (#2)
Insufficient Transport Layer Protection (#9)	Insecure Communications (#9)	-
Unvalidated Redirects and Forwards (#10)	-	Unvalidated Input (#1)
-	Malicious File Execution (#3)	-
-	Information Leakage and Improper Error Handling (#6)	Improper Error Handling (#7)
-	-	Buffer Overflow (#5)
-	-	Application Denial of Service (#9)

3.1 Web Threats

Web-related attacks usually target the server, by exploiting unvalidated inputs, or the browser, by tricking the user or subverting a plugin. In this section, we describe each threat/vulnerability that is part of the OWASP Top 10 from 2010, as those are representative for the actual situation and, at the same time, most of them reflect the trends in past years (see Table 1).

3.1.1 Injection

Injections, in this case, can be defined as the sending of untrusted (or tricky) data to a server in order to execute arbitrary commands or to access sensitive information. This data is usually sent as part of a command or a query, such as a SQL query. For example, lets suppose an application that builds the following vulnerable SQL query:

```
String vquery = \SELECT * FROM credentials WHERE userID="' + request.getParameter(\id") +"'";
```

An attacker can modify the "id" parameter to match a desired condition, such as `‘ or ‘1’=’1`, leading to the exposure of all the information from the credentials table. We searched for injection attempts on the logged SQL queries and present the results in Section 4.1.

3.1.2 Cross-Site Scripting (XSS)

Cross-site scripting attacks occur when an unvalidated user-supplied data is sent to an application that does not validate or escape its content, causing this data to exploit the interpreter in the browser. An attacker can modify a parameter in the user's browser so as to execute a malicious script from another site to, for example, steal the user's session ID and hijack the session. XSS is caused mainly by unvalidated or untrusted input data and the attack is performed in the user's browsers. However, it is possible to find clues of XSS attacks on the server logs, searching for external sites inside script tags or javascript code being sent as GET method parameters. Common XSS patterns* include:

```
<SCRIPT SRC="http://attacker.site"></SCRIPT>

<IMG SRC="javascript:alert('XSS');"> (this can also be formatted without double
quotes and semicolon)

With embedded encoded tab:
<IMG SRC="jav\&\#x0A;ascript:alert('XSS');">

With embedded encoded CR:
<IMG SRC="jav\&\#x0D;ascript:alert('XSS');">

Malformed IMG tag:
<IMG """><SCRIPT>alert("XSS")</SCRIPT>">
```

User-generated content is the a common vector for XSS attacks. User-generated content is not allowed in the SkyServer portal, nonetheless the logs show some attempts to execute remote code. Results are presented in Section 4.2.

*More complex XSS patterns can be found in `http://ha.ckers.org/xss.html`

3.1.3 Broken Authentication and Session Management

A misconfigured server or flaws in the authentication or session management, such as unprotected or unencrypted credentials, or identifiers that are exposed in the URL, allow that attackers steal accounts, recover password information or rewrite URLs to access sensitive data in an unauthenticated mode. A way to verify for the presence of this kind of vulnerability is to search for parameters (e.g., user or session IDs) being passed in the URL:

```
https://myportal.org/app/login;jsessionid=<HASH>?type=users
```

In order to have success, attackers must know enough about the internal workings of websites and portals. This type of attack cannot be easily detected by the HTTP logs alone, and due to its lack of risk (for our purtal's purposes) was not investigated.

3.1.4 Insecure Direct Object References

This vulnerability implies that the access to an unauthorized object is granted to an authorized user by the change of a parameter value that refers to this object. It happens due the lack of access control checks and can be automated by an attacker, causing the exposure of sensitive data. For instance, in the example below, a malicious user can script a brute-force tool to vary the `<NUMBER>` parameter and directly access objects that he is not authorized to do it:

```
https://myportal.org/app/Object.do?action=executeOpen\&param=<NUMBER>
```

Attackers explore this type of vulnerability using fuzz testing, trial-and-error, by directly reading contents of pages (forms and results) or by combination of those. Since in our portal there are several valid reasons to have users changing URL parameters (and there are no change on parameters that could lead to access to sensitive data) we did not further investigate this attack attempts.

3.1.5 Cross-Site Request Forgery (CSRF)

Cross-site request forgery is a kind of attack in which a user is tricked to submit a forged HTTP request to an attackers site. If there is no unique and unpredictable token for each user session and the user is authenticated, the attack is successful and the attacker can change or perform any function at the same level of access of the victim. For instance, if the user is logged in a Web site that allows settings changes to be done via HTML forms, an attacker can subvert this by inducing the user to request some change through the modification of these form's parameters. This way, an attacker can for example create an admin account in the system, change the user's password or obtain his credit card number.

For this attack to work the attackers must create a login or information change form in order to get a logged-in user to reenter his/her information in a third-party (the attackers') site, effectively getting the users' information. Traces for this kind of attack does not appear on http logs, therefore were not analyzed in this paper, besides this kind of attack wasn't considered a serious security risk for our portal.

3.1.6 Security Misconfiguration

This is an easily exploitable vulnerability, as it involves attacks to several levels of a Web application stack, such as the platform, server or custom code available. Also, a misconfigured system can be compromised by external attackers or users with valid account, as well as insiders. Attacks against vulnerable systems can be done through the access to default accounts or unprotected files and directories. An example of attack aiming to exploit misconfigured systems is related to the Nimda worm. It tries to compromise a system traversing the server site directories in order to access applications that can result in the execution of arbitrary code.

There were several attempts to exploit this vulnerability recorded in our logs. Some analysis of these are presented in section 4.3.

3.1.7 Insecure Cryptographic Storage

This vulnerability causes exposure of data that can be sensitive and is commonly exploited due to the lack of strong encryption (in some cases, the lack of encryption at all), the use of weak hash algorithms to protect passwords, the unsafe generation and storage of cryptographic keys, the insufficient length of those keys or the use of a poor cryptographic implementation.

This threat was not considered in this study because it would require the access to the servers' settings or the knowledge about the transmission of sensitive but badly encrypted information between server and clients, which are not registered in the logs.

3.1.8 Failure to Restrict URL Access

The "Failure to Restrict URL Access" vulnerability is very similar to "Insecure Direct Object References". The key point here is that a page that supposedly should require authentication to be accessed is accessible in an unauthenticated way if a user simply forces the browser to target the desired URL. For instance, an attacker that is not authenticated or a user that is not authorized to access an administrative page exploit this vulnerability if they can get to this restricted page. A legitimate access would be like the one below, in which the page could be accessed even anonymously by any external user:

```
http://myportal.org/app/general-page.html
```

On the other hand, if the vulnerability is present, a page that should not be accessible to any user (unless those with administrative privileges) would then be accessed:

```
http://myportal.org/app/administrative-page.html
```

Since there are no "hidden" pages on the portal that could be accessed without credentials, this analysis was not done.

3.1.9 Insufficient Transport Layer Protection

Improper transport layer protection, such as the use of SSL, can allow the monitoring of the network traffic, exposing users information. Even if SSL is used, outdated versions or those configured to accept weak algorithms (e.g., to maintain compatibility with older browsers) can still allow the can still allow the clear text recovery and consequently, the exposure of sensitive data. This kind of threat cannot be detected by log analysis.

3.1.10 Unvalidated Redirects and Forwards

When a Web application forwards users requests to other sites or internal pages, an attacker can craft a malicious URL that install malicious programs and then redirect the user to there. One way to search for this kind of attacks, it is necessary to verify the logs for external URLs that should not be there, as shown below:

```
http://myportal.org/redirect.jsp?url=attacker.site
```

In order to exploit the vulnerability the attacker must identify which functions in the portal allow the forwarding or redirection. Since the portal does not contains these functions this attack was not investigated.

4. ANALYSIS AND RESULTS

To identify compromise attempts to the portal, we have searched our logs for occurrences of patterns that are usual in threats such as injections, XSS or security misconfiguration. Those attacks, if successful, are very dangerous to the integrity of the portal, although we must point that in more than 10 years of operation the system administrators of our portal never found any trace of successful invasion or compromise attempts.

In the following sections we describe some of our findings that are relevant to illustrate the attack attempts along the years.

4.1 SQL Injection Stored Procedures Attacks

SQL injections are one of the most common type of injection attacks. One way to perform an SQL injection is through taking advantage of a database feature called stored procedure. Stored procedures are prewritten queries that often treat the passed data as literal and, in the case of Microsoft SQL Server, can integrate the database into the operating system, allowing them to execute system commands.[7] A dangerous stored procedure available on Microsoft SQL Server is the `xp_cmdshell`, as it allows the execution of arbitrary commands on the server. A query that uses this stored procedure to shut the server down would be as follows:[10]

```
EXEC master..xp_cmdshell 'net stop sqlserver', no_output
```

Other stored procedures allow the manipulation of the registry (e.g., `xp_regwrite, xp_regdeletekey`), the access to the file system (e.g., `xp_subdirs, xp_fileexist, xp_runwebtask`), the termination of programs (e.g., `xp_terminate_process`), the gathering of user information (e.g., `xp_logininfo`) and even the sending of spam by an attacker (e.g., `xp_sendmail`).

Our logs contains all SQL queries executed in the portal databases, which includes both queries executed as part of a portal function (where we expect no attempts of injections) and queries directly submitted by users. The portal, by design, allows the direct manipulation of databases by users, even allowing, in some cases, the creation of stored procedures. In order to identify possible abuses of this feature we've filtered all entries in the SQL log that contains the patterns described above.

Most of the patterns for execution of stored procedures that allow execution of commands on the server were absent from the logs: from a total of more than 170 million SQL commands collected by the log, only 45 instances of attempted execution of the procedure `xp_cmdshell` were recorded. Figure 3 shows the number of daily attempts for this attack.

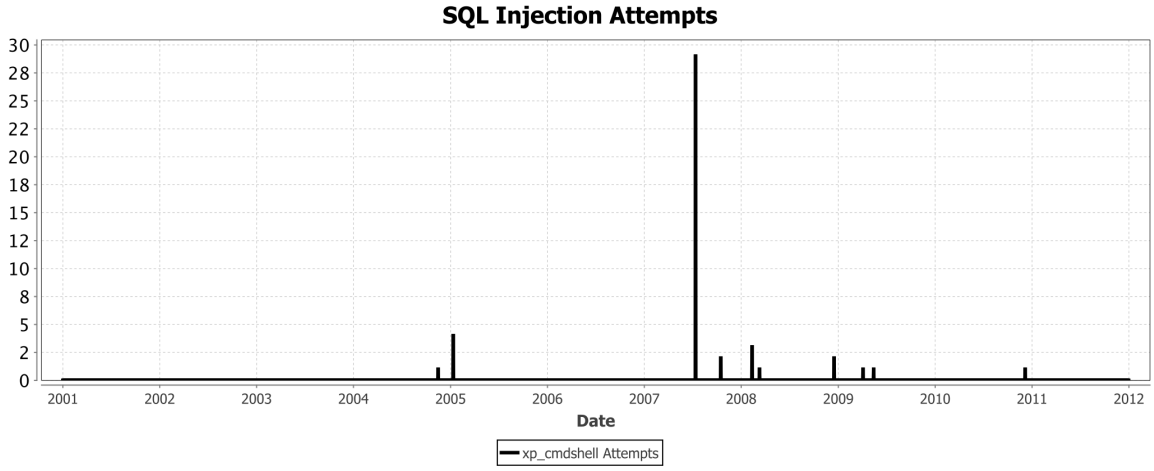

Figure 3. Command execution attempts in the SQL logs database.

Of the attacks of this type (with one notable exception, described later in this section), most were attempted by the same IP with few variations of the attempt, and apparently the attacker gave up on being unable to execute the command. The attacks were also well spaced in time, with several weeks or even months between one attack and other. One example of an attempt sequence is shown below:

```
exec MASTER..xp_cmdshell 'dir C:\ > jhu.txt & tftp -i 85.133.177.56 Put jhu.txt'
;exec MASTER..xp_cmdshell 'dir C:\ > jhu.txt & tftp -i 85.133.177.56 Put jhu.txt'
; exec MASTER..xp_cmdshell 'dir C:\ > jhu.txt & tftp -i 85.133.177.56 Put jhu.txt'
```

These three attempts were originated by the same IP with less than a minute of interval between the attempts. It seems the attacker was trying to test the possibility of command execution before running something more complex.

One notable exception to the attack patterns was one with 27 unique attempts in one day (July 13, 2007), originated from a single IP. Some of these unique attempts were repeated more than 10 times in a burst. These attacks looks like:

```
select name, type, description from DBObjects where  type in ('F'+'P') and UPPER(name)
  like '%PhotoType';use master dbcc addextendedproc('xp_cmdshell'+'xplog70.dll')--%'
select name, type, description from DBObjects where  type in ('F'+'P') and UPPER(name)
  like '%PhotoType';exec master.dbo.xp_cmdshell 'ecHo hÈåE%'
select name, type, description from DBObjects where  type in ('F'+'P') and UPPER(name)
  like '%PhotoType';DROP TABLE [X_2402];use master dbcc
  addextendedproc('xp_cmdshell'+'xplog70.dll')--%'
select name, type, description from DBObjects where  type in ('F'+'P') and UPPER(name)
  like '%PhotoType';exec master.dbo.xp_cmdshell 'del C:\down.exe'--%'
select name, type, description from DBObjects where  type in ('F'+'P') and UPPER(name)
  like '%PhotoType';CREATE TABLE [X_2402]([id] int NOT NULL IDENTITY (1+1)+ [ResultTxt]
  nvarchar(4000) NULL);insert into [X_2402](ResultTxt) exec master..xp_cmdshell
  'Dir C:\';insert into [X_2402] values ('g_over');exec master..sp_dropextendedproc
  'xp_cmdshell'--%'
```

Some of the entries on the log related to this attempt indicates a naïve attempt to re-enable the `xp_cmdshell` stored procedure† for further attack attempts (the SQL Server databases used in our portal were hardened by the removal of the `xp_cmdshell` stored procedure.) The frequency of the attempts (all occurred in less than two minutes) indicate the use of an automatic tool, but some of the SQL commands associated with the attempts reflect real features of the portal's database, so the attacker possibly knew about the portal when attempting the attack. We will further investigate this incident by correlating other logs, activity and information about that particular IP during the time of the attack.

4.2 Cross-Site Scripting Vulnerability

The improper validation of user-supplied input can incurr in successful Cross-Site Scripting (XSS) attacks. As an example, in 2005, a vulnerability related to improper input validation allowed attacker to exploit a XSS in a bulletin board application named YaBB (Yet Another Bulletin Board).[11] Nessus, a popular vulnerability scanner, has a plugin crafted to exploit this and identify vulnerable servers. If a server is vulnerable to this attack, an attacker could exploit it to steal cookie-based credentials. Attempts to use Nessus to find a server vulnerable to this XSS are logged as follows:

```
"action=usersrecentposts;username=<IFRAME%20SRC%3Djavascript:alert('Nessus%2Dwas%2Dhere')>
  <%252FIFRAME>"
```

Analysis of the logs shows several entries similar to that one, with several variations, but most attempting to execute a small snipped of JavaScript inside the page generated as response for the request corresponding to that entry.

Several of those entries appeared to be inoffensive (e.g. related to a function of the portal to display images in a pop-up window), but there were several attempts to find vulnerability as the one shown before or more obscure but recognizable attempts.

All entries in the log were separated into three categories: entries which clearly indicated some attempt to explore the vulnerability (usually by containing URLs for external suspicious resources); entries which could be

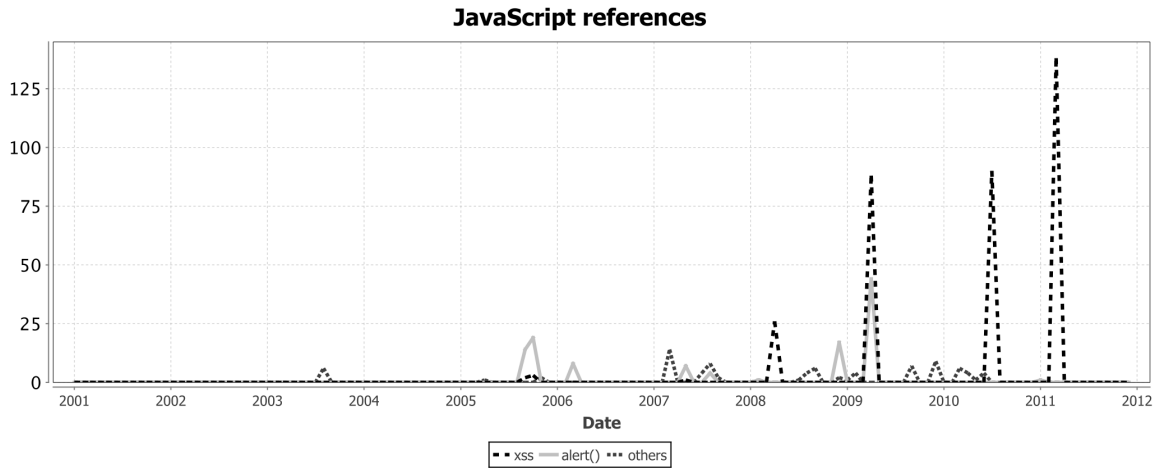

Figure 4. Command execution attempts in the SQL logs database.

automated but generic attempts (e.g. "nessus was here") and other, possibly inoffensive javascript snippets (not related to the portal's functions). The time plot for those categories is shown in Figure 4.

Figure 4 shows some bursts of activity possibly related with the XSS vulnerabilities. Some interesting points are:

- The "nessus was here" message appeared in some small bursts (less than 20 entries in a day) in between 2005 and 2009.
- There were several attempts to refer files in other servers. We've investigated some of those, but most of the servers appear to be inactive.
- Several entries contains the string "javascript:alert(cross_site_scripting.nasl);", and correspond to the peaks in April 2009, July 2010 and March 2011. We suspect that this is part of an automatic scan to verify whether the server is already compromised.

4.3 Security Misconfiguration Attack Attempt

There are several possibilities to attack a misconfigured server, such as to traverse the directories to try the access to system files or the execution of commands. This threat is better described in Section 3.1.6. In misconfigured Microsoft IIS servers, it is common that attackers or autonomous malicious code (e.g. worms) attempt to gain a shell – in this case, the command line prompt `cmd.exe` – to execute arbitrary commands on the server.

To identify this kind of compromise attempts we have searched in our logs for occurrences of URLs containing the strings `cmd.exe` or `root.exe`, as they can indicate an attack. Figure 5 shows the amount of entries on the log database which contains either the string `cmd.exe` or `root.exe`.

In Figure 5 we can see that most of the attack attempts were done almost as soon as the site was made public (April 2001) with a sharp increase in the middle of 2001, and most attack attempts ceased from the beginning of 2005 on. This is considered consistent with the appearance of worms that attempts to exploit vulnerabilities in Microsoft's IIS Server in the wild, e.g. NIMDA (September 2011), Code Red (July 2001), Sadmind (May 2001) and others.[12]

Figure 6 shows the same data as Figure 5 but considering only the log entries between October 2004 and February 2005 – around the time when attacks of this kind started to become rare. Most of the attack attempts were single attempts from an unique IP using several variants of the path to the `cmd.exe` or `root.exe` strings, as shown below:

[†]As described in `http://support.microsoft.com/kb/891984`.

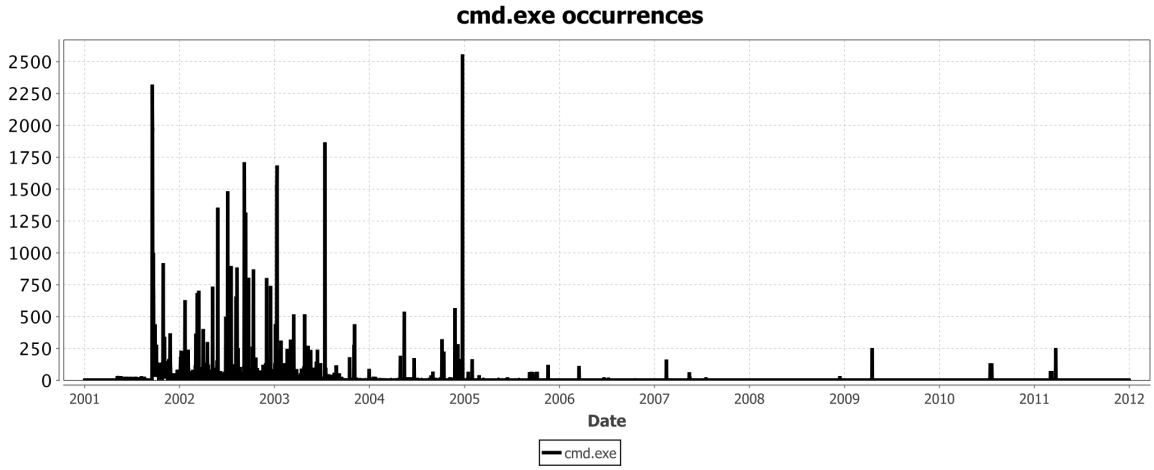

Figure 5. Number of occurrences of `cmd.exe` and `root.exe` in the http logs database.

```
/pbserver/..%5c..%5c..%5cwinnt/system32/cmd.exe
/msadc/..%5c..%5c..%5c..%5cwinnt/system32/cmd.exe
/cgi-bin/..%5c..%5c..%5c..%5c..%5c..%5cwinnt/system32/cmd.exe
/adsamples/root.exe
/d/winnt/system32/cmd.exe/c  /c+dir+c:\
/c/winnt/system32/cmd.exe  /c+dir+c:\
/iisadmpwd/root.exe  /c+dir+c:\
/cmd.exe  /c+dir+c:\
```

Also, Figure 6 shows a huge spike on Dec 24th, 2004: the logs indicate that the same attacker tried several variants of the attack in a short period of time, generating more than 2500 entries on the log in less than two hours.

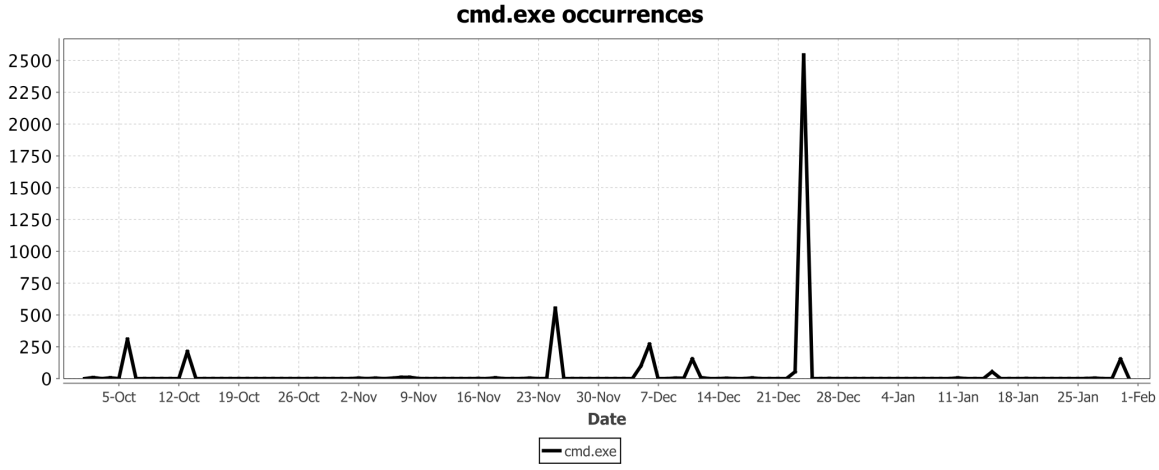

Figure 6. Number of occurrences of `cmd.exe` and `root.exe` in the logs database (late 2004/early 2005).

Interestingly, there are some small spikes on the plot shown in Figure 6 that points that there were some small-scale attack attempts even in 2011. The last three peaks on that plot corresponds to multiple attempts again with variations on the URLs, originated from the same computer and in a very short period of time. This

suggests that attackers were still searching for vulnerabilities using this approach even after patches were made available and the main worms were shut down.

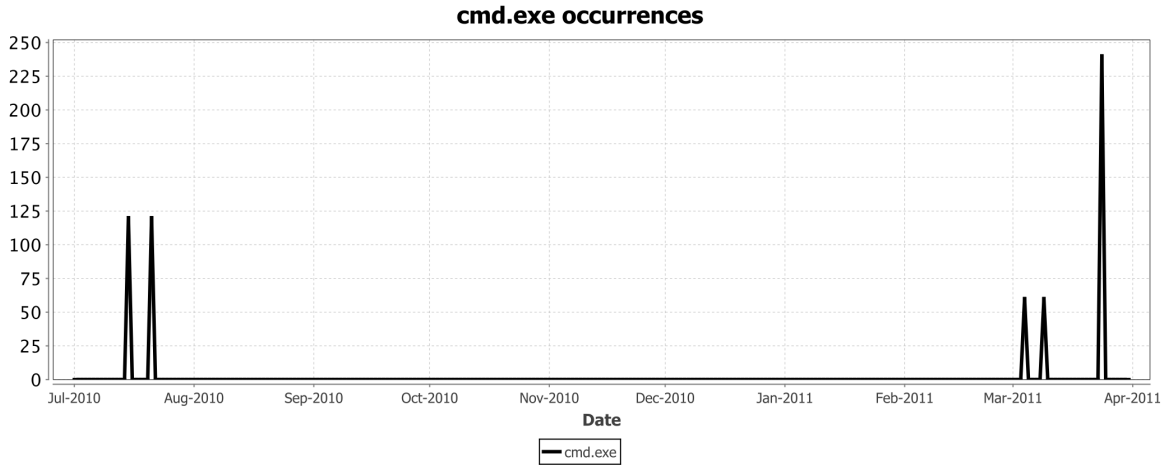

Figure 7. Number of occurrences of `cmd.exe` and `root.exe` in the logs database (late 2011).

5. CONCLUSIONS AND FUTURE WORK

This paper describes a work in progress – during the preparation of the logs and tools to analyze them we've identified some problems related with the collection of logs, representation in a SQL database and visualization issues that suggest that new methods and techniques may yield more detailed results. Nonetheless, the analysis and results contains an interesting view of ten years of logs and what we could find on them that's interesting from a systems security point of view.

Use of the OWASP list of top ten vulnerabilities served well as a guide to what we could find, but it is clear that most of the threats cannot be identified by logs alone, and some potential threats (e.g. distributed denial-of-service) that could have negative impacts on the portal are not considered as critical as they were almost ten years ago.

On the other hand, due to the nature of the portal, some threats (e.g. attempts to steal data) cannot be considered, since the data have no commercial value and the purpose of the portal is to distribute it. Perhaps due to the nature of the portal (scientific free data, very domain-specific) there were no serious directed attacks, and the sporadic attempts were not effective due to the security measures adopted by developers and IT staff.

We were able to identify some "friendly fire" – automated exploit attempts that were executed from within the same institution that hosts the logs. These are part of a normal scan for vulnerabilities.

Some problems were found when analyzing the logs: some data was not collected in the first years and some analysis tools and techniques proved hard to implement, especially when dealing with log entries that could be either part of the portal or an attempt to attack. Considering the analysis of logs for security purposes we suggest this list of best practices for future deployments:

- **Collect as much data as possible in the logs**, in order to be able to run queries and get comparable information for any time the portal was operating, and to precisely reconstruct user interaction.
- **Use a relational database** to store the data. Although this adds complexity to the log analysis system, the flexibility more than compensates for the extra work. Several queries related to this paper were composed and executed in a very short period of time, but would require complex text processing commands that would run longer if the logs were stored in their original form.

- Whenever possible, **use unique and clear identifiers** for pages and other objects, to facilitate separation between traffic that is part of the portal and exploit attempts. If possible use tokens as additional parameters to facilitate the reconstruction of navigation paths and sessions.
- Finally, **follow the OWASP suggestions and recommendations**, which are presented in detail in their web site.[8]

ACKNOWLEDGMENTS

This work was initiated and inspired by our late colleague, Jim Gray of Microsoft Research.

REFERENCES

[1] Grégio, A. R. A., Santos, R., and Montes, A., "Análise de logs: Abordagens tradicionais e por data mining," in [*Oitavo Simpósio Segurança em Informática – Anais*], (2006). (in Portuguese).

[2] Grégio, A. R. A. and Santos, R., "Análise e visualização de logs de segurança," in [*Computer on the Beach 2010: Livro de Minicursos*], da Rocha Fernandes, A. M., de Miranda, E. M., and Wangham, M. S., eds., 85–107 (2010). (in Portuguese).

[3] M. Wong, B. B. and Singh, R., "Characterization and analysis of usage patterns in large multimedia websites." `http://cs.sfsu.edu/techreports/reports_list.html` (2006). Technical Report TR-06.20, Computer Science Department, San Francisco State Univerity.

[4] B. Bhattarai, M. W. and Singh, R., "Multimodal usage visualization for large websites." `http://cs.sfsu.edu/techreports/reports_list.html` (2006). Technical Report TR-06.21, Computer Science Department, San Francisco State Univerity.

[5] Abdulla, G., "Analysis of SDSS SQL Server Log Files," UCRL-MI-215756-DRAFT. Lawrence Livermore National Laboratory (2005).

[6] Singh, V., Gray, J., Thakar, A. R., Szalay, A. S., Raddick, J., Borosky, B., Lebedeva, S., and Yanny, B., "Skyserver traffic report – the first five years." `http://research.microsoft.com/apps/pubs/default.aspx?id=64520` (2006). Microsoft Tech Report MSR-TR-2006-190.

[7] Andrews, M. and Whittaker, J. A., [*How to Break Web Software: Functional and Security Testing of Web Applications and Web Services*], Addison-Wesley Professional (2006).

[8] OWASP, "Owasp top ten project." `https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project` (2010). Access in March 2012.

[9] OWASP, "Owasp risk rating methodology." `https://www.owasp.org/index.php/OWASP_Risk_Rating_Methodology` (January 2012). Access in March 2012.

[10] Microsoft, "xp_cmdshell." `http://msdn.microsoft.com/en-us/library/aa260689(v=sql.80).aspx`. Access in March, 2012.

[11] The MITRE Corporation, "Cve-2005-0741." `http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2005-0741`. Access in March, 2012.

[12] Wikipedia, "Timeline of computer viruses and worms." `http://en.wikipedia.org/wiki/Timeline_of_notable_computer_viruses_and_worms` (2012). Access in March 2012.