
Master of Science ETH of Computer Science

Bojan Oliver Konic
8/15/2008

Supervisors:

Prof. Dr. S. Capkun, ETH Zürich

Nils Ole Tippenhauer, ETH Zürich

Peter Nold, Altran AG
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preface</td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>Abstract</td>
<td></td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>Introduction</td>
<td>7</td>
</tr>
<tr>
<td>3.1</td>
<td>Focus</td>
<td>8</td>
</tr>
<tr>
<td>4</td>
<td>Security implementation of SharePoint 2007</td>
<td>9</td>
</tr>
<tr>
<td>4.1</td>
<td>Introduction into SharePoint 2007</td>
<td>9</td>
</tr>
<tr>
<td>4.1.1</td>
<td>Components</td>
<td>9</td>
</tr>
<tr>
<td>4.1.2</td>
<td>Structure of a SharePoint installation</td>
<td>9</td>
</tr>
<tr>
<td>4.1.3</td>
<td>Event handlers</td>
<td>10</td>
</tr>
<tr>
<td>4.1.4</td>
<td>Web parts</td>
<td>10</td>
</tr>
<tr>
<td>4.1.5</td>
<td>Interfaces</td>
<td>11</td>
</tr>
<tr>
<td>4.2</td>
<td>SharePoint security</td>
<td>11</td>
</tr>
<tr>
<td>4.2.1</td>
<td>Code execution</td>
<td>11</td>
</tr>
<tr>
<td>4.2.2</td>
<td>Authentication</td>
<td>11</td>
</tr>
<tr>
<td>4.2.3</td>
<td>Zones</td>
<td>11</td>
</tr>
<tr>
<td>4.2.4</td>
<td>Access control</td>
<td>12</td>
</tr>
<tr>
<td>4.2.5</td>
<td>Role assignments</td>
<td>12</td>
</tr>
<tr>
<td>4.2.6</td>
<td>Scopes</td>
<td>13</td>
</tr>
<tr>
<td>4.2.7</td>
<td>Role definition compartments</td>
<td>13</td>
</tr>
<tr>
<td>5</td>
<td>Vulnerabilities &amp; risks</td>
<td>15</td>
</tr>
<tr>
<td>5.1</td>
<td>Introduction</td>
<td>15</td>
</tr>
<tr>
<td>5.2</td>
<td>Technical vs. logical vulnerabilities</td>
<td>16</td>
</tr>
<tr>
<td>5.3</td>
<td>Attacks</td>
<td>17</td>
</tr>
<tr>
<td>5.3.1</td>
<td>Collecting information about the target</td>
<td>18</td>
</tr>
<tr>
<td>5.3.2</td>
<td>Bypassing restrictions on inputs</td>
<td>23</td>
</tr>
<tr>
<td>5.3.3</td>
<td>State based attacks</td>
<td>24</td>
</tr>
<tr>
<td>5.3.4</td>
<td>Attacking user supplied input data</td>
<td>27</td>
</tr>
<tr>
<td>5.3.5</td>
<td>Language based attacks</td>
<td>29</td>
</tr>
<tr>
<td>5.3.6</td>
<td>Attacking the server</td>
<td>32</td>
</tr>
<tr>
<td>5.3.7</td>
<td>Breaking authentication</td>
<td>34</td>
</tr>
</tbody>
</table>
5.3.8 Attacks on web services ................................................................. 35
5.4 Testing web applications ................................................................. 38
5.5 Testing the bare system ................................................................. 38
  5.5.1 The bare system ....................................................................... 38
  5.5.2 The testing tool ....................................................................... 38
  5.5.3 First run: testing the bare system as administrator .................... 39
5.6 Testing the productive system ......................................................... 40
  5.6.1 The productive system ............................................................. 40
  5.6.2 Building a copy of the live system ........................................... 40
  5.6.3 First run: anonymous user on public web ................................ 42
  5.6.4 Second run: site admin user on public web ............................ 43
  5.6.5 Third run: site admin user on admin web ............................... 44
  5.6.6 Findings .................................................................................. 45
5.7 Logical vulnerabilities ....................................................................... 46
  5.7.1 Lack of overview ..................................................................... 47
  5.7.2 Lack of maintainability ............................................................ 47
5.8 Administration tools ....................................................................... 48
  5.8.1 STSADM.EXE ......................................................................... 48
  5.8.2 PSConfig.exe ......................................................................... 48
  5.8.3 ASP.NET website administration ............................................ 48
  5.8.4 SharePoint security inspector .................................................. 50
5.9 Best practices .................................................................................. 54
6 Risk analysis for a generic SharePoint 2007 setup ............................. 54
  6.1 Risk analysis for a generic SharePoint 2007 setup ................. 54
    6.1.1 Assets ................................................................................. 54
    6.1.2 Origins of threats ................................................................. 54
    6.1.3 Classification of damage levels ........................................... 55
    6.1.4 Classification of probability levels ....................................... 55
    6.1.5 Calculation of risk level ....................................................... 55
    6.1.6 Attack classification ............................................................... 55
    6.1.7 Countermeasures for high risks .......................................... 57
    6.1.8 Remarks on risk analysis ....................................................... 57
  6.2 Best practices to avoid vulnerabilities ............................................ 58
1 Preface

This master thesis was written in a collaboration of ETH Zürich and Altran AG Zürich. Altran is a consulting company with 17,500 employees worldwide that focuses on consulting in engineering, business strategy and IT. Altran Zürich provides solutions based on Microsoft Office SharePoint Server 2007 since its beta release.
2 Abstract

Microsoft Office SharePoint Server 2007\textsuperscript{1} has become an important platform for enterprise content management and collaboration in companies around the world.

This thesis evaluates the security of SharePoint in terms of technical vulnerabilities (flaws of the implementation) by applying several thousand attacks through a web application scanner on two different systems. The first system is a minimal setup containing two sites and only out-of-the-box features while the second system is a clone of a productive system with dozens of customized web parts and around 70 sites. In the default implementation no vulnerabilities could be detected. The productive system revealed multiple serious issues (cross site scripting) in the customized parts.

In terms of logical vulnerabilities (flaws of the design and architecture) SharePoint’s architecture allows potential issues caused by administrators: The core of SharePoint’s access control features role assignments that combine principles with scopes (a group of securable objects with the same role assignments) and role definitions (groups of base permissions). SharePoint implements its security by letting newly created sub sites, lists or list items inherit the collection of role assignments from their parent scope. The inheritance can be broken to allow the creation of a new scope and the assignment of new role assignments. This mechanism produces a highly complex and confusing security topology - even in smaller systems - that is not properly represented in an overview in SharePoint’s default administration web user interface.

The lack of overview and the resulting lack of maintainability lead to logical vulnerabilities. As part of this thesis, SharePoint Security Inspector was developed to address those problems by providing additional views on the security topology. A scope-centric view displays the complete structure of a site collection and indicates inheritance breaks visually. A user-centric view collects all permissions and group assignments for a selected user in one view and also displays all role definition compartments and the including scopes with the according role definitions for that user. The findings of this thesis lead to a summary of best practices to avoid logical and technical vulnerabilities.

\textsuperscript{1} Further on referred to as SharePoint or SharePoint 2007
3 Introduction

Microsoft's SharePoint 2007 provides a broad palette of functionalities to rapidly implement intranet, extranet and internet solutions with the main focus on collaboration for all kinds of businesses.

While the product has huge success on the market, the amount of possibilities requires a lot of know-how and expertise to find best-practices to fulfill the customer’s requirements. Even more, security concerns are left beside in most cases and are assumed to be already solved by SharePoint itself.

This master thesis will inspect SharePoint 2007 in terms of security from different perspectives and will try to find weaknesses. The overall goal is a best-practice guide that will consider all found and known vulnerabilities\(^2\), misconfigurations\(^3\) and attacks\(^4\).


This part will focus on the functionality behind the curtain (authentication and access control) and will also show which parts can be configured by an administrator and what parts can be used by a programmer through the API.

Part 2 – Vulnerabilities and pitfalls

With the amount of configuration possibilities, the probability of misconfiguration grows as well. We will try to find the most common pitfalls. At this time\(^5\) there are no publicly known vulnerabilities in terms of security in SharePoint 2007. We will test its security structure against common attacks (e.g. cross-site-scripting, session hijacking, authentication bypassing, SQL-injection)

Part 3 – Best-practice guide for security in SharePoint 2007

Part 3 will conclude the thesis with a best practice guide that will show how to avoid mistakes in configuration and how to harden a SharePoint installation for internet or extranet use.

\(^2\) Vulnerability: Hole in the security structure that might compromise security (disclosure of secret data, ways of impersonation or access to undisclosed assets) and can be used for an attack.

\(^3\) Misconfiguration: Configuration that seems to cover a security requirement, but enables other vulnerabilities.

\(^4\) Attack: The attempt to exploit a vulnerability.

\(^5\) February 2008 until August 2008
3.1 Focus

Since SharePoint consists not only of an administrator's interface but most of all of a versatile programming interface, the thesis will consider both parts to deliver a comprehensive picture.

The thesis will not consider hardware security or the security of the programming language, but focus on the security of SharePoint 2007 itself as a software running on an ideal system connected to the internet\(^6\) and its API. (Since SharePoint 2007 runs on Windows Server 2003 (using IIS 6.0), the according configurations in the OS will also play a role.)

SharePoint 2007 can be purchased in two different packages, one is Windows SharePoint Services 3.0, which is free but has reduced functionality. The other one is Microsoft Office SharePoint Server 2007 which comes with certain license costs but has the full amount of features. The thesis will mainly focus on the full product, but will mention differences when appropriate.

SharePoint 2003 or older versions are not part of the thesis, since the security layout and the functionality are completely different.

The thesis will have a short introduction into SharePoint 2007, but the reader will be required to have a basic understanding of Office 2007, C#, Visual Studio 2005, both Windows SharePoint Services 3.0 and Microsoft Office SharePoint Server 2007.

---

\(^6\) The internet is not an ideal channel. This will be fully considered.
4 Security implementation of SharePoint 2007

This chapter will first give a very short general introduction into the general functionality of SharePoint 2007 and then focus on the most important security mechanisms.

4.1 Introduction into SharePoint 2007

This short introduction into SharePoint 2007 will not be comprehensive, but focus on the terms and concepts that are important during the security analysis following in the next chapter.  

4.1.1 Components

SharePoint 2007 is not just a document management system but a platform that fulfills multiple requirements, such as:

- Collaboration: Integration into Office 2007, use of workflows
- CMS – content management system: Publication of content into the web through a publishing process
- DMS – document management system: Versioning, check-in/check-out of documents
- Search: Highly configurable search engine that can index sources of almost any kind
- Business Data: Integration of almost any kind of business data (e.g. SAP)
- Excel Services: Integration of Excel calculations.

4.1.2 Structure of a SharePoint installation

The ‘inner structure’ of a SharePoint Installation consists of a farm that contains one or multiple web applications. A web application groups site collections as a container. A site collection is a web site within SharePoint and provides a boundary for site groups.

A site collection has one root site under which multiple sub sites exist, which are sites as well. A list is a configurable data structure within a site where end-user data can be stored. A list item is a unique row inside a list.

---

7 There are more than a dozen books in the stores that provide comprehensive introductions into SharePoint 2007.
The following example should demonstrate how a typical site collection can be structured:

4.1.3 Event handlers
Every object (site, list or list item) has a palette of events that can be triggered. Those events can be intercepted by event handlers and custom code can be executed before (synchronous) or after (asynchronous) the time of the event. Event Handlers cannot be administrated through the web interface but must be created and attached to the targeted object by code.

Example:
Whenever an employee is added to an employee list, that employee should be deleted from the list of job candidates. This can be easily implemented by adding the event handler for the ItemAdded event to the employee list and writing the necessary code in the event handler to delete the item from the job candidates list.

4.1.4 Web parts
Web parts encapsulate their own functionality and layout and can be dragged and dropped onto a page or site of choice within the web user interface. In most cases, a productive system needs many customized web parts to fulfill all the customer’s requirements since the out-of-box web parts are rather basic.
Example:
A company has a site for every team. On each such team site, the team manager shall be shown with his name, phone number and a photograph. The corresponding web part searches for the current team manager and displays the information.

4.1.5 Interfaces
SharePoint is mostly accessed as a web application through a browser. This is how SharePoint is used in daily business as in a company’s intranet, for instance. Also, most administrative tasks can be done through the browser as well, such as adding a new site or publishing content to the web. More specialized administrative tasks are done through a web site called central administration which is automatically created after SharePoint is set up the first time on a server. Central administration allows more critical tasks as for example the creation or deletion of a site collection or the management of zones (see 4.2.3). Since SharePoint consists of a huge framework based on .NET, the API allows access to even more functions and properties that can be configured through code.

As of now, the reader knows the general concepts that will be referred to in the next chapter, which lays out SharePoint’s security mechanisms.

4.2 SharePoint security

4.2.1 Code execution
No matter where an action is triggered – if in the web user interface, through a web service or directly in the API – all code always executes under the privileges of the logged-in user. This means that if the code accesses an object where the current user has no privileges, the operation fails.

4.2.2 Authentication
SharePoint takes windows domains as a default membership provider. This means that all users log into the web user interface with their domain login. In some scenarios it might be required that a certain set of users have accounts that do not depend on a windows domain (e.g. external partners). In those cases SharePoint allows to plug in custom membership providers.

The easiest one to implement is a SQL-database that stores simply user names and their passwords. The ASP.NET framework already contains a setup file to create such a database. [1]

4.2.3 Zones
Zones provide different authentication views on the same site collection which are accessed through separate URLs. There can be up to 5 different zones for the same site collection. The names of the zones do not have any meaning, except that the default zone should be through windows authentication to avoid issues with the search crawler.
If a custom membership provider is used, the web application has to be extended, a new zone is created and authentication runs over forms based authentication if accessed through that zone. This only changes the way of authentication. The content and the structure of the objects stay the same for all zones.

### 4.2.4 Access control

All actions, no matter from which interface are always permitted or denied through the same code lines. Hence there is complete consistency in privileges between access from all interfaces on a given object.

### 4.2.5 Role assignments

Role assignments are the very core of access control in SharePoint. A role assignment always consists of a triple: Role definition, scope and a principal which can be a user or a group.
Permissions are never assigned directly, but are collected in groups of base permissions called role definitions. SharePoint knows 32 base permissions. For details, see: [2]. By default SharePoint knows the role definitions full control, read, contribute, design and limited access. The last one, limited access, is not a real set of permissions but points out, that there are privileges in a scope within the current scope (see next paragraph for details about scopes).

4.2.6 Scopes
A group of objects that are connected through a tree of sites and subsites and in which the set of role assignments stays the same is called a scope or role assignment compartment. This means, that all principals have the same permissions on all objects within that scope. By default, if a new subsite within a scope is defined, all role assignments are inherited to the new subsite and its contents. In most practical cases, some object shall be only accessed and read by a subset of users. These objects will then be in a new scope. This can be done in two ways:

First, an administrator can assign an object its own privileges through the web user interface. This will automatically break the inheritance of those privileges and create a new scope. The other way is to break permission inheritance through the API. Every site, list or list item object has a corresponding method to break inheritance and have its own set of role assignments.

4.2.7 Role definition compartments
Through the API, a developer has the possibility to define new role definitions and even a group of objects in which those role definitions are valid. We call such a scope a role definition compartment. (From now on, the term scope refers only to a compartment of role assignments). Although this is not often necessary on productive systems, this mechanism allows having a group of objects with its very own role definitions. By default – similar to role assignments – role definitions are inherited when a new subsite is created. This inheritance can explicitly be broken through the API. Since this also renders role assignments invalid, permission inheritance is broken implicitly whenever role definition inheritance is broken.

Example for role definition and permission inheritance
The concept of inheritance and inheritance breaks are important to understand the proceedings of the thesis. The following example should illustrate the concepts:
As mentioned before, if no breaks would apply, the whole tree structure would be in the same role definition compartment and within that compartment would be one single scope (role assignment compartment). Now, the role definition inheritance is broken at the level of subsite B. This cuts the initial compartments off (colored in blue) and creates a new role definition compartment (role definition compartment 2). Since this also breaks role assignments, there is automatically a new scope created (Scope 2.1, colored in orange).

Additionally, permission inheritance is broken at the level of list B.1. This creates a new scope (Scope 2.2, colored in red) and cuts off Scope 2.1).

It is important to notice that the inheritance is actually a pseudo-inheritance, since the new scopes do no longer have any connection in terms of privileges to their parent scope. (Their only connection is that the broken inheritance can be reverted any time such that the scope rejoins the former scope.) Also note that a scope cannot be split by another scope. All objects in the scope are always grouped together.

We will see in the next chapter that the architecture of inheritance leads to confusion that can easily result in security leaks.
5  Vulnerabilities & risks

5.1  Introduction

The beginning of this chapter will categorize vulnerabilities in two main groups: technical and logical vulnerabilities. In terms of technical vulnerabilities, some information about general attacks and techniques that can compromise the integrity of web applications in general will be presented. Then, most of these attacks will be applied against a ‘bare’ SharePoint Installation. ‘Bare’ in that way, that the system is not configured for real use and has no content in it, but consist only of a handful of sites and lists that are secured by SharePoint’s out-of-the-box security. In comparison, a real and productive system will be cloned and tested against the same attacks. This productive system is an actual project that was implemented at Altran AG in 2008 and consists of many dozen sites and thousands of custom code lines.

The remainder of the chapter will discuss logical vulnerabilities and introduce a handful of tools that might help to avoid mistakes, such as SharePoint Security Inspector, an administration tool explicitly designed for SharePoint 2007 that was developed by the writer of this thesis to address some logical vulnerabilities and ease the task of daily administration in SharePoint environments.
5.2 Technical vs. logical vulnerabilities

Vulnerabilities can be divided into two subgroups: technical and logical. Technical vulnerabilities suffer from mistakes that were made in the implementation of a system or in the implementation of underlying components. Since those mistakes mostly result in error messages or a breakdown of the system, automatic scanners can be applied to search for these vulnerabilities.

Example for a technical vulnerability:
A user input field can be overloaded with input such that the server crashes when the data is submitted. This is a typical buffer overflow vulnerability, which is a technical vulnerability, since the implementation lacks of input validation.

Logical vulnerabilities are flaws of the design or the architecture of a system. This includes also flaws that origin from misconfigurations by administrators. It is easy to see, that checks for logical vulnerabilities are very hard to integrate in automated scanning processes since an application would need to verify if the semantic behavior of the system under inspection is as expected. (Of course, this would be rather possible with model-based testing, but the construction of a model for SharePoint 2007 would easily go beyond the scope - and the available time - of this thesis.)

Example for a logical vulnerability:
A developer expects a user to navigate from site A, over site B to Site C, where site B is an authentication mechanism and Site C is in a protected area. If it is possible to navigate directly from site A to site C, this would be a typical logical vulnerability.
5.3 Attacks

Technical vulnerabilities can mostly be found through automatic attacks. This section will shortly explain what attacks & techniques apply to SharePoint 2007 as a web application and describe the way they work. To start with, the first few techniques will give ideas on how to collect information about the target server and where this information can be found. Second, it will be pointed out, that information from the client cannot be trusted and how client-side restrictions can easily be circumvented. The third section focuses on state based attacks, like session hijacking. After that, we address a handful of the most dangerous and widespread techniques: Attacks against user-supplied input data, like cross-site-scripting and SQL injection. Further, buffer overflows and other language-based attacks will be explained. The enumeration of attacks will be finished with an overview on attacks against web services followed by a summary overview of all attacks and techniques. For each technique it will be pointed out, how it impacts SharePoint, what tools support the technique and which countermeasures can be taken.

The categorization of the attacks is mostly taken from [3] while some attacks have been added and others have been omitted where no application to SharePoint could be found. The attacks are not ordered by their degree of impact on SharePoint. The section 5.4 will then be a protocol of how the attacks have been executed on the two test systems.
5.3.1 Collecting information about the target
The following three techniques will give an idea on how to harvest critical information about a targeted system.

5.3.1.1 Google hacks
Google Hacks take the vast amount of indexed sites by Google to search for information that might be relevant for security or even disclose security critical information. This is done by specific search patterns (called Dorks\(^8\)) that try to find results for the URL under inspection in sites indexed by Google.

Impact on SharePoint 2007
Google hacks are far from being new, since all Dorks are based on well known vulnerabilities that have sometimes been discovered years ago. They work by querying Google's site index for security relevant information. While some of them look for critical information in the source code, others just check for the use of certain third party products (like certain php guestbooks) that have proven to be unsecure in their initial configuration. Dorks that directly apply to SharePoint are not known at this time. Future dorks might manifest themselves through comments developers left on aspx pages or controls that are used in a SharePoint site. Hence, the impact is rather small (at least for now). So, it might do no harm to run a complete scan once with a tool like Goolag.

Supporting tools
Goolag Scanner [4] is an application developed by the cult of the dead cow. The current version comes with more than 1400 dorks. A complete scan can take up to several hours since Google starts blocking IPs as soon as too many requests originate from them (see german version of block message below). The IP can be unblocked by passing a captcha\(^9\) test.

---

\(^8\) Dork = A detailed search pattern - heretofore used with Google's search engine - that uses Google to show untapped results for web sites previously indexed by Google. The intention of a dork is to find results that might show information relevant to security issues and/or confidential data.

\(^9\) Captcha = is a type of test used to ensure that the response is not generated by a computer. The test often displays a picture with a graphically deformed text phrase that has to be correctly recognized.
Security Issues in SharePoint 2007

5. Vulnerabilities & risks

Figure 4: Goolag scanning

Figure 5: IP is blocked by Google
When the selected scans are finished, Goolag presents a summary with the results. Failures indicate how many queries could not be executed because the IP was blocked at that time.

Countermeasures

If the system under inspection is available through the internet, there are the following countermeasures:

- Run Goolag scan every few months
- Try to replace all ‘dorks’ by the methods recommended by Goolag.
- ‘dorks’ that cannot be removed by the methods recommended by Goolag, can be removed from the Google index completely (http://www.google.de/intl/de/remove.html)
5.3.1.2 Source code & error messages

Among developers it is part of the etiquette and of good coding style to comment their code with precise and helpful descriptions such that other programmers can easily understand it. But what is helpful for other programmers might also be very conclusive for attackers. Of course attackers do not have access to the source code of a program in most cases, but this does not apply for HTML or JAVA script, where the ‘source’ is fully readable in the client browser. Hence, if the HTML source does include debug information like connection strings or the like, a hacker might get important information for an attack.

Impact on SharePoint

The standard parts of SharePoint do have very little obsolete information in it. But an inexperienced developer might write a web part or a web control for his SharePoint application that includes such critical information. Besides, there is the possibility to switch on debug information and hide custom error pages for each web application (This is done directly in the web.config files). If these modes are selected, SharePoint and ASP.NET will show a detailed error message and part of the stack trace in case of an application error provoked by an attacker. This information might give the attacker clues to fine tune his attacks.

Supporting tools

Paros proxy [5] has filter functions that can validate regular expressions. With regular expressions for HTML comments and SQL Connection Strings, a large site can be easily scanned in seconds for critical information in the source code.

Countermeasures

- Implement coding guidelines to prevent comments in HTML or Java script code
- Switch off debug information when system is productive
- Run scans with Paros Proxy to check for critical info.
5.3.1.3 Guessing files & directories

In 2002, a Swedish software company’s quarter earnings have been disclosed to the public unintentionally because the files were not linked but already uploaded to the website and not protected by any password. A journalist from Reuters guessed the files location by knowing the URL from the previous one and published the results before they were officially released [6].

Although, this technique seems to be weak, there is always a good chance that parts of an application are not mediated by access control, especially if the amount of securable objects is as huge as in a SharePoint environment.

Impact on SharePoint 2007

SharePoint itself fulfills the criteria of complete mediation and has bound every site, list etc. to an access control list. But as we will see later: It is very easy to lose overview when the hierarchy of secured objects grows bigger and security inheritance is broken at multiple places. It is a likely scenario that lists and their contents can be read by someone who should not.

Supporting tools

SharePoint Security Inspector is designed to give a better overview on what user account has privileges on which objects. See 5.8.4 for more details on the tool.

Countermeasures

- Double-check the customer’s security requirements using SharePoint Security Inspector.
- Make sure that all linked files are mediated by SharePoint Security.
5.3.2 Bypassing restrictions on inputs

In contrast to normal applications that run locally, it is not sufficient to check user inputs just at the client-side. All information that is being sent to the server can be changed after the validation applies and before it actually leaves the client’s computer.

Impact on SharePoint 2007

Custom made forms and web parts are very vulnerable to this technique, since overly long and unvalidated inputs can lead to unexpected behavior such as deformation of the site layout or even crashes of the server.

Supporting tools

Paros Proxy [5] provides the possibility to trap HTTP requests and to change the content of the message before it finally reaches the server. This circumvents all validation mechanisms that are located on the client side.

Countermeasures

- All input restrictions must be revalidated on the server side
- Never rely on information coming from the client
5.3.3 State based attacks

5.3.3.1 Hidden fields

Hidden fields are input fields in HTML code that have the hidden tag (type attribute is set to hidden). Since the HTTP protocol is stateless, hidden fields are one way to keep track of a web users choice. The downside is that hidden fields are openly readable in the source code of the page. Besides, it is easily possible to iterate over a document by the Document object model (DOM) and enumerate all hidden fields.

Impact on SharePoint 2007

Hidden fields as a track of state might only appear in custom made forms.

Supporting tools

The PageSpy Tool helps to easily navigate through the DOM and to inspect the HTML source.

Countermeasures

- Hidden fields should be avoided whenever possible.
- The state of a client should be kept explicitly on the server.
5.3.3.2 **CGI parameters**

CGI parameters are passed through the URL after the ‘?’ character using the GET method. They are separated by the ‘&’ character. Hence CGI parameters are visible in the address bar of the browser such that a user can change and manipulate them freely.

**Impact on SharePoint 2007**

As we will see later on, CGI parameters are critical for the security of productive SharePoint environments. Unvalidated CGI parameters are very risky and even allow the execution of user supplied scripts.

**Supporting tools**

Cenzic’s Hailstorm [7] is a web application scanner that automatically tests sites for unvalidated CGI parameters.

**Countermeasures**

- Validate the URL before processing
- Terminate processing if validation fails. Do not try to crop the input since this might have unpleasant side effects.
- Use a white list\(^{10}\) approach

5.3.3.3 **Cookie poisoning**

There are four forms of cookies, characterized by 2 settings:

- Secure/nonsecure
- Persistent/nonpersistent

Secure cookies are only transmitted over SSL (HTTPS) but the cookie itself is not encrypted in any way. Persistent cookies are stored on the clients hard disk until an expiry date while non-persistent ones are deleted when the browser closes. Some applications use the dates in their cookies to give access to certain content on websites. Since the cookies are stored on the hard disk, it is easy to forge this date and get extended access to the site.

**Impact on SharePoint 2007**

SharePoint’s internal cookie management can be configured through the central administration\(^{11}\), where the maximum request duration can be set. The cookies that are stored on the client are encrypted and non-persistent. Hence, cookie poisoning has only an impact on SharePoint if additional information is programmed to be stored unencrypted in the cookies.

---

\(^{10}\) White list approach: Only allow inputs that fulfill a certain pattern. In opposite to a black list approach that tries to find bad input with patterns.

\(^{11}\) See 4.1.5 for details about central administration.
Supporting tools

CookiePal [8] and CookieCrusher [9]

Countermeasures

- Do not rely on a cookie’s own expiry date.
- Encrypt the cookie if the information must be stored on the client.

5.3.3.4 Session hijacking

Session identifiers are used in web applications to make the stateless HTTP protocol stateful and to keep track of a user’s history and display personalized content. An attacker might try to fake a session ID of another user to see information he is not intended to by:

- Modifying an ID randomly to get a valid ID of another user
- Guessing the sequence of IDs that the site generates

Impact on SharePoint 2007

SharePoint’s session identifiers are encrypted and hence hard to guess or forge. Hijacking is therefore unlikely.
5.3.4 Attacking user supplied input data

5.3.4.1 Cross site scripting
Cross site scripting is about scripts that are entered by an attacker into the URL or a form field of a web site and that get executed on the computer of a victim (client-side). Most of the time, these scripts and their operations are harmless since the victim’s browser only has a restricted amount of operations. It is for example not possible to format the victim’s hard disk by using cross site scripting. Nevertheless, an attacker might be able to collect information that was only intended for the victim, such as session cookies or present fake content to the victim.

The easiest way to find out if a site is vulnerable to cross-site-scripting is to try to insert a script like the following:

```html
<script>
    alert("XSS alert!")
</script>
```

If the target website is vulnerable, this will lead to a popup that displays the message “XSS alert” in the victim’s browser. Of course, this type of attack is annoying but harmless. More advanced scripts, can easily steal a cookie and send it to the attacker’s server. The following example reads the client’s cookie as soon as the client’s browser tries to display a 1x1 pixel sized image (.gif-format). For the web server, the additional cookie parameter is redundant, but the request will be stored in the server logs. Hence, the attacker has a valid session cookie.

```html
<script>
</script>
```

Generally, cross site scripting attacks can be divided into two major categories:

5.3.4.2 Stored cross site scripting
Stored cross site scripting uses websites that allow users to enter information that will then be stored on a server and is displayed again for other users. Examples for such sites are guest books or blog comments. The scripts executes as soon as another user visits the site and the stored content is displayed.

5.3.4.3 Reflected cross site scripting
Reflected cross site scripting embeds the script in the CGI parameters of a URL. With this technique the content of the original site can be changed by the script. A famous example for reflected cross site scripting attack happened in august 2006, when CBS News published an
official announcement of George W. Bush, that a 9 year old boy will be the new chairman of the United States Information security department [8].

**Impact on SharePoint 2007**

As one will see later in the tests against the productive system, Cross-Site-Scripting is an important source of vulnerabilities in customized code segments that use URL parameters. Even SharePoint 2007’s own early implementation had such a vulnerability that was later patched by Microsoft.

**Countermeasures**

The web application needs to validate possible ways for a client to submit input, such as forms, headers, cookie fields, hidden fields and parameters. The validation has to make sure that all script parts in the input are changed into an unexecutable form. As a best practice it is wise to use a **white list approach**, meaning that the validation focuses on letting through **only** inputs that match a certain pattern. The other possibility is to recognize **evil input** (black list approach) and to let everything else through. This approach can be very cumbersome for two main reasons:

- New ways of bad input are discovered on a daily base and hence the code needs to be updated each time a new attack is disclosed.
- Bad input has more than one face, meaning that a string can be encoded in different ways. One way to avoid this is **canonicalization**

**5.3.4.4 SQL injection**

If user input is forwarded to a SQL query without proper validation, the query can be changed to an attacker’s advantage. The sample query below illustrates that if the application takes the userid from an input field without to check it for escaping characters, an attacker can rewrite the query.

```sql
//“jim” as input
Select userdata from userinfo where userid = ‘jim’ and userpassword = ‘quebec’;

//“jim’ or ‘1’='1” as input
Select userdata from userinfo where userid =’jim’ or ‘1’='1’ and userpassword=’quebec’
```

---

12 Canonicalization: Process of converting data that has more than one possible representation into a normalized, canonical representation. [3]
Impact on SharePoint 2007
The queries SharePoint executes by default seem – according to the tests – not to be subject to SQL injection vulnerabilities. Critical issues result from custom code parts developed by integrators that forget to validate inputs.

Countermeasures
- Validate input that is used in SQL queries

5.3.5 Language Based Attacks
Language-based attacks exploit known problems of programming languages in which components of the server are implemented. Some of them, like buffer overflows, were used in the very first malware ever applied in history. Nevertheless, some of them still appear in today’s web software.

5.3.5.1 Buffer Overflows
Probably the most famous attacks are buffer overflows. They exploit missing validation of length of user input by overwriting other memory locations on the execution stack. If an attacker successfully applies a buffer overflow, he has a good chance of taking control over the server by overwriting the return address on the stack with an address in his own code. Hence, he can execute his own code.

Fortunately, .NET languages run managed code and hence are practically immune to this kind of attack. Nevertheless, some components still call methods of unmanaged code that can potentially contain buffer overflow vulnerabilities.

Impact on SharePoint 2007
Although the SharePoint Framework uses some methods that run in unmanaged code, the applied tests found no sign of a buffer overflow vulnerability in the standard functionality. (The tests with Hailstorm report buffer overflows because the server did not respond in reasonable time. The reason for this is that the server had to compile the pages first, which lead to longer loading times.)

Countermeasures
- Use unmanaged libraries with precaution and perform automated buffer overflow tests against the routines that use them.

Supporting Tools
Cenzic’s Hailstorm performs many buffer overflow attacks on any possible input. Many reported issues are false positives. But if the server stops responding, it is very likely that the web scanner detected a true vulnerability.
5.3.5.2 Canonicalization

As seen before, many attacks origin by a lack of input validation. Unfortunately, filtering bad input is very cumbersome due to the many representations they can have. Canonicalization is about presenting data in a standard form.

Example

UTF-8 [9] is an encoding that the Internet Engineering Task Force (IETF) demands to be supported by all internet protocols. The algorithm maps all ASCII characters to a one-byte representation that has the same bits set as the ASCII presentation. For all other characters (e.g. latin letter, diacritics and other widely used symbols) UTF-8 uses a 2- to 4-byte representation, depending on the symbol. The first byte of the UTF-8 presentation always indicates how many bytes are about to follow, indicated by the number of 1’s before the first 0. If a character is represented by more than one byte, the second and all higher bytes start with ‘10’.

E.g. characters, represented by 2 bytes are in the range of 110xxxxx 10xxxxxx, where the 11 x’s mark the bits for the actual representation. So, $2^{11} - 2^7$ equals 1920 possible characters in the range of the 2-byte representation.

Hence, an ASCII character in UTF-8 representation always starts with a 0 and is in the range of 00000000 to 01111111. Even though the standard demands to always use the shortest possible representation, some attacks successfully passed firewalls by submitting escaping characters in overly long representations. E.g. the ASCII character 01100001 was submitted as 11000001 101000001, where the first byte indicates, that a character encoded in 2 bytes follows. But as mentioned before, in this case, only the 1 byte representation would have been allowed by the standard. So unallowed sequences of characters could possibly sneak through validation check on the server side if not all forms of representation are considered.

Overly long encoding applied to the character ‘/’ leads to 3 possible representations: ‘/’, %5c and %c0%af, where the last one was successfully used in an attack against IIS4 and IIS5 web servers by requesting the following URL:

http://www.myvictim.com/app/..%c0af..%c0af../winnt/system32/cmd.exe?/c+dir

Obviously, the attacker was able to perform random commands on the attacked server.

Impact on SharePoint 2007

So far, there could have been found no traces of canonicalization vulnerabilities in the implementation of SharePoint. Again, customized parts (web parts or event handlers) may potentially fail to block all forms of bad input.
Supporting tools

Napkin [12] is a small tool for decoding/encoding to different representations and comes in handy when validation checks need to be tested.

Countermeasures

It is important for a developer of customized parts to understand how the browser and the web server encode and decode before the input reaches his code. Only then, information can be canonicalized in the right way.
5.3.6 Attacking the server

5.3.6.1 Fingerprinting the server

Fingerprinting means to get information about the software installed on a server and the corresponding patch levels. With that information an attacker might find a known but unpatched vulnerability that he can exploit. With sources like BugTraq (on www.securityfocus.com) or CERT (www.cert.org) known vulnerabilities for certain versions and patch levels of software can easily be found.

Impact on SharePoint 2007

Right now, the only known vulnerability is a cross-site-scripting vulnerability [11] that has already been patched with Service Pack 1 for SharePoint Services.

Countermeasures

Since new issues might be discovered, it is important to keep track of the news published on vulnerability databases like BugTraq or Cert. Furthermore, it is helpful to know how much information about the server actually can be discovered by fingerprinting it with a tool like Httprint [12].

Supporting tools

Vulnerability Databases keep track of all disclosed issues:

- Open Source Vulnerability Database [15]
- Securityfocus: BugTraq e-mail list and vulnerability database [16]
- Carnegie Mellon University's CERT (Computer Emergency Response Team) [17]

Httprint [12] sends different requests to a server and tries to identify the web server by the order of information in the response.
5.3.6.2 Denial of service

The goal of DoS (Denial of Service)-Attacks is to overload the web server with requests. It is important to know that this can not be done with a common browser. According to the RFC 2616 standard, browser can at most have 2 connections to a server. Hence, a network of machines is necessary to really bring a server down to its knees. Of course, servers can limit the number of connections they accept, but if all slots are taken by an attacker, no legitimate user can access the server anymore.

Impact on SharePoint 2007

Since SharePoint 2007 runs on IIS 6.0, DoS attacks are more an issue for the correct configuration of IIS. Nevertheless, a SharePoint farm with multiple web front servers can reduce the chance of a DoS attack by every additional load balanced machine.

Countermeasures

Servers need to be protected by intrusion detection systems (IDS) or bandwidth management solutions that recognize patterns of misuse and block incoming requests from IP’s that match those patterns. Additionally, multiple web front servers make the system more robust for a vast amount of requests.

Supporting tools

Due to the number of machines that are needed to perform such an attack, there is no easy-to-handle, downloadable tool that tests a system for DoS attack by the click of a single button. But a custom program might be developed in reasonable time that forks off multiple child processes that all access resources on the targeted server.
5.3.7 Breaking authentication

Password protected sites can easily become targets of brute-force attacks. These attacks simply try out every reasonable combination of login name and password to get into a protected area. One way of protection is to force stronger passwords with a minimal length and a certain amount of non-alphanumeric characters. This enlarges the domain of possible passwords to an almost unfeasible amount. To further reduce the risk most sites set a limit to the amount of times a user can enter a wrong password until the account gets locked. Which again, opens a new way of attack: Trying to lock all accounts on a server. As one can easily see, security is a kind of arms race with the methods of attackers.

Impact on SharePoint 2007

Normal login and FBA as well are fully configurable in the terms of password strength (windows login zones use the policies of the domain) and the use of SSL. This can lead to an unsecure system if passwords are allowed that are too weak. If the communication happens over unsecured channels passwords might be intercepted through eaves-dropping.

Supporting tools

Brutus [13] is a brute-force authentication tool that uses a library of passwords and common login names to login into protected sites

Countermeasures

- Use SSL (HTTPS) wherever reasonable
- Use strong passwords
- Lock accounts after a certain number of false trials
- Additionally secure login pages with a captcha.
5.3.8 Attacks on web services

Web services provide a new way of adding functionality to the web. Before attacks on web services are explained, here a short summary on what technologies web services are build upon.

UDDI (universal description, discovery and integration)

Discovery services help to advertise and locate web services. UDDI is a framework that describes and classifies details about the interfaces of the service and other information like the whereabouts of the organization that provides the service.

WSDL (web service description language)

WSDL describes in which way a client has to interact with the service, where it is located, what interfaces it provides and what the service can actually do.

SOAP

SOAP is a technology to transport XML documents over standard transport protocols (e.g. TCP, HTTP, SMTP). A soap message consists of a SOAP envelope in which the actual XML message resides as SOAP body while the SOAP header contains details about the message (e.g. length).

5.3.8.1 Web services: WSDL scanning attack

Since WSDL files have the .WSDL extension an attacker can easily enumerate them over the internet by using Google (e.g. searching for “filetype:wsdl”). As mentioned before, web services are supposed to be discovered through UDDI and not directly by scanning for WSDL files. This might become a security problem where WSDL files are generated automatically and contain services which were not meant to be made publicly available and are not under any kind of access control.

Impact on SharePoint 2007

The web services that ship with SharePoint 2007 demand user credentials on every invocation. The code that handles the call is run under the given credentials. Hence, web services are restricted to the same users and permissions as if the demanded functionality was used through the web interface.

It is possible to add custom web services to SharePoint 2007. As long as the service uses the interfaces of SharePoint 2007 and the code does not run under elevated privileges, the service will always need valid credentials to properly perform the necessary operations.

Supporting tools

Google can be used to search for files with the WSDL extension.
Countermeasures
It is important to make sure that web services stay under complete mediation. SharePoint 2007 standard web services already fulfill that criterion. All custom services need to be inspected to make sure that they do not open backdoors for attackers by running under elevated privileges without asking for user credentials.

5.3.8.2 Web services parameter tampering
As with any kind of service that allows user inputs, web services are not automatically save against invalid or bad input. SQL-injections, buffer overflows and the like apply to web services as well, depending on how the underlying code is implemented. Just because the parameters are transmitted by XML does not mean they cannot be tampered.

Impact on SharePoint 2007
As far as the standard web services could have been tested in the context of this thesis, inputs are properly validated in out-of-the-box web services and even fire conclusive exceptions about which parameter could not be validated.

Countermeasures
The fact, the web service requests are transmitted as XML does not improve its security. Its parameters have to be validated exactly the same as for form data or CGI parameters.

5.3.8.3 Web services XPath injection attack
Like SQL for databases, XPath is used to query XML documents. With an XPath Injection, an attacker could inject malicious expressions as part of a SOAP request. This can open up even more possibilities than an SQL injection, since XML documents are in a standardized format and an attacker does not have to know details about the structure (e.g. table names in a database) to construct malicious input.

Impact on SharePoint 2007
As mentioned before, the standard palette of web services properly validate input. Hence, it is important that developers of custom web services are aware of the risk of XPath injection.

Countermeasures
Developers need to properly validate web service requests for XPath expressions or escape characters (like “/” or “/”).

5.3.8.4 Web services: recursive payload attack
XML nesting is a way to represent non-trivial relationships among elements. If an element appears between the tags of another element it is nested. Now, an attacker can construct a document that nests 100’000 elements to attempt to crash the web service by overwhelming the parser. Most parsers use the technique of a push-down automaton, where a map is created of the XML document, telling the parser which steps to take, if it reaches certain elements. If the
payload is recursive, this could lead to an endless loop for the parser that can consume a lot of memory or even crash the machine.

**Impact on SharePoint 2007**

The applied tests showed no crashes in SharePoint 2007 through recursive payload.

**Supporting tools**

No according tool known.

**Countermeasures**

If custom code is added, it might make sense to validate input for recursive nesting and drop the request if the recursion makes no sense in the context of the service.

**5.3.8.5 Web services: oversize payload attack**

If a web service does not check the length of the XML received before parsing it, the server is vulnerable to buffer overflow attacks. An attacker could easily try to send XML files in the range of several gigabytes. If the application loads the XML document into memory before accessing it, it is especially vulnerable (as seen earlier in the discussion of buffer overflow attacks).

**Impact on SharePoint 2007**

The impact is the same as with the classic buffer overflow attack: Standard SharePoint 2007 seems to be safe against that kind of attack, while newly added code has to implement input validation in a proper way.

**Countermeasures**

Proper input validation and check of reasonable length.
5.4 Testing web applications
As one can easily see, looking at the large amount of possible techniques and thinking of an average SharePoint 2007 system with more than a few dozen sites, it is very time consuming to perform all tests manually. Security companies started in the 90’s to implement automatized security scanners that checked web applications for the most common flaws. With the help of these tools, applications could be scanned for many hundred attacks in a few minutes. In the next section, we will introduce two different systems and examine them in terms of technical vulnerabilities with a web application scanner.

5.5 Testing the bare system

5.5.1 The bare system
The purpose of the bare system is to have a small running system that has no customized pages or web parts but consists only of out-of-the-box parts and a minimum of sites:

- One root site
- One sub site
- One list beneath the sub site

Each of these objects has its own scope (its own set of role assignments). The list is configured to be only accessible by a few users. The system has no anonymous users allowed.

A broad collection of manual attacks has been performed against the bare system. But because of the huge amount of pages even in a reduced system, the test could not be comprehensive [14]. This is why a security tool called Hailstorm was used later, that runs up to two thousand attacks in one run.

5.5.2 The testing tool
Since manual testing for vulnerabilities can be very time consuming and cumbersome, a security testing tool was used against the livesys. The choice felt on Cenzic’s Hailstorm due to its
reputation in the web. Hailstorm is a one-click solution that performs several hundred attacks on the site URL provided by the user. The attacks consist of SQL injection attacks, buffer overflow attacks and cross site scripting attacks.

5.5.3 First run: testing the bare system as administrator

The tests on the ‘bare’ system with Hailstorm showed no detected vulnerabilities, but a few recommendations for improvements.

Vulnerabilities: 0

The scan found no vulnerabilities in the ‘bare’ system.

Informations: 2

The scanner detected IIS 6.0 and ASP.NET as used platforms.

Warnings: 4

Based on the found platforms, Hailstorm found 4 reported issues in its vulnerability database.
5.6 Testing the productive system

5.6.1 The productive system
The system under inspection is a real and running application that ALTRAN AG has provided for one of its customers. (Due to legal terms, all critical names and URL’s are blurred out or anonymized. The system will be further called ‘livesys’. The system consists of a server SharePoint installation that basically consists of two different site collections. One site collection provides content for an internet facing site and the other site collection is explicitly for administration purposes. In the remainder of this chapter, the site collection for the internet site will be called ‘public web’ and the other will be called ‘admin web’.

The public web provides 3 different zones (see 4.2.3 for details about zones): the internet zone (www.livesys.com), a intranet zone for content administrators (webadmin.livesys.com) and an internal default zone (http://live).

The admin web consists of 2 different zones: the default zone (http://liveadmin) and an internet zone (adm.livesys.com).

Besides the standard functionality of SharePoint, livesys has an extensive amount of customized web parts and event handlers that have been developed specifically for this system. The further tests will show that these parts are the most vulnerable in the system.

5.6.2 Building a copy of the live system
For obvious reason it might have been risky to run many hundred attacks against a system that is already in production. This is why a copy of the live system had to be made that comes as close as possible to the original.

5.6.2.1 Pitfalls of copying a live system
There are several issues that make copying a live system not as easy as one might think. The first one is that every configuration database of SharePoint has its own unique identifier which matches with the SID\(^\text{13}\) of the underlying system. With other words, an exact copy of all databases would never run at all.

The second issue is that SharePoint uses several system and domain accounts under which the system runs (e.g. service account, search account). A copy of the system will not have the same users. Hence the existing users must be replaced by the new ones.

Furthermore, one might think, that the integrated backup solution of SharePoint should do the job. Sadly, Microsoft itself points to the issue, that a restore of the configuration database is not

\(^{13}\text{SID: Security Identifier, see [18]}\)
possible [2]. This means that if the system hardware crashes, a recovery is not possible without first recreating the configuration database from scratch.

5.6.2.2 Step by step: copy a live system

What follows is a rough step by step guide to quickly clone an existing SharePoint system.

1. Make backup copies of all content databases from the original system (copies of the configuration database or the central administration are not necessary)
2. Install a new machine (virtual or dedicated) with Windows server 2003 (Service Pack 2)
3. Set up a domain controller (depending on original system)
4. Install mail features
5. Create necessary system accounts
6. Run SharePoint 2007 configuration wizard
7. Install MOSS Service Pack 1 (depending on original system)
8. Copy the www-root (under C:\inetpub)
9. Get all necessary files from the 12 hive
   a. Control templates
   b. Features
   c. XML definitions
10. Restore the backups from the content databases into either a SQL-server running on the same machine or an SQL server somewhere in the same farm
11. Startup the central administration and create the same web applications that were on the old system.
5.6.3  First run: anonymous user on public web

Hailstorm is run against the public web without any user credentials.

Vulnerabilities: 3

The report shows 3 cross site scripting vulnerabilities, all provoked by the injection value `<script>alert(12136078.1277)</script>`

Information: 2

The scanner detected IIS 6.0 and ASP.NET as used platforms.

Warnings: 9

Besides the warnings from the vulnerability database, the scanner detected five possible buffer overflow vulnerabilities. Those were false positives due to late reaction from the system.
5.6.4 Second run: site admin user on public web

The public web was tested with the credentials of an administrator.

Vulnerabilities: 5

The scanner detected 5 actual cross site scripting vulnerabilities. All of them resulted from custom web parts that lack of proper input validation, such that injection of cross site scripts is possible.

Information: 2

The scanner detected IIS 6.0 and ASP.NET as used platforms.

Warnings: 8

Besides the warnings from the vulnerability database, the scanner detected four possible buffer overflow vulnerabilities. Those were false positives due to late reaction from the system.
5.6.5 Third run: site admin user on admin web

The *admin web* was tested with the credentials of an administrator.

Vulnerabilities: 9

Hailstorm detected 3 buffer overflow vulnerabilities which were false positives (caused by a slow server response). 5 vulnerabilities were detected by cross site scripting attacks. Hailstorm performed the attacks by attaching a short script injection instead of valid CGI parameters. The URL looked as follows:

```
Hostname/path/subpath/page.aspx?id=>><script>alert(12136191.1177)</script>
```

Important is the closing bracket (‘>’) after the equal sign. This will escape the context tag in which the CGI parameter normally would be used.

Information: 2

The scanner detected IIS 6.0 and ASP.NET as used platforms.

Warnings: 7

Besides the warnings from the vulnerability database (4), the scanner detected four possible buffer overflow vulnerabilities. Those were false positives due to late reaction from the system.
5.6.6 Findings

The testing results on the bare system show, that the underlying concepts and the implementation of SharePoint 2007 give no coherent reason for distrust. As history obviously shows, there is no proof that SharePoint 2007 is completely free of vulnerabilities, as there is no such proof for any complex software. Nevertheless, the tests show, that the out-of-box features withstand the most common attacks and hence provide a solid base even for critical applications.

The issue that gives reason to worry about is that all found vulnerabilities in live systems origin from customized parts, like web parts or custom pages, as the results on the live system shows. Even though, the out-of-box features might cover some of the requirements of a real-life customer project, it will always need extensions that must be programmed by developers to satisfy all business requirements. This will lead to new vulnerabilities if developers are not aware of the risks.

Chapter 6.2.1.1 will try to give ideas for best practices to minimize these risks and to avoid the most common mistakes.
5.7 Logical vulnerabilities

As mentioned before, logical vulnerabilities cannot be detected automatically, but can only be found by manual inspection. During this thesis, some of SharePoint’s logical mechanisms were inspected. Those inspections cannot be comprehensive or cover all logical mechanisms in the system. This is why mechanisms were selected where vulnerabilities would have the most critical impact.

Due to the fact, that none of them turned out to have any detectable vulnerability and hence worked as they are supposed to the discussion is reduced to a short listing of the attempts:

- **API: Trying to connect to another SharePoint farm that is not on the same server by using the SPFarm.Open() statement:** It was ‘hoped’ that this statement could be tricked to connect to remote farm without properly checking the user credentials. The effect would have been that the whole farm could have been mounted by only knowing the server and the name of the configuration database. It turned out, that remote connection is possible but only with user credentials that have the proper rights on the remote SQL server.

- **API: Using web services to get user information:** A small client application was implemented that tried to connect to SharePoint’s web service getUserInfo(string username). The suspicion was that web services were not properly integrated in the access control mechanism. This would mean a backdoor through which operations that needed administrator privileges could have been done without any credentials. It turned out, that the web services use the same methods as the web interface. This means that web services are under the same access control as the web interface.

Similar to the discussion of technical vulnerabilities, logical vulnerabilities have been found in productive systems that origin from administrators of those actual systems. Hence, those logical vulnerabilities were misconfigurations or design flaws by the architects of the specific instances of SharePoint. The remainder of this chapter will deal with some pitfalls in terms of configuration and administration of SharePoint’s security infrastructure that are very common in real-life projects. The problems addressed are followed by recommended countermeasures.
5.7.1 Lack of overview
The security user interface in SharePoint focuses on the secured objects. As known from chapter 4.2.5, a security triple (called a role assignment) consists also of a principle (user or group) and a role definition. If an administrator has to perform tasks that are depending on another of the three dimensions, he has no other possibility than to navigate through all scopes.

5.7.2 Lack of maintainability
The possibility to break security inheritance can lead to a very complicated security structure. On a system that has a rather deep hierarchy of sites, the parent sites will not indicate if there is a child site or list that breaks the inheritance. (The interface only points to the next parent that that breaks the inheritance or that is the top level site.) Hence, the administrator has no way to see which child of the inheritance tree breaks the inheritance unless he navigates to all leafs and traverses from the bottom to the top. In a large system with some dozen top level sites and an overall amount of some hundred sites and lists, this task can take hours.

The following example will demonstrate how the discussed flaws can lead to security leaks.

Example
An administrator has to delete a user X from a team site called team site A. Let’s assume that the team site A has a list L with its own privileges (permission inheritance is broken) and that list is only accessible by a subset of users from team A, including user X.

If now the administrator removes the privileges for user X and site A, the permissions for the list remain. The administrator has no direct indication that user X still has access to any item on site A, unless he navigates to all objects (sub sites, lists and list items) beneath site A and investigates the given permissions. In this case, user X still has access to the list L. Hence, the lack of overview can quickly lead to a security leak.

To solve the problem of insufficient control and overview, a new tool called SharePoint Security Inspector was developed by the writer of this thesis. Its sole purpose is to ease everyday tasks in the domain of SharePoint security administration. The issue of overview is solved by a scope tree that indicates where security inheritance is broken and by a user-centric view that lets administrators select a SharePoint user and displays all rights and scopes on which he has permissions in an easy-to-read table. See chapter 5.8.4 for all details.
5.8 Administration tools
The SharePoint user interface provides only the necessary features for the first steps. A broad palette of tasks needs to be done by other tools. This sub-chapter will discuss all tools that come in handy to administrate security. The next chapter then will point to some applications that help to review a SharePoint system in terms of possible attacks & vulnerabilities.

5.8.1 STSADM.EXE
Ststadm.exe is a command line tool that comes with the basic installation of SharePoint and that can perform all operations that are available in central administration. Some administrative tasks can be solved, that are not possible within the web user interface of a site collection or the central administration. The tool is located in %COMMONPROGRAMFILES%\microsoft\shared\web server extensions\12\bin and has to be run locally on the server. Since the interface is the command line, batch files can be used to run multiple operations at once and hence much faster than through the web interface.

Some of the most frequently used operations concerning security that are not available in central administration:

- **Migrateuser** (migrates user accounts between domains)
- **Updateaccountpasswords** (Updates the Web application pool passwords)
- **Updatefarmcredentials** (Updates the Web application pool for the SharePoint central administration web site)

For more details see [15].

5.8.2 PSCfg.exe
Psconfig.exe is located at %COMMONPROGRAMFILES%\Microsoft Shared\web server extensions\12\bin and is a command line tool similar to stsadm.exe. Its purpose is to configure a SharePoint installation from scratch. It can be used alternatively to install SharePoint through the SharePoint Products and Technologies Configuration Wizard, which is started by default whenever SharePoint is installed on a server. There are 9 operations that need to be performed in the correct order to fully configure a SharePoint installation for an initial setup.

5.8.3 ASP.NET website administration
As soon as form based authentication is needed, the according user accounts will be stored in a dedicated database that is set up through a wizard. To administer these accounts, there is a default web application that can be started through Visual Studio 2005. The way to do this is rather cumbersome:

- Open the www-root of the targeted web application as website in Visual Studio 2005
- Create new default page called default.aspx and open it in design mode.
Security Issues in SharePoint 2007

5. Vulnerabilities & risks

- Select the login control from the toolbar and drop it onto the page.
- Click on “administration” in the control and the administration site pops up.

SharePoint Security Inspector provides a configuration view for custom membership provider that replaces the ASP.NET website administration. See the following section for details.
5.8.4 **SharePoint security inspector**

SharePoint Security Inspector was designed and developed by the writer of this thesis to address logical vulnerabilities in SharePoint 2007 as mentioned in 5.7.1 and 5.7.2. Additionally, the tool replaces the ASP.NET web administration by providing an administration interface covering most tasks concerning forms based authentication.

SharePoint Security Inspector provides an overview to the security concerns of a SharePoint installation. In particular, it provides information in a compact way about

- Users
- Permissions
- Secured objects.

One reason for this tool is the fact that the SharePoint web user interface focuses on the secured objects, such that to find out all permissions for a certain account, a user has to navigate to every securable object and consult its list of permissions. If you are interested what a particular user is able or allowed to do, you need a user focused approach. Especially for anonymous users, which are handled separately from normal user accounts, it is not straightforward to manage their permissions in a comprehensive way.

Further, the break of inheritance in a site collection can occur on site-, subsite-, list- and list item level. This - on one side very flexible - feature brings some complexity in administration tasks with it. If one tries to keep an overview, he will have to navigate through all securable objects. Security Inspector displays broken inheritances in the site structure to give a much better map of the permissions on the secured objects.

For some users, the default permission levels might not be as intuitive as desired. E.g. the differences between ‘restricted read’, ‘view’ or ‘limited access’ might be confusing (See chapter 3 for details about permission levels). To better control if the assigned permission levels fit the principle of least privilege\(^\text{14}\) SI also displays the basic permissions that are assigned with every permission level.

### 5.8.4.1 Components

SI is divided into a group of components. Each component provides a different view on the security landscape of the targeted SharePoint installation.

### 5.8.4.2 User-centric view

Focused on the selected user, the view displays all permissions for that user for all scopes on the whole site structure.

\(^{14}\) “...referring to the concept that all users at all times should run with as few privileges as possible, and also launch applications with as few privileges as possible.” [19]
All role definition compartments (see 4.2.7) are scanned and are displayed as tables with the according scopes as rows in the tables. Scopes where the user has no privileges are displayed as empty and can be hidden. Additionally all groups in which the user is member are displayed. This view gives a comprehensive overview on all privileges the selected user has on the whole site collection.

Figure 11: User-centric view displaying all privileges for a user
5.8.4.3 **Scope-centric view**

The scope-centric view displays a tree of all securable objects. On startup, the tool finds all available web applications. If a web application is selected the application builds up a tree of secured objects, where each node indicates if it breaks inheritance to its parent by displaying a particular ‘flash’ sign. By clicking on a specific node in the tree the application displays all users that have rights assigned to the object. Additionally all base permissions are displayed as well (Remember, that not base permissions but role definitions are assigned to principles, which are groups of base permissions).

![Security Inspector v1.0](image)

**Figure 12:** SharePoint Security Inspector: scope-centric view indicates inheritance breaks with flashes
5.8.4.4 FBA administration

As already discussed in Chapter 3, forms based authentication is an important part for all SharePoint setups that are connected to the internet. Since FBA is external to SharePoint 2007, there is no out-of-the-box way to easily administrate FBA users. SharePoint Security Inspector provides a straight-forward interface that displays all users found and allows creating, deleting or unlocking (in case of too many password trials) FBA users. Also, the change of passwords or e-mail addresses is possible. The membership provider is automatically detected by parsing the web.config file in which the membership provider is defined.

In case of multiple membership providers, the web.config can be changed easily by pointing to it using the ‘settings’ menu. The file will then be parsed again.

Figure 13: Administration interface for forms based authentication
6 Best practices

6.1 Risk analysis for a generic SharePoint 2007 setup

The following chapter summarizes the attacks & tests discussed in the last chapter by providing a proper risk analysis. First, the assets are listed, then the origins of the risks are explained and finally, the risks are classified into 3 levels and then some best practices to avoid the most dangerous ones are presented.

The risk analysis is held generic. This means that we assume that the installation is connected to the internet, has certain objects which require authentication but does not necessarily use specific interfaces to any third party products or have a specific kind of content.

To properly classify the attacks, 2 dimensions are introduced that shall help to anticipate the danger a possible attack represents.

6.1.1 Assets
Before the risks can be anticipated, the assets need to be defined.

6.1.1.1 Primary assets
• User data (names, addresses, passwords)
• Data in secured objects
• Uptime

6.1.1.2 Secondary assets
• Content
• Application code
• Network traffic
• Configuration files (e.g. web.config)

6.1.2 Origins of threats
• Vulnerabilities in supporting software (IIS 6.0, Windows Server 2003 SP2)
• Vulnerabilities in SharePoint 2007
• Vulnerabilities in customized code (web parts, event handlers, web services)
• Missing input validation
• Misconfigurations by administrators
• Disgruntled employees
6.1.3 Classification of damage levels

<table>
<thead>
<tr>
<th>Damage level (D)</th>
<th>Results of a successful attack</th>
</tr>
</thead>
<tbody>
<tr>
<td>High</td>
<td>One or more servers are down, no legitimate user can access the web sites, possible data loss, critical data has leaked, company’s reputation takes damage</td>
</tr>
<tr>
<td>Medium</td>
<td>Some legitimate users cannot access web sites, servers take longer to respond than usual, no critical data has leaked</td>
</tr>
<tr>
<td>Low</td>
<td>No impact or unusual behaviour that does not impact user experience and does not violate data protection.</td>
</tr>
</tbody>
</table>

6.1.4 Classification of probability levels

<table>
<thead>
<tr>
<th>Probability level (P)</th>
<th>Prerequisites for a successful attack</th>
</tr>
</thead>
<tbody>
<tr>
<td>High</td>
<td>Experienced attacker with effort of less than 1 hour</td>
</tr>
<tr>
<td>Medium</td>
<td>Experienced attacker with effort of less than 1 week</td>
</tr>
<tr>
<td>Low</td>
<td>Experienced attacker with effort of more than one week</td>
</tr>
</tbody>
</table>

6.1.5 Calculation of risk level

<table>
<thead>
<tr>
<th>Probability level</th>
<th>Damage level</th>
<th>P High</th>
<th>P Medium</th>
<th>P Low</th>
</tr>
</thead>
<tbody>
<tr>
<td>High</td>
<td>D High</td>
<td>High</td>
<td>High</td>
<td>medium</td>
</tr>
<tr>
<td></td>
<td>D Medium</td>
<td>high</td>
<td>medium</td>
<td>medium</td>
</tr>
<tr>
<td></td>
<td>D Low</td>
<td>medium</td>
<td>medium</td>
<td>low</td>
</tr>
</tbody>
</table>

6.1.6 Attack classification

The following table should give a overview on which attacks actually are a risk to generic systems. This list is not comprehensive, since more attacks can be discovered any time.

<table>
<thead>
<tr>
<th>Name of attack</th>
<th>Level of damage on SharePoint 2007 (D)</th>
<th>Probability of an incident (P)</th>
<th>Risk level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Google hacks</td>
<td>High</td>
<td>Low</td>
<td>Medium</td>
</tr>
<tr>
<td>Source Code &amp; Error Messages</td>
<td>Medium</td>
<td>Low</td>
<td>Medium</td>
</tr>
<tr>
<td>Guessing files &amp;</td>
<td>Low</td>
<td>Low</td>
<td>Low</td>
</tr>
</tbody>
</table>
### Security Issues in SharePoint 2007

#### 6. Best practices

| directories | Bypass restrictions on inputs | Hidden fields | CGI parameters | Cookie poisoning | Session Hijacking | Cross Site Scripting | SQL Injection | Buffer overflows | Canonicalization | Fingerprinting the server | Denial of Service | Breaking Authentication | WDSL scanning attack | Web services: parameter tampering | Web services: XPath Injection attack | Web services: Recursive payload attack | Web services: oversize payload attack |
|-------------|-----------------------------|---------------|----------------|------------------|------------------|---------------------|---------------|------------------|------------------|------------------------|------------------|------------------------|------------------------|-----------------------------|-------------------------|------------------------|
|             | Medium                      | Low           | Medium         | High             | High             | Medium              | High          | Medium           | Medium           | Medium                 | High             | Medium                 | Medium                 | Medium                     | Medium                  | Medium                 |
|             | High                        | Low           | Medium         | High             | Low              | High                | High          | Low              | Low              | Low                    | Medium            | Medium                 | Medium                 | Low                         | Medium                  | Low                    |
|             | Low                         | Medium        | High           | Low              | Medium           | Medium              | High          | Medium           | Medium           | Medium                 | Low              | Medium                 | Medium                 | Medium                     | Medium                  | Medium                 |
|             | Low                         | Medium        | High           | High             | Medium           | Medium              | High          | Medium           | Medium           | Medium                 | High             | Medium                 | Medium                 | Low                         | Medium                  | High                   |
|             | Low                         | Medium        | High           | High             | Medium           | Medium              | High          | Medium           | Medium           | Medium                 | High             | Medium                 | Medium                 | Low                         | Medium                  | Medium                 |
|             | Low                         | Medium        | High           | High             | Medium           | Medium              | High          | Medium           | Medium           | Medium                 | High             | Medium                 | Medium                 | Low                         | Medium                  | Medium                 |
|             | Low                         | Medium        | High           | High             | Medium           | Medium              | High          | Medium           | Medium           | Medium                 | High             | Medium                 | Medium                 | Low                         | Medium                  | Medium                 |
|             | Low                         | Medium        | High           | High             | Medium           | Medium              | High          | Medium           | Medium           | Medium                 | High             | Medium                 | Medium                 | Low                         | Medium                  | Medium                 |
|             | Low                         | Medium        | High           | High             | Medium           | Medium              | High          | Medium           | Medium           | Medium                 | High             | Medium                 | Medium                 | Low                         | Medium                  | Medium                 |
|             | Low                         | Medium        | High           | High             | Medium           | Medium              | High          | Medium           | Medium           | Medium                 | High             | Medium                 | Medium                 | Low                         | Medium                  | Medium                 |
6.1.7 Countermeasures for high risks

**Bypass restrictions on inputs**
All validation has to be implemented on the server side and information from the client cannot be trusted.

**CGI parameters / CGI parameters / SQL injection**
Parameters in the URL and input fields have to be properly validated (e.g. through regular expressions) before they are inserted to the application.

**Denial of service**
Server farms can be made more robust by implementing multiple web frontend servers and an intrusion detection system.

6.1.8 Remarks on risk analysis
As mentioned before, the risk analysis targets a generic system. Keep in mind that a similar risk analysis for an actual productive system might have other assets, other origins of attacks and even give other results, especially in the resulting risk levels.
6.2 Best practices to avoid vulnerabilities

6.2.1 Best practices summary
To avoid most of the attacks discussed above, a collection of best practices is presented.

6.2.1.1 Avoid technical vulnerabilities
1. Implement reliable input validation in reusable modules.
2. Use white list approaches.
3. Establish coding guidelines against injection vulnerabilities.
4. Locate and test all calls to unmanaged code.
5. Use web scanners against own code.

6.2.1.2 Avoid logical vulnerabilities / misconfigurations
1. Security is a design criterion (Easy structure – easy administration).
2. Automatise daily tasks (e.g. addition of users).
3. Assign permissions to groups, never to users directly.

6.2.2 Best practices in detail
The specific parts from the summary above are explained in the remainder of this chapter. Since the actual implementation of the practices leaves many possibilities, only the main idea is presented without going into actual examples or case studies.

6.2.2.1 Reusable modules with reliable input validation
As the test cases showed, a lack of input validation for CGI parameters rendered multiple custom made web parts into a security leak by allowing the execution of java script code passed in with the parameters. To avoid this kind of danger, reusable modules can be implemented once and also easily be changed centrally when new ways of attacks should be discovered. ASP.Net already provides validator controls and regular expression classes, that can be used as a base for an own security framework.

6.2.2.2 White list approaches
When implementing input validation there is always the choice between the black - or white list approach.
A black list approach searches for specific attack patterns and tries to filter input, whenever a pattern matches. Since there might be many patterns, multiple checks have to be performed. Additionally, not all kinds of bad input might be known. Hence, not all attack patterns are known and therefore the validation might still have leaks.

A white list approach reduces the set of possible inputs to only those formats that are expected and all other kinds of inputs are rejected. E.g. if only a name is expected, there is no need to allow digits or even special characters in the input.

Of course, when new attacks are discovered, even a white list pattern might be subject to optimization. But the probability that a change is really necessary is much smaller.

6.2.2.3 Coding guidelines
Coding guidelines exist in most companies, although they almost never include security related rules. Especially junior developers might not be fully aware of the risks a website is exposed to. This is where coding guidelines can avoid common mistakes. Tools like Jetbrains’ Resharper [16] for Visual Studio allow the implementation of custom coding guidelines, such that they can be automatically verified at design time.

6.2.2.4 Buffer overflows
One of the oldest vulnerabilities is the buffer overflow. Today, programming languages like C# or java, manage code such that the memory stack can’t be violated. But some calls, even in the .NET Framework, still execute native code. Since native code runs unmanaged, buffer overflows might still occur. Hence it is important to know when native code is called and those calls must be checked for overflow vulnerabilities. Now, it seems to be unfeasible to know all calls to native code in the .NET-framework. A more practical approach is to fill all input fields in the application under inspection with as much data as possible (> 100 thousand characters) and check if the system still behaves correctly. If it is a real buffer overflow, the system might even crash. (You might not want to check this directly on your customer’s productive system but on a copy of it.)

6.2.2.5 Use of web scanners
As mentioned during the tests, even a small SharePoint system consists of many sites and pages that need to be checked. For an experienced hacker, these checks might still be feasible to be done manually. For everyone else, web scanners are helpful tools that also produce very comprehensive reports. Unfortunately most of them are very expensive (e.g. IBM Watchfire [17]).

6.2.2.6 Security as a design criterion
In most cases, security is regarded as the last and very optional step in the lifecycle of a software project. This leads to unpleasant surprises, when the security structure of an implementation gets more and more complicated, the maintainability suffers. This is why security should already be considered in the design phase. Simplicity and maintainability should be maximized.
6.2.2.7 Automatization of daily tasks

Security administration as it comes out-of-the-box often requires multiple steps for very common tasks. For instance, the addition of a new external user that can access the site collection through forms based authentication requires the creation of the new user in custom membership provider and the addition of user to all necessary SharePoint groups. A group assignment can be forgotten very easily. This is why custom tools (web parts or winform applications) should be provided for administrators that reduce the risk of misconfigurations. Since the API of SharePoint is well implemented, the necessary development efforts are relatively low and mainly consist of the implementation of the user interface, while the background operations in most cases already exist.

6.2.2.8 Assignment of permissions

There is no obvious reason to assign role definitions to a user directly. It complicates the security administration and can easily lead to a security landscape that is not maintainable any more. Whenever possible, try to assign user to groups and permissions to those groups.

6.2.2.9 Gain necessary overview

The pseudo-inheritance for role definitions and role assignments is just one cause for the lack of overview in SharePoint’s security administration. Another one is the lack of the necessary views in the initial set of web user interfaces. It is important to have all necessary information at your fingertips in a glance. SharePoint Security Inspector is one way, to provide the necessary information. See chapter 2 for more details about Security Inspector.
7 Future work

As presented in the best practices chapter, the automatization of security tasks will reduce errors made by the administrators. The security inspector tool could be easily extended to include such tasks, as for example to clone a user rights and assign them to another user which has to have exactly the same privileges.

In its current version, Security Inspector runs only locally on a SharePoint Server. With relatively small effort, the functionality could be included in web parts that would allow execution over the web without using the remote desktop protocol.
8 Bibliography


