Secure Wiki System

A plugin-based solution to wiki security

Written by

Kasper Lindberg - s052257

Technical University of Denmark
Department of Informatics and Mathematical Modelling
Master's Thesis - IMM-M.Sc.-2012-28
March 2 - 2012

Abstract

Wiki systems have become an important tool for knowledge sharing among people. From the small wikis for knowledge sharing in organizations to the larger project-related wikis on the Internet. In addition, Wikipedia, which is in a class of its own when it comes to size, has managed to collect an impressive amount of information based solely on the cooperation between strangers from around the world. Any open wiki, with a user-community so large that members of the community have a certain degree of anonymity, suffer from the effects of directed and random vandalism. This vandalism is a problem because it reduces the trustworthiness of the content provided by the wiki system.

The secure wiki model is an integrity model that has been proposed to help prevent vandalism and improve the trustworthiness of articles in wiki system. This model is based on both static and dynamic document access controls, which enforce a simple integrity based security policy. This thesis improves this model by proposing a new policy for use with the model. The proposed policy is evaluated and compared to the original policy. The evaluation shows that the new policy is highly configurable and can be configured in such a way that it requires significantly less reviewers than the original policy, which can benefit small systems with a low number of users.

An implementation of a base wiki system have been created, which on its own equals any other wiki in terms of its vulnerability to vandalism. In addition to this, an implementation of the secure wiki model has also been created. The implementation is made as a plugin to the base wiki system and adds an integrity model to the existing soft-security model that is used by the base system and other wiki implementations. The integrity model provides harder security guarantees and limits the ability of attackers to compromise the integrity of wiki articles, without compromising the *all can edit* policy of open wiki systems.

Preface

This thesis was prepared at the Department of Informatics and Mathematical Modelling at the Technical University of Denmark in partial fulfillment of the requirements for acquiring the M.Sc. degree in Computer Science and Engineering.

The thesis deals with preserving integrity of documents in wiki systems using a previously defined integrity model for wiki systems. The main focus is on the design and implementation of the specific policy used in this model and the implementation of a proof-of-concept.

The project was completed in the period from October 3rd, 2011 to March 2nd, 2012 under the supervision of Associate Professor Christian Damsgaard Jensen.

Lyngby, March 2012

Kasper Lindberg

Contents

List of Figures

List of Tables

List of Listings

1 Introduction

The Internet started as a means for researches at various American universities to exchange data. In the early 1990's HTML was created by researches at CERN [3] to make sharing of knowledge easier. Over time, the use of the Internet for knowledge sharing have evolved to be much more. Online newspapers compete to be the first to publish news-stories, people from all around the world get together to work on open source projects and some engage in the creation of online encyclopedias. The dynamic nature of the Internet and the ease of accessing it has made the Internet an important source of information for everyone connected to it. The ease at which information can be published on the Internet has increased the sharing of information but, at the same time, also increased the risk of getting bad information, even from well-meaning authors. Some of the collaborative knowledge gathering and sharing systems on the Internet are Wikipedia and Citizendium. Both allow everyone to edit existing articles and create new articles, although Citizendium requires editors to register an account tied to a known and verified real-world identity. In addition to these, everyone can start wikis on any subject on wikia.com. These and other collaborative systems are using the concept of crowdsourcing, i.e. using the combined efforts of a large crowd of people to accomplish a task, to gather knowledge within the scope of the system. Wikipedia, Citizendium and wikia are all based on the wiki technology. Wikis are open systems where anyone can edit everything which puts any wiki at risk of being vandalized by malicious users. Vandalism ranges from the undirected vandalism performed out of boredom to the more directed vandalism performed with specific intentions and therefore the motivation to work around protection measures put up by the wiki. In addition to vandalism, bad edits by well meaning users may introduce errors in the wiki articles, that can be hard to detect by readers of the article. The open nature of a wiki means that the reader of wiki articles must be aware that a number of caveats related to the writing process [5]. The most significant being

Accuracy The information contained in an article may be incorrect, without the possibility to distinguish good from bad.

Motives The article may have been written with a specific point-of-view for commercial, political or religious reasons.

Author expertise The expertise of the author or authors who wrote the article may be low. This can lead to the inclusion of speculation and rumors in the article.

Despite the issues with the open nature of a wiki, wikis have proven to be a popular and successful tool for knowledge sharing. The success of the wiki idea can be illustrated by the Wikipedia - *the free encyclopedia that anyone can edit*, which aims to cover just about anything, smaller subject specific wikis at wikia.com and company and department wikis having only a small group of people as users.

A study made by Wikipedia contributors concluded that 4-6% of edits made to Wikipedia was vandalism [1, 22]. Although the sample-size used in the study is small, the number, if assumed to be representative of Wikipedia, shows that the amount of vandalism that Wikipedia receives is significantly lower than what could have been feared, given the open nature of the system. In the month of December 2011, close to 5.5 million edits were made to the English

Wikipedia[1]. Given the large amount of edits made, 4-6% is still a large amount of vandalism to correct.

To combat vandalism and correct errors by well meaning authors, Wikipedia contributors regularly monitor recent changes to Wikipedia and checks the edits made in order to detect and correct any problems [24]. In addition, automated bots have been deployed to monitor changes to detect and correct vandalism [23]. With 4-6% vandalism, the need to do this is apparent, but the time spent doing this is a waste of time that could go into researching and writing articles.

In an attempt to reduce the amount of vandalism, basic vandalism detection have been added to the Wikipedia software. As an example, when attempting to create a user-page filled with garbage[2], Wikipedia will, at the time of writing, responded with the message *"An automated filter has identified this edit as potentially unconstructive, and it has been disallowed. If this edit is constructive, please report this error."*

The soft-security model used by Wikipedia have proven itself to be sufficiently effective in keeping Wikipedia relatively vandalism free. The success of this model is that most vandals are demotivated by the quick removal of their vandalism. To combat the remaining vandalism that does occur, new models providing harder security guarantees are needed. The challenge for these models will be to stay within the open nature of the wiki, which requires the system to be managed by the community. If such a system can remove all vandalism and at the same time require significantly less time to manage than what would have been spent correcting vandalism, it can be considered a success.

The following will describe the secure wiki model [12] that aims to protect Open Collaborative Authoring Systems (OCAS), of which wikis are a subset, against vandalism. The model does not prevent legitimate contributors from contributing to a wiki system, but puts a cost on the ability to edit articles in the system. For individual contributors this cost is negligible, but for attackers this cost makes it costly to control the multiple colluding malicious users needed to break the security of the model. Controlling multiple colluding malicious users is known as a Sybil-attack [6]. The model also puts a cost on whitewashing [8] where users change accounts to escape the bad reputation that the account has gained through malicious actions.

This thesis documents the proof-of-concept implementation of the secure wiki model and the modifications to the model identified as beneficial wrt. having a model that can be accepted by administrators and users of wiki systems. The implementation uses an alternative policy than the one proposed by the original model, and compares this new policy to the original policy wrt. the amount of work required by attackers to violate the policies.

The results produced by the thesis shows that the secure wiki model is an effective model that can protect articles from vandalism and can be tailored to fit most systems, regardless of their size. The evaluation shows that the work required to violate the new policy is sufficiently high to make it infeasible for most attackers to break the security of the model and that the larger the system the harder it is to break. The system has yet to go through a full-scale evaluation in a real-life environment with both honest users and malicious colluding users attempting to break the security of the system.

[1]First edit on December 1 2011 created revision 463382251, Last edit on December 31 2011 created revision 468850236. Difference: 5,467,985 revisions.

[2]specifically the content: *asdfasdf asdfa sdfasdfasd fasd fasdfasd fad*

1.1 Objectives

Wiki systems allow groups of people to share knowledge, allowing individual members to obtain the knowledge resulting from other individuals prior research. The danger is that the articles in the wiki are not accurate, complete and unbiased. Current wiki systems uses citations, much like scientific articles, to allow readers to verify the claims that are presented in an article. Depending on whether a user checks the references and if the reference is any good, there is still the chance that users can be misinformed by errors, omissions and bias in the article.

The goal of the project is to create a proof-of-concept wiki system that uses a plug-in based approach to integrity and by that approach, improve the ability of wiki systems to ensure the qualifications of contributors, in relation to the quality of article these contributors edit and to better communicate the accuracy, completeness and lack of bias of articles to readers.

This thesis will present a brief summary of some of the current approaches to integrity in existing wiki systems. The thesis will describe why a new system is needed and present the secure wiki model that addresses these problems. The main goals of the thesis is:

- Propose a new policy for use in the secure wiki system and evaluate its effectiveness compared to the original policy.
- Implement a base wiki system with a plug-in architecture that allows the seamless exchange of the secure wiki model implementation.
- Implement a plug-in version of the secure wiki model that are using the proposed policy and can be configured to match the specific system.

1.2 Thesis Structure

Previous work in Section 2 describes a number of approaches that are currently used to prevent vandalism and/or provide a measure of the integrity of wiki documents. The section also introduces the secure wiki model, which forms the basis for the work done in this thesis, and describes the previous implementations and the experience gathered from these.

The analysis in Section 3 describes the analysis of a generic wiki, the secure wiki model in the context of a generic wiki and analyses the requirements for the policy to use with the secure wiki model.

Based on this analysis, Section 4 describes the design of a base wiki and the secure wiki model to be used with it. The design section also defines a new policy to use with the secure wiki model. The implementation of this system is documented in Section 5.

The evaluation in Section 6 establishes a guideline for the specific values used by the proposed policy, evaluates the proposed policy and shows how it compares to the policy used by the original secure wiki model.

Future work in Section 7 describes possible future extensions of the implemented system, particularly with respect to the secure wiki model and the proposed policy.

The conclusion of the thesis is presented in Section 8 and describes the overall results of the thesis along with closing remarks.

1.3 Terms and Definitions

The following list defines the special terms used throughout this thesis.

Reader A wiki user that only reads the content on the wiki, but does not write or contribute to the wiki.

Contributor A wiki user that contributes to the wiki by writing content, fixing errors and performs administrative task associated with the running of the wiki.

Review A procedure used for assessing the qualifications of a contributor, based on the contributor's contributions to the wiki.

Reviewer A wiki-user chosen to participate in a review.

2 Previous Work

Addressing wiki security and the trustworthiness of articles have previously been done using differing approaches. Some of these approaches have been specifically designed for wiki systems while some can be applied to any set of objects. The classic wiki security, used by most wiki implementations have proven itself to be sufficient in the creation and maintenance of both small and large systems, provided an active community of people supporting the many manual operations needed to support it. Although this system have proven effective, the fact remains that it is not sufficient to rule out the presence of incorrect information added by vandals.

In order to provide users with a measure of the level of trustworthiness for an article, both content-based and collaborative filtering approaches have been taken. Content-based filtering systems uses only the content that can be obtained from the running system, while collaborative filtering systems rely on input received from users.

Building on the behaviour of the classic wiki security mechanism, the Wikitrust system can calculate a trust-score of a wiki article, based on the reputation of the authors that contributed to the article. Trust-values and author reputation is based solely on the information retrieved from article revisions and is therefore a content-based system. At the moment, Wikitrust is active for only a handful of language-specific Wikipedia wikis. The Wikitrust system requires no human interaction but acts exclusively on the content provided by the WikiMedia foundation servers running Wikipedia.

Where Wikitrust is content-based filtering, the Wikipedia Recommender System (WRS) uses a collaborative filtering approach, that allows WRS to predict the users rating for articles that the reader have not seen before. Readers are asked to rate articles, which are then compared to ratings by other users, which can then be used to identify users with similar rating-patterns and predict the reader's rating of new articles. When no data is available, WRS uses the WikiTrust system as a fall-back to ensure that WRS will always be able to give a rating.

These approaches are all based on providing a trust-value for the article as it is when reading the article, but does not have the ability to prevent bad content from being added to the articles. This is the goal of the secure wiki model, which establishes rules for who can edit which articles, with the intention of protecting high-quality articles from low-quality contributors and all articles from vandals.

2.1 Classic Wiki Security

Wiki systems rely on soft security where repairing the system is as easy as, or easier than, creating the problem. The basic problems that any wiki needs to handle is the introduction

of bad content into articles and deletion of good content from existing articles. Introduction of bad content can come from both malicious and well-meaning users. Well-meaning users will edit articles in a way that they believe improves the article. Their edit will usually add factually incorrect content or maybe delete good content. Some malicious users will add content that is completely unrelated to an article subject, simply to vandalize the article, while other malicious users may have a specific purpose with their malicious edit. Articles dealing with controversial topics, specifically those that contradict a belief or conviction held to be true despite clear evidence to the contrary, e.g. conspiracy theories, are targets for directed vandalism.

In general, wikis deal with malicious edits by undoing the edit to restore the article to the previous good revision. In small wikis where contributors know each other outside of the wiki and where contributors can communicate through the wiki system, a contributor that made a bad edit can be told so. Well-meaning contributors will take such a message into consideration next time the contributor makes an edit. Malicious users may realize that their actions are disruptive and stop, but the ones that continue, can be blocked from the system. Since newly created accounts can do the same as old accounts, the cost of being blocked is relatively low.

Wikipedia is the biggest and best known wiki. For this reason Wikipedia also attracts malicious users with the intention to vandalize Wikipedia articles. The size of Wikipedia, both in terms of number of articles and the range of topics covered by those articles, means that Wikipedia has extended its vandalism prevention features beyond that of a generic wiki and may be the most advanced vandalism prevention used by any wiki. In addition to the normal strategy of vandalism detection and correction with an optional block if the contributor repeatedly vandalizes articles, Wikipedia has introduced measures to protect articles from vandalism by preventing edits from certain users.

Wikipedia uses several types of article protection schemes, depending on the protection needed. *Full protection* locks an article such that only administrators can edit the article while *Semi-protection* locks an article such that new and unexperienced contributors cannot edit the article. The specific condition used to determine if a contributor is new and unexperienced is that the account should be more than four days old, and have made more than 10 edits[3] [25]. A contributor meeting this criteria is said to be autoconfirmed. Using the semi-protection, Wikipedia is able to prevent anonymous malicious edits intended at changing specific articles. In addition, any autoconfirmed account can be blocked from editing, requiring the user to create a new account and invest the time and effort needed to be autoconfirmed again. This effectively puts a cost on being blocked, when engaging in directed vandalism. In addition, Wikipedia has *creation-protection* to protect articles from being created (usually used to prevent the recreation of a deleted article) and *move protection* to prevent the moving (renaming) of articles. The need for full protection arise when a divided community engages in edit-wars and repeatedly changes an article between two or more opinions. To resolve such disputes, an administrator will enforce a full-protection of the article, forcing the community to discuss the issue on the article talk-page. When the community reaches a consensus, the article can be unprotected and the administrator can enforce the consensus by blocking contributors refusing to follow the consensus of the community. In wikipedia terminology, an administrator is a user that have been chosen from the community by the community to receive additional permissions. Out of the English Wikipedia's 15.9 million registered users, only about 1500 are administrators [20]. In addition to administrators, Wikipedia has a small number of bureaucrats with more permissions than administrators, e.g. it requires a bureaucrat to promote a user to administrator. There

[3]The condition for contributors editing through the Tor network is 90 days and 100 edits

are around 30 bureaucrats on the English Wikipedia [21].

Despite the fact that preventing certain users from editing specific articles goes against the policy of *anyone can edit*, the additional vandalism protection features introduced by Wikipedia has helped combat vandalism and edit-warring by both anonymous and registered users. Full and semi protection in Wikipedia is applied only for the time needed and as soon as the protection is no longer required it is removed from the article. Semi protection is especially effective since regular contributors have no problem being autoconfirmed, allowing them to edit semi-protected articles, while vandalism is usually performed by anonymous and new users [1]. New or anonymous users can still edit talk-pages, so if they have a contribution to make, they can always ask to have their contribution added to the article by another contributor.

2.2 WikiTrust

Wikitrust is a system developed by researchers at UC Santa Cruz, Computer Science Department. Wikitrust is a system that computes the reputation of contributors (system-internal use only) and a trust value for articles [1]. Wikitrust is a content-based reputation system and requires no direct interaction with wikipedia users.

Wikitrust tracks changes made to articles and can remember when individual words were added to an article. When an author modifies an article, Wikitrust considers the author to have performed a review of the existing text and therefore, any text that survives the review has the authors approval while text that is removed by the author is not approved by the author. Wikitrust computes this on a word-by-word basis, and can recognize when a section of text is reworked compared to being replaced. For each review of an article, the trust placed in the words that survived the review is increased by a value dependent on the reputation of the author performing the review. Authors gain reputation by writing content that survives reviews by other authors and their gain in reputation is dependent on the reviewing author. When high-reputation authors write content, the words they write are automatically assigned a trust-value corresponding to the reputation of the author. Anonymous and new contributors have gained no reputation while older and more experienced authors have gained reputation on the basis of their contributions. With vandalism primarily being performed by anonymous and new contributors [1] this effectively marks vandalism as low-trust, while the contributions made by the reputable contributor are marked as high-trust, even when the added text have not been reviewed by other authors.

To use Wikitrust, the user can install an addon for the Firefox web browser, which will modify Wikipedia pages and add a tab that, when clicked, will load Wikitrust trust-values for the article. When using the addon to get the Wikitrust trust-values for an article, the addon will modify the background-color of the text to indicate the trust-score of each word. Text that Wikitrust finds to be trustworthy maintains its white background, while new and untrustworthy text has an orange background. The shade is a dark orange for untrustworthy text and gets progressively lighter as the trust in the text increases with each review performed by high-reputation authors.

The result of using Wikitrust is that the trustworthiness of articles will be directly affected by the majority opinion in the community. If the community writing articles all agree on a specific, but incorrect, view of things, the community will maintain articles to that viewpoint. Since the community does not revert this incorrect point of view, Wikitrust will assume authors to have a high reputation and therefore the articles describing this point of view will be marked

as trustworthy. However, this will be common to all systems relying on the actions of the community and not an authoritative third party. Since an authoritative third party is unlikely to be available, this is something that such systems will be affected by.

2.3 Wikipedia Recommender System

The Wikipedia Recommender System is a collaborative filtering approach to providing users with an article-rating of Wikipedia articles [13]. The system collects user ratings of articles and stores them in a centralized database managed by the WRS server. When a user reads an article from Wikipedia, WRS will compare the user's ratings with ratings given by other users for articles which the users have already rated. By comparing the ratings, WRS can identify other users with similar ratings. Assuming the other users with similar ratings have rated the article that the user is reading, WRS can use these ratings to predict the rating that the user would assign to the article. When no ratings are available for reference, WRS comes short since no data is available to base a prediction on. In this situation, WRS uses Wikitrust as a fallback to provide an article-rating. Where Wikitrust computes ratings on a word-by-word basis, WRS provides a single rating for the entire article. Mapping from Wikitrust ratings to WRS ratings is done using a weighted average of the Wikitrust ratings and a subsequent normalization to move the rating from the interval $[0 : 10]$ used by Wikitrust to the interval $[1 : 9]$ used by WRS [13].

Compared to Wikitrust that calculated a consistent rating regardless of the user requesting a rating, the Wikipedia recommender system can report differing ratings to different users if the users disagree on their ratings. A user who rates articles in much the same way as the majority of users in WRS will receive the ratings given by the majority of WRS users. Another user who consistently rates articles contrary to the majority of WRS users will get ratings contrary to the ratings of the majority. Despite this difference, both users receive the rating that they would have given the article.

2.4 The Secure Wiki Model

The primary function of both Wikitrust and WRS is to establish the trustworthiness of a Wikipedia article as it is, with all the possible inaccuracies and vandalism contained in the article. Both systems lacks the ability to prevent inaccuracies and vandalism from entering the article in the first place. The classical wiki security features can do this, but does so only on specific articles and only when it is clear that the article would otherwise be vandalized and the protection will be removed as soon as it is perceived that the threat of vandalism is gone.

The secure wiki model, suggested by Jensen [12] is an integrity model for use in Open Collaborative Authoring Systems (OCAS), which wiki systems are a subset of. The purpose of the secure wiki model is to provide a classification of articles in terms of their correctness, accuracy and lack of bias. To provide a reliable classification, the secure wiki model defines a number of constraints that control who can edit specific articles.

Traditional wikis are lacking effective means for supporting, the traditional security process based on *prevention*, *detection* and *response*. This is, in part, due to the fact that few prevention mechanisms exists, detection is a manual process left to the community and the only response is to undo the undesirable changes. For this reason, the secure wiki model combines the reputation-system approach of deciding what is good and bad content with traditional integrity

mechanisms to prevent undesirable content from entering good content. The secure wiki model works by enforcing integrity levels for authors and articles in a manner based on the Biba integrity model [4] and Biba's Low Watermark model [4]. The secure wiki model also defines the process with which to manage the integrity levels.

The Biba integrity model defines the Simple Security Property and the *-Property

1. Simple Security property; A subject at a given level of integrity may not read an object at a lower integrity level (*No Read Down*)

2. *-Property; A subject at a given level of integrity must not write to any object at a higher level of integrity (*No Write Up*)

The user may have multiple programs open, e.g. web-browsers, so the simple security property will be impossible to enforce in practice due to the lack of complete mediation of the user's access to information. The secure wiki model therefore only enforces the *-property, which it can, due to the fact that data written to the system will have to go through the server-side of the OCAS system.

Each author in the OCAS-system registers an identifier to be used when editing documents. The identifier serves as a pseudo-identity for the real author and enables the OCAS-system to connect the author's actions regardless of the point of access (such as home or work PC) from where the author edits documents. The secure wiki model assigns Integrity Levels (IL) to documents and Quality Confidence Values (QCV) to authors through their pseudo-identity identifier. The IL and the QCV describes the level of correctness, completeness and lack of bias in documents and in the authors previous writings. An author is assumed to be consistent in his writings and therefore the QCV is also considered an indication of the quality of the writings the author will produce in the future.

The Secure Wiki Model builds on two separate integrity models, each intended to handle different aspects of the mechanism that ensures the integrity of the system. The first is the static integrity part, which enforces access control and prevents unauthorized editing of articles. The second part is the dynamic integrity model, which manages the assigned integrity levels of articles and authors.

2.4.1 Static Integrity Model

The IL and QCV are used in the static access control, to ensure that the *-property is enforced, such that authors are not able to corrupt documents with a higher integrity level. Formally, the secure wiki model defines the following sets:

$\mathbb{A}$ is the set of identifiers of authors who have registered to use the system.

$\mathbb{D}$ is the set of documents that are managed by the system.

$\mathbb{I}$ is a totally ordered set of integrity levels.

In order to compare the IL and the QCV, the model defines the following two functions, that maps the IL and QCV to the integrity levels in the totally ordered set $\mathbb{I}$.

$qcv : \mathbb{A} \to \mathbb{I}$ which gives the quality confidence value of the author $a \in \mathbb{A}$.

$il : \mathbb{D} \to \mathbb{I}$ which gives the integrity level of the document $d \in \mathbb{D}$.

These functions are then used to define the predicate

$can\_edit : \mathbb{A} \times \mathbb{D} \rightarrow \{0, 1\}$ which indicates if author $a \in \mathbb{A}$ can edit document $d \in \mathbb{D}$.

The formal definition of *can_edit* is shown in (1).

$$can\_edit(a : \mathbb{A}, d : \mathbb{D}) = \begin{cases} 1 & \text{if } il(d) \leq qcv(a) \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

The *can_edit* function is evaluated each time an author attempts to edit a document. The function prevents authors from editing documents with an IL higher than the authors QCV. This effectively enforces the *-property of the Biba integrity model, in that it prevents authors from writing to articles with an IL higher than the authors QCV.

2.4.2 Dynamic Integrity Model

In addition to the static integrity model, the secure wiki model defines a dynamic integrity model to manage the integrity levels assigned to documents and authors. Inspired by the low-watermark model [4], the secure wiki model will automatically raise the IL of a document to the QCV of the author that edits the document. This is done based on the assumption that authors will improve the quality of documents they edit, to their QCV. The effect is the opposite of the original low-watermark model.

The original low-watermark model says that when a subject accesses (i.e, reads) an object with an integrity level lower than the subject, the subject will be given the integrity level of the object. The secure wiki model turns this around, such that when the object (the document) is written by the subject (the contributor) the document is given the integrity level of the contributor. This is based on the assumption that authors with higher QCVs will improve articles to the point where they deserve the higher IL.

In addition to having good authors improving documents, whose IL is lower than the author's QCV, and thereby increasing the document's IL, users with a QCV equal to the document IL can also improve documents to the point where the documents deserve a higher IL. Since the QCV of the author equals the IL of the document, another method is needed to raise the IL of the document. For this, the secure wiki model uses a promotion procedure based on users reviewing the document and casting votes to determine consensus on the quality of the document. The secure wiki model rewards such authors by promoting the *principal author* of documents that are promoted through reviews. The secure wiki model uses the term *principal author* to mean the author who contributed the most, but it could also be the author who contributed last or the contributing author who requested the promotion review or a combination of factors [12].

In order to reduce the ability of an attacker to perform a denial of service attack through repeated promotion-requests, the secure wiki model limits the set of authors who can request such a review to the authors who contributed to the document.

The secure wiki model also has a demotion procedure that will allow the community to reduce the IL of documents that no longer meet the criteria for the given IL. Demotion can be necessary if the general quality of the documents at the given IL improves without similar improvements being made to the document that is to be demoted. If a demotion procedure results in the demotion of a document, the principal author will also be demoted. The secure wiki model limits the set of authors who can request a demotion review to the authors who can edit the document according to the *can_edit* predicate.

The secure wiki model defines the following notation for the purpose of describing the review policy used in a review procedure.

L_i is the symbolic name used for the i^{th} integrity level in $\mathbb{I}$.

Λ_i is the set of registered users at level L_i.

r_i is the number of authors randomly selected from level L_i to perform a review.

Λ_{R_i} is the set of randomly selected users from level L_i to perform a review.

τ_i is the threshold of votes necessary for level L_i to return a positive result.

z_i is the number of malicious colluding authors in Λ_{R_i}.

Using these definitions, the review decision for reviewer $j \in \Lambda_{R_i}$ on the review of document $\mathrm{d} \in \mathbb{D}$ is then $\delta_j(\mathrm{d}{:}\mathbb{D})$. The combined judgment of level L_i is then defined as shown in (2)

$$\mathcal{D}_i(d : \mathbb{D}) = \begin{cases} 1 & \text{if } \sum_{j \in \Lambda_{R_i}} \delta_j(d) \geq \tau_i \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

The secure wiki model uses a policy termed Π_1. Using Π_1, two out of three levels must approve the promotion of a document in order for the promotion review to succeed. To have a document promoted from level L_i to L_{i+1}, Π_1 dictates that each of the levels L_i, L_{i+1} and L_{i+2} must review the document and at least two of them vote to promote, as expressed in (3).

$$D(d : \mathbb{D}) = (\mathcal{D}_{i+2}(d) \wedge \mathcal{D}_{i+1}(d)) \vee (\mathcal{D}_{i+1}(d) \wedge \mathcal{D}_i(d)) \vee (\mathcal{D}_i(d) \wedge \mathcal{D}_{i+2}(d)) \tag{3}$$

2.5 Implementations of the Secure Wiki Model

As part of the description of the secure wiki model, the effectiveness of the model was evaluated using theoretical examples and calculations to determine the probability of an attacker's ability to influence reviews. Having shown that the model is sound, when subjected to various scenarios, the next step in evaluating the effectiveness of the secure wiki model is to have the model implemented in a real-life system. Two such implementations have been made using the JamWiki system as base. JamWiki is an open-source implementation of a wiki using the same article-markup as MediaWiki and thus Wikipedia. JamWiki is licensed under LGPL and runs on most application servers [11].

2.5.1 Secure Wiki Model in JamWiki

As a first prototype implementation, Sander [15] has implemented the secure wiki model in JamWiki. The first prototype implementation contains small extensions to the original secure wiki model that gives authors a finer control of the document IL. These extensions are based on a number of observations. First, when reviewing articles, the system needs to lock articles from modification for the duration of the review and that this is a problem as it prevents authors from improving the article further while locked. Second, the watermark model, as used in the original secure wiki model, does not support the work-process of the typical wiki-contributor.

A contributor with a high QCV with the intention of improving the quality and thereby the IL of a long article may rewrite the whole thing offline before submitting the changes in one go, but due to the time-consuming process of improving a long article, it is more likely, that the improvements will be made in steps, e.g. one section at a time. This means that just because the article were modified by a contributor with a high QCV, the article may not deserve an IL matching the contributor's QCV. This is a problem since automatically determining the appropriate IL of the revision is difficult. With the need to lock articles during reviews and the problem with determining the appropriate IL when articles are edited by contributors with a higher QCV, the first prototype implementation asks the author what the appropriate IL of the article should be, with choices ranging from the article's IL to the contributor's QCV. Effectively, this changes the secure wiki model from automatic document promotion to manual document promotion.

The same considerations are made in relation to demotion, where the actual demotion is managed by administrators. Administrators are trusted to only demote articles for which there is a consensus to demote in the community. This consensus would be reached through a discussion on the article's talk-page.

The first prototype implementation relies solely on individual authors and administrators to promote and demote articles. Author QCV is managed by the community using a configurable voting policy. Reviews of authors are based on the individual author's contributions to the wiki. These contributions can easily be identified using the user-contribution function available in JamWiki and most other wikis.

Using JamWiki as a base system, the prototype implementation relies on existing features to inform the user of the IL of the given article. Specifically the implementation places the classification of the article as part of the page-structure above the article-content. As far as voting is concerned, no existing features exist in JamWiki that can be used. For this, the prototype implementation has created separate pages that facilitate easy access to all necessary information in order for the reviewers to reach a decision.

The prototype implementation uses a configuration file that allows system-administrators to change values for the parameters used to configure the voting-policies. In addition, the prototype implementation has an interface for the plugin that allows anyone to create a new secure wiki model implementation that can replace the secure wiki model of the prototype implementation, if the new policy to be used is different from the policies that can be configured in the prototype implementation.

The first prototype implementation is a system that assigns ILs to articles and QCVs to contributors as well as enforcing the *can_edit* predicate defined by the secure wiki model. In addition the prototype implementation can manipulate the ILs of articles and QCVs of contributors. The prototype implementation has identified a number of issues with the secure wiki model that needs to be resolved before the secure wiki model can be applied to wikis in general use. These issues relate primarily to the difference between the model's and wiki-contributors' method of doing things. The prototype implementation has solved these issues in a straight-forward way by giving a finer control of document IL to authors.

2.5.2 Secure Wiki Model Plugin in JamWiki

A second implementation, made by Følsgaard and Ludwigs [9], focused on making a more modularized implementation in order to extend the ability of the wiki system to be configured to use alternative review policies. The second implementation is based on the first prototype

implementation and the Open Services Gateway Initiative (OSGi) framework that enables java applications to dynamically load classes and services from bundles added to the system at runtime. OSGi will be described further in section 2.6. The second implementation will be termed the OSGi-implementation.

The easy integration between OSGi and java based web applications makes the OSGi-framework suited for the task of providing the necessary plugin-capabilities. With OSGi being able to do this at runtime without having to restart the web-application itself means that downtime of large websites can be avoided. This fits perfectly with the plugin-framework-requirements that the OSGi-implementation has.

The OSGi-implementation runs on the SpringSource Dynamic-Module server (DM-server) instead of the Tomcat server used by the first prototype implementation. The advantages of using the DM-server is that the DM-server is based on the OSGi-framework and tailored for Spring-module based web applications. In addition, the OSGi-framework used by the DM-server can be accessed and used by web applications running in the DM-server.

The OSGi-implementation is based on the first prototype implmentation of the secure wiki model but has successfully taken the first prototype implementation and extended it to make it simpler to change the secure wiki model plugin, such that any new policy, not originally supported, can be added to a running system.

The OSGi-implementation uses the OSGi-framework concept of services, which allows the implementation to work on abstract services defined by an interface. The use of an interface means that the code itself can implement the flow of operations and call the services and have them perform the plugin-specific operations, such as the evaluation of a completed review. One plugin may accept the review, where another would determine that the review were unsuccessful. In addition to services, the OSGi-implementation also makes use of the ability to split an application into multiple modules (or bundles in the OSGi terminology). Using multiple bundles allows the system (or parts of it) to be updated to a new version, without taking down the system.

2.5.3 Summary

The first prototype implementation was made by directly modifying the source code of the official JamWiki. In terms of updating the system, the system is very monolithic in the sense that the secure wiki model plugin needs to be reimplemented in each new version of JamWiki. The OSGi-implementation has extracted the secure wiki model plugin to a separate bundle, allowing the rest of the system to be updated, needing only to add back the hooks that allows the secure wiki model plugin to work. When, at some point, the JamWiki system implements a plugin-framework, the OSGi-implementation should be able to be used after an initial modification to fit into the future JamWiki plugin-API.

The changes made by the prototype implementation extends the JamWiki functionality by adding the secure wiki model functionality. The OSGi-implementation adds little functionality compared to the prototype, but instead restructures the implementation to make it easier for administrators to switch secure wiki model implementations and policies.

2.6 OSGi

This project will build on the experiences documented in the theses describing the the first prototype and the OSGi-implementation. Most can be referenced directly, where relevant, but

since the OSGi-framework will be a key part of the project and is not familiar to most, the OSGi-framework concepts and techniques used in this project will be detailed in this section. For a more extensive description of OSGi, see Følsgaard and Ludwigs [9].

The Open Services Gateway Initiative (OSGi) created by the OSGi Alliance is a specification that enable the modular assembly of software built with Java technology [14]. Despite OSGi being a specification and not an implementation, the term OSGi will be used to denote an implementation, when the specific implementation is irrelevant.

2.6.1 Classloader

Java programs consists of classes. These classes can be stored individually in the file-system, allowing the JVM to use the class-path, package-location and class name to locate the class in the file system. More often, however, classes are packaged into jar-files, containing multiple class-files plus other resources, that java programs can access through the class-loader. Large programs will usually consist of several projects that go together to form the finished program. A program created in java, that needs to read and parse an input-file, perform statistical calculations on the input, plot the result on screen and log various messages to disk could implement all of these features itself, or it could rely on some of the existing implementations available for use. Reading and parsing, could be done using ANTLR, statistical calculations using Colt[4], graphs using Java Advanced Imaging (JAI)[5] and finally use log4j to control the log-output. Each of these projects are supplied in separate jar-files containing the class files that make up the project. A java program would then contain all of these jar-files, allowing the program to use the classes contained in those jar-files. Each of the jar-files may have dependencies, but these can be retrieved and added to the program class-path as well. Assuming a stable API, this separation of functionality also allows a program component to be updated without affecting the functionality of the program.

The JVM uses the class-path parameter to locate classes and jar-files containing classes in order to find and load the classes that are instantiated from the running program. The

in order for a program running in a normal jvm to load and unload classes, the program itself must manipulate the classloaders, which is normally handled exclusively by the jvm. Without manipulating the classloaders, the replacement of classes at runtime impossible.

The way OSGi can handle the runtime addition and removal of bundles is through the manipulation of the classloaders used to load classes. Using OSGi to handle the dynamic loading and unloading of classes means that individual programs do not need to implement the classloader manipulation functionality themselves.

2.6.2 Manifest

A bundle is simply a jar-file with extra attributes in the manifest used by the OSGi-framework to identify the OSGi properties of the bundle.

A typical jar-manifest would specify the Main-Class and optionally the Class-Path plus meta-information such as the creator of the jar, when it was created etc. The OSGi specific entries are information such as the packages that the bundle imports (i.e. its dependencies) the packages the bundle exports (i.e. the packages that other bundles can import) and the private

[4] http://acs.lbl.gov/software/colt/index.html
[5] http://java.sun.com/javase/technologies/desktop/media/jai/

packages used by the bundle which should neither be exported to other bundles, nor imported from other bundles.

A bundle can be started and stopped by the OSGi-framework, which would add or remove the classes of the bundle from the classpath used by active bundles. In order for a jar-file to be a valid OSGi-bundle, a number of properties must be specified by the bundle-manifest. The *Bundle-Activator* property specifies a class implementing the *BundleActivator* interface. The interface specifies *start* and *stop* methods that are invoked when the bundle is, respectively, started and stopped. A bundle manifest would typically also contain the *Import-Packge*, *Export-Package* and Private-Package properties. The *Import-Package* property specifies all packages that the bundle imports, in order for the bundle to resolve its dependencies on other classes. The *Export-Package* property specifies the packages that the bundle exports to other packages. Exported packages usually contains API classes that provide access to the functionality of the bundle. The *Private-Package* property specifies the packages that the bundle uses internally and should not be replaced with classes from other bundles.

Listing 1: Example Bundle Manifest

```
1  Manifest-Version: 1
2  Bnd-LastModified: 1324991661475
3  Build-Jdk: 1.6.0_26
4  Built-By: kasper
5  Bundle-Activator: dk.lindbergonline.wiki.api.WikiApiActivator
6  Bundle-Copyright: Kasper Lindberg
7  Bundle-Description: Wiki API OSGi Bundle
8  Bundle-ManifestVersion: 2
9  Bundle-Name: Secure Wiki System Interfaces
10 Bundle-SymbolicName: dk.lindbergonline.wiki.api
11 Bundle-Vendor: Kasper Lindberg
12 Bundle-Version: 0.0.3.SNAPSHOT
13 Created-By: Apache Maven Bundle Plugin
14 Export-Package: dk.lindbergonline.wiki.api.action;version="0.0.3.SNAPSHO
15  T",dk.lindbergonline.wiki.api.authentication;version="0.0.3.SNAPSHOT",d
16  k.lindbergonline.wiki.api.data;uses:="dk.lindbergonline.wiki.api.model,
17  dk.lindbergonline.wiki.api.authentication";version="0.0.3.SNAPSHOT",dk.
18  lindbergonline.wiki.api.plugin;uses:="dk.lindbergonline.wiki.api.model"
19  ;version="0.0.3.SNAPSHOT",dk.lindbergonline.wiki.api.model;uses:="javax
20  .servlet.jsp,javax.servlet.http";version="0.0.3.SNAPSHOT",dk.lindbergon
21  line.wiki.api.tracker;uses:="dk.lindbergonline.wiki.api.data,org.osgi.u
22  til.tracker,org.osgi.framework,org.osgi.service.log,dk.lindbergonline.w
23  iki.api.plugin";version="0.0.3.SNAPSHOT"
24 Import-Package: javax.servlet,javax.servlet.http,javax.servlet.jsp,org.o
25  sgi.framework;version="[1.5,2)",org.osgi.service.http;version="[1.2,2)"
26  ,org.osgi.service.log;version="[1.3,2)",org.osgi.util.tracker;version="
27  [1.4,2)"
28 Tool: Bnd-1.43.0
```

As shown in the example bundle manifest in Listing 1, the *Export-Package* property specifies both the package and the package version of the the packages being exported. Similarly, the *Import-Pacakge* property specifies the version-range of the packages to import. The format used interprets [and] as inclusive and (and) as exclusive. This allows OSGi, in contrast to the JVM class-path, to handle multiple versions of the same bundle. e.g. A bundle that requires antlr-runtime version 3 can execute using version 3, while another bundle using Antlr-runtime version 2 can execute using classes from the version 2 bundle. This is assuming that

the two bundles using antlr-runtime will never interact in a way that could mix the two version. The Uses-clause specified in the *Export-Package* property is intended to prevent such conflicts at runtime by moving the conflict forward to the dependency-resolution stage of the bundle activation. Effectively this fail-fast approach will cause the OSGi-framework to complain and fail to start a bundle, for which such a conflict could occur.

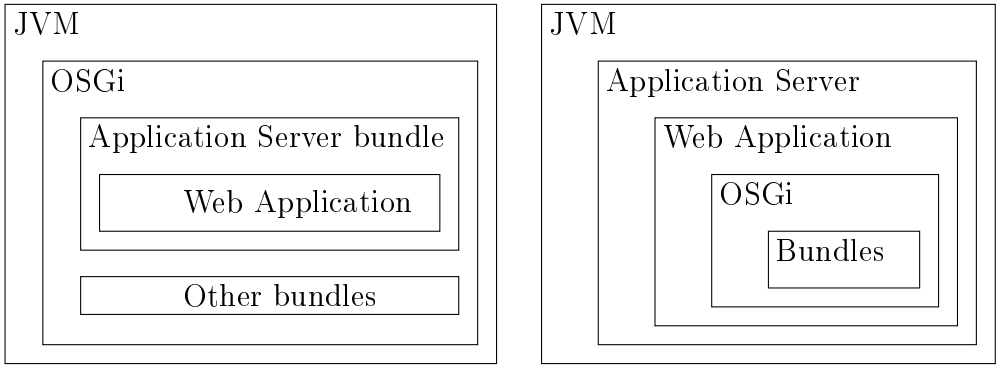
Creating a program based on bundles is similar to creating a program based on regular jar-files. Using jar-files, the JVM will have a class-path and the program will be able to access all classes on the class-path. Using OSGi, the OSGi-framework will start all bundles, making their exported classes available to other bundles importing those classes. The difference is that when using OSGi, bundles can be added, removed and replaced at runtime, which will affect the functionality available to the running program. This allows a bundle to be updated if the bundle is found to contain a bug without restarting the program.

2.6.3 Services

A feature of OSGi that this project will make use of is the concept of services. Using OSGi, a bundle can define a service, that can then be retrieved by other parts of the system. Services are registered as an implementation of a class or an interface, allowing one bundle to define a service-interface, a second bundle to retrieve and use that service to have an action performed, while a third bundle provides the implementation of the service.

2.6.4 Application Servers

Normally, a java based application server, such as the Apache Tomcat, has a fixed class-path set at startup. Integrating OSGi and application servers can be done using different methods. Figure 1 illustrates two methods to do this. Embedding an application server in an OSGi-environment, as shown in Figure 1(a), requires the application server to be contained in one or more bundles, which could allow the modification of the application server's class-path. This application server would then contain a minimalistic web-application that could be as simple as calling modules defined in bundles added to the OSGi-environment outside of the application server. Since both application server and the OSGi-framework is pure java, OSGi can also



(a) Application Server in OSGi (b) OSGi in application server

Figure 1: Two ways to combine an OSGi-environment and an application server

be embedded into the web application container, as illustrated in Figure 1(b). This works by

creating a bridging application that bridges the gap between the web-application container and the OSGi-environment and manages the activation of bundles in the same manner as non-web-based applications.

3 Analysis

A wiki is a knowledge sharing system managed by a user-community. The best known example of a wiki is Wikipedia, which contains more than 3.5 million articles in the English edition alone [20]. The ease at which users can add content to a wiki means that it has become a popular tool for communities to collect information. At sites such as wikia.com, anyone can create a new wiki, on a subject of their own choosing, from television shows[6] to bacon[7]. In addition, a number of implementations of the wiki-software exists, e.g. doku-wiki, media-wiki, jam-wiki and more, allowing anyone to set up their own wiki when needed. Having the ability to set up a private wiki is useful for organizations and companies that may prefer to have the system running behind a firewall and limit the access to the wiki to the members/employees through logins, perhaps using their own authentication-infrastructure.

Users can be divided into to distinct categories. Users that limit their actions to searching, reading and following links will be referred to as *readers*. Readers do not contribute to the expansion and maintenance of the wiki. In contrast, users that actively participate in the wiki-community by creating and editing articles and/or maintain the wiki by deleting bad content are referred to as *contributors*.

The reasons for contributing may vary from individual to individual. Some may be readers who upon having read an article, decides to contribute to the article by fixing errors or adding content that they believe is missing from the article. Having contributed, they revert back to readers. Others are regular contributors that regularly contributes by actively looking for and fixing errors, creating new articles, expanding existing articles and/or checking contributions for vandalism.

Most wikis allow users to create user-accounts in order for the users to be identified regardless of which IP the user connects to the wiki from but does not prevent anonymous contributors from contributing. this means that contributors can be anonymous contributors or they can register an account for the wiki system such that their edits can be traced back to them.

By registering an account, the contributor gets a pseudo-identity that becomes his identity in relation to other users on the wiki. Unless the user directly reveals his true real-world identity, the user is effectively anonymous to other users but still accountable for his actions, since those actions will reflect on the pseudo-identity used. As is the case in the real world, gaining a good reputation for a pseudo-identity requires an effort and this reputation can be damaged if the user vandalizes the wiki. To avoid the consequences of bad reputation, a user can abandon the account and create a new account with no bad reputation. Changing accounts to get rid of bad reputation is known as a whitewashing attack [8]. Doing so can be significantly easier than trying to restore the reputation by doing good. For contributors with an account with a good reputation, changing the account and thereby pseudo-identity will be costly. To restore the new account to the same level of reputation as the last account, the effort required would be equal to the effort put into the previous account.

[6] http://stargate.wikia.com
[7] http://bacon.wikia.com

Usually, anonymous users will be identified by their IP address. For anonymous contributors, gaining good reputation is difficult since contributions are linked to the IP from where the contribution was made. IP-addresses may be shared when contributors are behind a NAT-router and a given contributor may contribute from different IPs for various reasons. The same argument can be made for gaining bad reputation, but when an IP-address repeatedly vandalizes a wiki, administrators can hold the IP-address accountable and block it from making further changes to articles. Given the structure of IP-address allocations to various ISPs, a contributor will often be assigned IP-addresses belonging to a narrow range of addresses. Changing IP-address (if possible) will therefore only be a temporary solution for the contributor as wikis often have the ability to block entire IP-ranges. Should any legitimate contributor be affected by such a range-ban, the contributor can register an account and have the account exempted from the IP-ban.

3.1 Functional Analysis

The following will describe a functional analysis of a generic wiki system. This analysis identifies the components of the generic wiki and enables the breakdown the system in later stages of the project.

3.1.1 Articles

Wiki systems consists of a set of articles, on which a number of operations can be made. Figure 2 shows the structure of an article in a generic wiki. The primary function of an article is to display the current text to the reader. Displaying an article requires the wiki to render the article-markup used to format an article into html used by web-browsers to produce the intended formating of the content. In addition to rendering the article for reading, a number of additional functions are available to contributors.

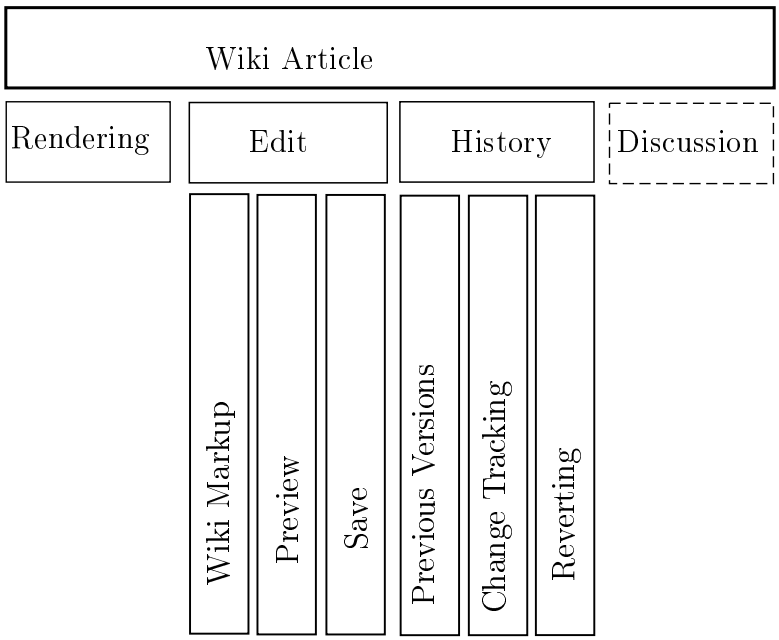


Figure 2: Structure of a generic wiki article. Discussion is not present in all implementations, hence the dashed line.

Articles are written in an implementation specific markup-language that enables contributors to define section-headlines, bold text, italic text and other formating. The Edit-function available from the page of the rendered articles allows a contributor to edit the markup of the article. Using the edit-function, a contributor can do anything from fixing minor errors to adding entire new sections to an article. A few wikis have a WYSIWYG[8]-style editor that functions as modern text-processing programs. The WYSIWYG editor allows the contributor to see the result of the changes while editing. The rest of the wikis uses a preview function that renders the modified markup and displays the result to the contributor. The contributor can then continue editing the markup until the result is satisfactory and eventually save the result and thereby creating a new revision of the article. The preview function is particularly useful if the contributor is experimenting with formating and layout since it removes the need for creating new revisions for each experiment.

Article pages also provide access to the history of the article. The history is a list of previous revisions of the article plus information about the edit itself. This information includes the user who made the edit, the time and date the edit were made and the edit-summaries written by the user who made the revision. From the content of revisions, the changes made between any two revisions can be calculated and displayed, allowing anyone to determine if the changes introduced between the two revisions were good or bad. The final part of the article history is the ability to revert changes introduced by any revision.

Although not all wiki-implementations have them, discussion pages allows contributors to discuss changes before adding them to the main article. Discussion-pages can therefore be an important part of the community-based writing process. As an example of their use, consider an article that describes an actor and the article regularly changes the actors nationality between two (or more) nationalities, a discussion between contributors can be made on the discussion page and the result (consensus) of the discussion used in the main article. After the discussion, an administrator can enforce the consensus and punish contributors that insists on changing the text against the consensus as that can be considered vandalism.

3.1.2 Generic Wiki

The typical user can enter a wiki through links from other pages or through bookmarks in the user's browser. In public wikis, the user is most likely to enter the wiki as a side-effect to doing a search in an external search engine. In a private wiki, the entrypoint is usually the main page, from where the user will use the wikis own search function. Figure 3 shows the functional model of a generic wiki.

Users that enters through the main page will usually start a search to find articles that describe the subject of which the user needs information. When a user has navigated to an article, using internal or external search engines, the user can perform a new search or follow internal links on the wiki, that will lead the user to other articles describing a different subject.

As described previously, contributors have the ability to edit articles and revert revisions if and when they feel that it is necessary. unrelated to specific articles, wikis can use the revision history of all articles to generate a list of recent edits, and most do so. The list of recent edits, provides contributors with a simple method of monitoring changes to the wiki. Some contributors contribute to the wiki by inspecting these changes and checking them for vandalism and other types of bad edits, e.g. addition of unsourced content. Using the article

[8]What You See Is What You Get

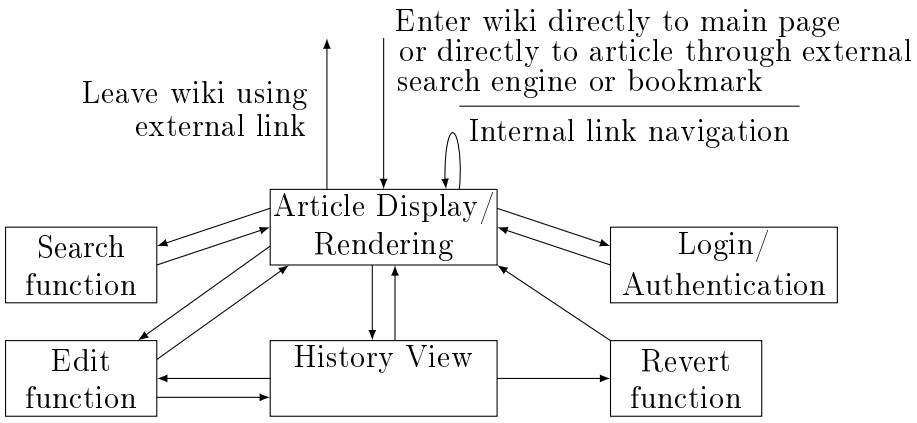


Figure 3: Functional model of a generic wiki

history and revison-diff function, these contributors can easily identify the change that were made and decide if the edit were a good or bad edit. In case of a bad edit, the contributor have the option of using the edit-function to fix the problem or use the revert function to restore the article to the state of a previous, good, revision.

3.2 Generic Wiki Architecture

Using a web-browser, a user can interact with a wiki. The wiki front-end that is sent to the browser as html and displayed to the user provides the means for navigating the wiki.

After navigating to a wiki, a user may authenticate to log in as a previously created user, giving the user a pseudo-identity that will identify the user's actions on the wiki. If the user does not authenticate, his identity will be dependent on information that can be extracted from the connection and will be subject to change if the user changes location and can be shared by others if their connection-information equals the users connection-information[9].

Most wikis do not directly provide a table of contents, with which user can locate specific articles. For small wikis, a table of content can be created manually or automatically (assuming a method of grouping/ordering exists), but for large wikis, such as Wikipedia, with a huge number of pages and a large number of categories, which provides little to no real grouping, a table of content would be extremely large and probably somewhat incomprehensible and therefore unusable. The main method of finding information in wikis is therefore the search function, usually available from any page in the wiki. The ease at which information can be found is then directly tied to the quality of the search tool. For publicly available wikis, external search engines can be used, but for private wikis in organizations the wiki's users only have the wiki's own search tool.

Figure 4 shows the architectural model of a generic wiki. The backend web server, which provides the content to the client web browser, is one of the major components of a wiki. Another is the data storage from where the backend server retrieves the content to display. Around this, is components that handles the articles, the edit-functionality, the revert functionality, the search functionality and the authentication functionality. When working together, these components enable users to find and view articles as well as create and edit articles. The following describes these components in more detail.

[9]This can happen when multiple users on a NAT-network share the same external IP

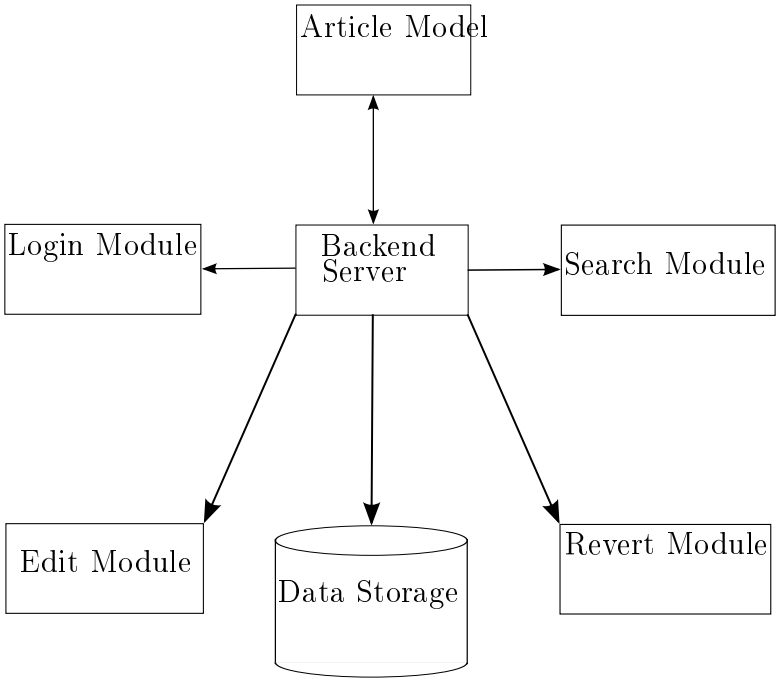


Figure 4: Architectural model of a wiki

3.2.1 Backend Server

A part of the success that wikis have become, is the fact that wikis can be accessed using nothing more than a simple browser, something which is provided by all modern desktop OS environments. It is uncertain if Wikipedia would have grown to the size it is now, with millions of articles, if access to the site required the users to install a client-program for the specific purpose of accessing Wikipedia. Web browsers are the client part of the client-server setup of any website with web-servers being the server-part. Web-server is a type of program, for which several implementations exist, all with different capabilities. The open-source Apache-webserver is a popular choice for simple static webpages but can easily be extended to run server side scripting such as PHP, on which many wikis, including MediaWiki, the software that is used by Wikipedia, are based. For Wikis that are written in Java (e.g. JamWiki, xwiki), the basic Apache HTTP server is insufficient since it is lacking the ability to run Web applications. For this, Apache has created the web application server Tomcat, but other servers such as Jetty and IBM Websphere Application Server are equally capable of running web applications and therefore wikis written in java.

Small wikis have small backend requirements while big wikis that serves a high number of requests have big requirements to their backend servers. Small wikis dedicated to narrow topics with a low number of visitors can run on a single machine with standard or low-end hardware. Larger sites, such as Wikipedia and wikia.com have a significantly larger number of visitors and needs to run on high-end hardware while taking advantage of clustering and load-balancing, both wrt. web-servers but also wrt. the databases that store data.

3.2.2 Editor & Rendering Engine

The article rendering and editor components of a wiki systems are closely related in the sense that editors usually assists the contributors in formating the article. Editors can range from very advanced WYSIWYG-editors that works just as modern text-processing programs to a simple text-area where the contributor can enter unformatted plain-text. Most wiki-systems use a combination, where the input is plain-text, but the editor provides the features of modern text-processors. The combination results in a plain-text input containing character sequences/-markup indicating the formating to use for the following and/or enclosed text.

Articles are usually written in a simple implementation-specific wiki-markup language that the rendering engine can convert into html that will be sent to the client and displayed in the browser.

The purpose of the wiki-markup language is to simplify the creation of articles, while allowing the contributors to format text as they normally would. A well written text will as a minimum contain sections and paragraphs, where each section is started by a heading. Html defines six headings in different sizes and wiki-markup tends to match this number. Editors usually provide buttons and/or shortcuts to insert the start and end character sequences for creating such headlines as well as bold and italic text, ordered and unordered lists and implementation specific formating features of the wiki. Wiki markup usually provides the same core set of formatting options, but different implementations can use different and directly incompatible markup syntax.

3.2.3 Backend Database

The backend server's database can be managed using anything from large-scale enterprise database systems distributed across multiple servers to database systems on the same machine as the webserver or it may even be managed using plain textfiles stored on disk. The specific approach used by the wiki depends on the size of the wiki and the number of visitors the wiki has. Regardless of which type of datastorage the wiki uses, the database needs to store information on user-accounts, existing pages, the text of pages and the revisions that are created during the lifetime of the wiki.

If the wiki is to be optimized to use only the minimum amount of storage required, each new revision could be stored as a diff, relative to the previous version of the article. This approach could dramatically reduce the storage required, when the wiki contains long articles, to which only small changes are made in each revision. The downside of this approach is that in order to display an article, the system must sequentially calculate each revision using the diffs. This will affect the loading speed of articles and will worsen for each new revision. To speed things up, the full text of the current version of an article could be cached, but for comparing revisions, the overhead still exists. Luckily, storage is cheap, so storing the full text for each new revision is not a problem. Besides storing the text for each revision, the database should contain information on which pages have been created and be able to link this information the information on revisions. The revision data would normally contain information on the user who created the revision, the date and time on which the revision was created and the edit-summary given by the user.

3.2.4 Reverting Revisions

Despite people's best intentions, it will happen that one person writes something that another person finds to be factually incorrect and/or unsourced, i.e. unverifiable. For this purpose contributors have the option of reverting an edit to the state of the previous revision. Being able to revert edits is also helpful in case of vandalism, which, due to the open nature of wikis, they are vulnerable to.

When reverting a revision, basic wikis will copy the previous revision and make a new revision using the copied content. This has the unfortunate consequence of reverting all newer revisions too. Since most vandalism is reverted quickly, this is not necessarily a big problem. Good revisions made after a vandalism-revision will most likely have removed or reverted the vandalism manually, removing the need for reverting the vandalized revision. More advanced wikis have the ability to revert any edit without affecting any newer revisions. This is done by reading the revision to revert and the revision that should be reverted to, from the database and calculating the changes that were introduced by the edit. These changes will then be reverted in the newest revision and the result presented to the contributor, who initiated the revert, in the editor. This allows the editor to fix any problems before completing the reverting by saving the new revision.

3.3 Secure Wiki Model Plugin

When adding functionality to a base implementation of a system through the use of plugins, two approaches can be taken. Depending on the type of functionality, the plugins may be informational only or they may be capable of affecting the normal operational flow. In the first case, such a plugin could be a Page-Ranking plugin to a browser that can look at the address of webpage displayed in a browser and present information about the page based on information from a third-party server. The second case could be a Phishing-filter plugin[10] that would check addresses against a database of known malware sites before the browser retrieves the page. If the phishing filter determines that the address is dangerous to the user, the plugin needs the capability to stop the browser from connecting to the address. Technically, the plugins are similiar in that they both check a URL against a database, the difference being how much control over the base system they need. The Page-Ranking plugin needs only the ability to display information which it can do by initiating new actions in the browser whereas the phishing filter plugin needs to be able to terminate a running procedure being performed by the browser.

In terms of supporting the two types of plugins in the base system, type 1, exemplified by the page-ranking plugin, needs only an API to call and the ability to be notified when certain events occurs. In contrast, type 2, exemplified by the phishing filter plugin, needs to be called as part of a procedure, and the procedure needs to be designed such that the type 2 plugin can prevent the procedure from continuing.

[10] Phishing filters are a standard part of many modern webbrowsers, but a plugin using different databases implementing a whitelist-approach in stead of the normal blacklist approach may be relevant in a high-security corporate environment

3.3.1 Overview

The secure wiki model described in Section 2.4 is a combination of both plugin types. Figure 5 shows the architecture of the secure wiki model plugin on top of the generic wiki architecture from figure 4. In order for the secure wiki model to guarantee the consistency of the database, the secure wiki model plugin must be able to control all operations that can modify the database. As shown in Figure 5, this requires that the secure wiki model plugin is consulted before editing, reverting and storing articles in order for it to be able to stop the action from completing, when necessary. In addition, the secure wiki model adds promotion and demotion actions to the wiki system. These actions are responsible for changing the QCV of contributors. Finally, the secure wiki model plugin also needs to be able to add the integrity level of an article to the article before it is sent to the user.

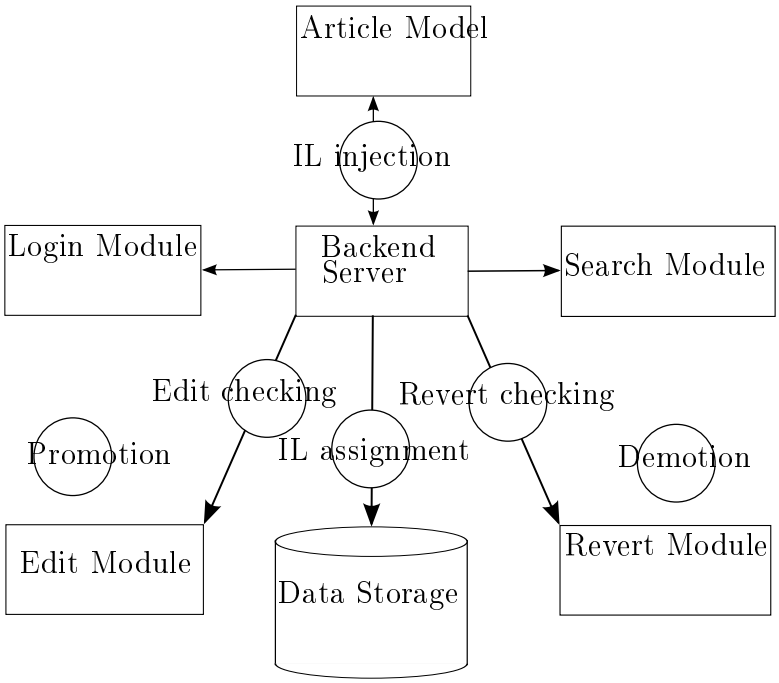


Figure 5: Architectural model of a generic wiki with the secure wiki model

3.3.2 View Model

The aim of the secure wiki model is to secure the correctness, completeness and lack of bias in articles and to report this measure to the readers. Without a means to communicate an articles rating to the reader, the secure wiki model would fail its purpose. It is therefore crucial, that the secure wiki model plugin can inject a representation of the article IL into the output sent to the reader. The representation of the integrity level can be numbers, but can also be phrases such as good, bad, neutral, excellent etc. Using phrases instead of numbers might be easier to relate to and therefore more user friendly, giving a better user experience.

3.3.3 Edit Model

In terms of functionality, there are two steps needed to edit an article. At first, the article markup must be retrieved for editing and then the modified markup must be submitted to the

system to be stored. Any attacker determined to change the markup of the article can bypass the first step and submit a new markup directly to the system. For this reason, any security features put in the first step will be of little consequence to attackers. Any legitimate user would on the other hand benefit from a system with a fail-fast approach, so that the users will not spend time writing articles that cannot be stored.

Users will usually access the edit-page from the article-page. The secure wiki model could therefore be given the ability to either disable the edit-link or display a warning that the user is unable to store the modified markup. If the user, despite the efforts made to disable the edit-feature, manages to access the edit-page, the same warning should be displayed. The user can chose to ignore the warning, but the checks made when performing the actual storing of the new revision would stop the user from making an unauthorized modification to the article. The condition for editing specified by the secure wiki model is the *can_ edit* predicate defined by (1) and, in a simplified form, shown in (4).

$$il(d) \leq qcv(a) \tag{4}$$

3.3.4 Revert Model

The secure wiki model does not directly specify the criteria for reverting revisions. Reverting a revision effectively alters the article, which suggests that the criterias for editing should also apply to reverting. When reverting an article, the revision that is being reverted to can have an IL that is lower, equal to or higher than the current revision. It could be argued that since the *can_edit* predicate defined by the secure wiki model is based on the assumption that any edit will not reduce the quality of an article, reverting to a revision with a lower IL should not be allowed. However, it must be assumed that an attacker has the markup of the previous revision, and therefore could simply edit the document to achieve the same effect. The condition for revert is therefore reduced to the condition for editing shown in (4).

3.3.5 Storage Model

Both the edit-feature and the revert-feature ultimately calls the storage module to store a new revision. The edit and revert feature should perform their respective checks to ensure that no unauthorized modifications are made. The storing of a new revision requires that the revision is given an IL. As the first prototype concluded, the automatic promotion of articles to the QCV of the contributor, as suggested by the secure wiki model, does not take into account the times when contributors (with a higher QCV) makes small changes that does not significantly increase the IL of the article. To solve this problem, the automatic promotion could be removed from the secure wiki model and then let the community promote revisions. The wiki community, however, cannot be expected to review and promote every new revision that is made of every article. The compromise used in the first prototype is to let the author of a revision decide the IL of the revision with the limitation that it must be between the IL of the previous revision and the QCV of the author. Formally, this is expressed by (5).

$$il(d_{old}) \leq il(d_{new}) \leq qcv(a) \tag{5}$$

The need for demotion, however, removes the lower bound. This is argued in Section 3.3.7 and formalized by (6).

3.3.6 Promotion Model

Promotion of articles and contributors has no corresponding action in a traditional wiki. Based on the decisions to give authors the ability to and responsibility for promoting articles with an IL lower than their QCV, promotion reviews of articles is not relevant in this model. This leaves promotion of contributors to be handled by the secure wiki model. Contributors who are selected to perform a promotion review of another contributor are referred to as reviewers. Reviewers should have a means of navigating to the page containing the necessary information about the review, including a means for the reviewer to indicate his decision on the review. The review-page itself needs to contain information relating to the contributor being reviewed. The relevant information about a review would be the contributor's contributions to the wiki dating back a limited time, no longer than when the contributor were last promoted. The reason for not including all contributions, is that contributions before the last promotion should not be relevant to the promotion of the contributor to a higher QCV. The time-constraint limits the number of contributions to be looked at by reviewers, while preventing early bad contributions from affecting the promotion review of a contributor who have improved in quality. In contrast, the same time-limit would prevent early good contributions from affecting the promotion review of a contributor whose writing quality have worsened. For a stable contributor, the time-limit means nothing as the included contributions are representative of his writing. Despite this, the full list of contributions made by any contributor can be seen and therefore reviewers can, if they really wish, use the entire contribution-history against the contributor, for good or bad.

An attacker that controls a (large) number of malicious colluding users has a certain probability of controlling a review. This means that if the attacker repeatedly attempts to be promoted immediately following a failed promotion review, the attacker should in theory, by virtue of the random selection of reviewers, get a selection of reviewers which is controlled by the attacker. To slow down this kind of attacks, a cooling-off period can be enforced, such that an attacker cannot start a new promotion review immediately following a failed review. This is also beneficial wrt. normal users, who has nothing to gain from starting a new promotion review, as the outcome is most likely the same. Enforcing a cooling-off period means that reviewers are saved the trouble of re-reviewing a contributor and ultimately re-cast a negative vote.

Storing modified QCVs is directly handled by the secure wiki model plugin since ILs and QCVs are not a part of normal wiki-systems.

3.3.7 Demotion Model

Demotion, like promotion, does not have a corresponding action in traditional wikis. In the first prototype, demotion of articles are left to the administrators of the system. However, since this takes demotion out of the control of the community, and a discussion among contributors takes time, this makes promotion fast and demotion slow. In the secure wiki model all users who can edit articles, i.e. for those who the *can_edit* predicate is true, can request a demotion review. Since individual contributors have been given the ability to promote documents without a review, so should individual contributors be given the ability to demote documents without a review.

This decision means that demotion reviews of articles are not part of the model while demotion of contributors should still be handled by the secure wiki model. Reviewers should have the same information available as when making a promotion review.

In contrast to the promotion reviews, demotions reviews should not have a cooling-off period. Any malicious user that survives a demotion review should not be protected by a cooling-off period. If a user is repeatedly made the subject of a demotion-review, without this review succeeding, the user who initiates the demotion-reviews can be demoted due to his actions. For this to be effective, a demotion review should only be started by users with a QCV equal to or higher than the QCV of the user that is made the subject of a demotion review.

Demotion of a document requires the contributor to indicate an IL lower than the document's current IL. With this consideration, (5) must be modified to allow lower ILs. When creating a new revision, contributors should be able to select ILs freely up to their own QCV. This is expressed by (6).

$$0 \leq il(d_{new}) \leq qcv(a) \tag{6}$$

3.3.8 Voting

Wiki systems normally do not include the functionality needed to manage secret voting. Voting in Wikipedia is conducted through editing a talk-page and writing signed statements of support of opposition. Voting in other wiki-systems would presumably be conducted in a similar manner. As evidenced by the e-bay rating system, where both buyers and sellers are rated, and where a bad rating from one party can be retributed with a bad rating from the second party, a public vote could become problematic as no-one would be interested in giving a negative review. To avoid contributors using the voting system as retribution, voting needs to be secret[11]. The secure wiki model needs to manage the voting data independently of the wiki system, as voting is not part of normal wiki-systems.

3.4 Secure Wiki Model Policy

The general philosophy behind a wiki is that everyone can contribute to everything. This, however, is also the problem that wikis needs to deal with, everyone can contribute with vandalism instead of good content. If readers of wiki systems are to be able to trust the content of the wiki, this central concept needs to be bend a little to allow for the wiki (through a plugin) to prevent users from negatively changing good articles.

The original secure wiki model requires that contributors and articles are associated with an integrity level and that these integrity levels determine if a contributor can edit an article. For any contributor, not at the highest integrity level, there will be a set of articles that the contributor cannot edit while at that level. The important thing to note is that the contributor is not permanently prevented from editing the set of articles, but only until the contributor proves himself to be capable of writing at the same level of quality as the articles. Wikipedia uses a similar approach on their semi-protected articles, where new contributors are prevented from editing until their account is four days old and have made 10 other edits. The two polices are similar in that they prevent some users from editing a set of protected articles, until the users meets a set criteria. The difference being that the criteria for the later is a simple task, where the former can require a bigger effort than the user is willing to invest. The exact effort is based on the community managing the integrity levels and it will therefore be up to them

[11] The secrecy is only required between contributors. The system can know the source of votes, as it does not reveal this information to users of the system, neither directly nor indirectly.

to decide the required quality of each integrity level and by extension the integrity level that matches a user's quality of writing.

3.4.1 Overview

The policy used by the secure wiki model can vary depending on the seize and needs of a specific wiki. The original secure wiki model used community based voting procedures to determine if a document should be promoted or demoted. In a community where each account is given a vote, controlling multiple accounts means controlling multiple votes. This is often referred to as a sybil-attack. If a single attacker controls enough accounts, the attacker will be able to control the outcome of a review. Preventing such compromise is a requirement for a successful promotion policy, while the demotion policy should be able to restore a partially compromised user-base to an honest one. If an attacker manages to completely compromise the user-base, where no demotion policy has any chance of performing a successful demotion, the system must be cleaned by system administrators.

An attacker wishing to compromise a promotion policy must first get users in a position to influence the promotion policy. This requires the attacker to invest the effort required to have a user promoted. If a single honest user can be promoted, a malicious user will also be able to get a single malicious user promoted. Getting the second and third user promoted will also be within the capabilities of an attacker. The requirement for a promotion policy will be to allow normal users to be promoted but prevent an attacker from getting a large number of malicious users promoted, such that they will be able to control a review.

To be promoted, an effort needs to be made, demonstrating the writing skills of the users. The brute-force approach to attacking this system would be to follow the rules and have a number of malicious users promoted on the basis of an honest effort. To reduce the effort required, an attacker can take a number of shortcuts. Copying existing works outside the system, possibly translating the text in the process, can be an easy way to contribute complete, accurate and unbiased content to a system. However, automatic translators usually does not produce well-written translations. Making these translations well-written therefore still requires some effort by the attacker, although less than if the attacker were to create the text from nothing.

If an attacker can acquire high-level accounts in good standing through the use of blackmail, bribes, phishing[12] etc., the attacker can bypass the effort required by the promotion policy. No policy can prevent against account ownership changes made using means outside of the system. The secure wiki model can only react to ownership changes by starting a demotion review to determine if the new owner meets the requirements for the, at that time, current QCV. Maintaining the integrity level of a compromised account is relatively cost-free and allows the attacker to amass many accounts before starting to perform malicious actions with the acquired accounts. Whether accounts are acquired through bribe, blackmail, phishing or other methods, each acquired account represents an effort made by the attacker. In addition, owners of accounts acquired through fraud may inform the administrators of the system who can then take action against the compromised accounts and have them returned to the rightful owners

Lastly, any attacker who can figure out which users are able to cast votes on the promotion vote, can attempt social engineering. How successful this approach will be is difficult to say, but it is plausible that reviewers who are undecided can be convinced to cast a positive vote,

[12]Beatings are unlikely as account-owners are assumed not to be in physical contact.

where they would otherwise have cast a negative vote. This reduces the amount of work needed to be promoted, however, the social engineering requires some work, which may equal or exceed the amount of work otherwise saved.

3.4.2 Promotion Model

The original secure wiki model uses a promotion model, where the decision to promote is made based on the majority decision between three levels, each holding their own individual vote to determine the level-decision. In the first prototype implementation, this procedure was changed to being only a single vote between all users having higher integrity levels and the majority of the vote were the result of the review.

Regardless of the procedure used to promote contributors, the trustworthiness of an article, comes from the trustworthiness of its integrity level, which in turn comes from the trustworthiness of the promotion procedure used to promote authors.

A strong promotion procedure that only promotes the best of the contributors will increase the trustworthiness of articles. However, too strong a promotion procedure could be said to be incompatible with the all-can-edit policy of open wiki systems. In contrast, if the promotion procedure is too lose, too many low-quality contributors would be promoted and the trustworthiness of articles would decrease as a result. The appropriate policy to use for a given wiki system will most likely depend on the type and size of the system. Small wikis will benefit from a loose procedure that will allow the wiki to grow while systems the size of Wikipedia can benefit from a stronger promotion procedure to build reputation and trustworthiness.

Regardless of the wiki and its needs, the promotion policy must be sufficiently secure to prevent attacks by malicious users. Assuming that an attacker cannot bypass the promotion procedure by hacking the system, the primary threat to a voting-based promotion procedure is collusion between users in order to get a specific outcome. Two colluding authors is not a problem, but collusion between a significant number of authors can at best affect the apparent support for a promotion and at worst, control a review.

The policy used in the original model required an attacker to compromise at least two of the three levels participating in a promotion review, one of which being above the level being promoted from. Compromising a level required the attacker to control enough users in the level to have enough colluding malicious users included in the set of selected reviewers of the level.

The promotion policy used in the first prototype implementation changed this to a simple majority vote between all contributors in and above the level being promoted from. As a consequence, votes at the lowest level will count as equal to votes at the higher levels. With all contributors included in reviews relating to levels equal to or lower than their own level and the potentially high number of reviews for contributors to participate in, contributors have little motivation to vote on a specific review and they cannot be expected to vote on all reviews. This means that there will be a limited number of users that participate in a given review. Assuming that the total number of honest contributors that participate in a review is X, the number of colluding malicious authors needed to compromise a promotion review would be $X + 1$. The X honest contributors will be distributed between all participating levels, but since all votes have an equal weight, the $X + 1$ malicious colluding authors can all exist in the lowest level. The security of this policy therefore depends on the value of X being high enough to make it infeasible for an attacker to control $X + 1$ malicious users at the lowest level participating in the review.

A possible attack on this policy would be an attacker that systematically compromises each level with malicious users. An attacker trying to violate an L_0 review can register new users, at little to no cost, until the attacker has the needed number of colluding malicious users. Once the L_0 review has been compromised, the attacker can get users to L_1 at little to no cost. This procedure is then repeated until all levels are compromised. To make this kind of attack harder to perform, the lowest level must be weaker than the higher levels, thus requiring more malicious colluding users at the lowest level. Giving votes from different levels different weights could change the balance of power towards the higher levels. This increases the protection from colluding users, as more users would be needed at the lower levels to account for the increased power given to the higher levels.

Promoting a malicious user from one level to the next means that the malicious user will have the ability to compromise documents at the next level. If the malicious user is promoted from the lowest to the second lowest level, this wrongful promotion is of little consequence, as only low-level documents are at risk of being compromised by the user. Doing the same at the highest levels have greater consequences since compromise of the highest levels can compromise the promotion/demotion procedure beyond repair. The promotion of a malicious user to the highest levels means that the user can influence promotion reviews and assist other malicious users in being promoted to the highest levels. For the low-level case, the user will also have an influence on the promotion/demotion procedure, but when the low-level user demonstrates malicious intent, the user can be demoted. In the high-level case, this is not necessarily the case, if an attacker has enough users to control a demotion review of a user at the highest level.

In order to protect the promotion procedure from being compromised, the ease at which a contributor is promoted must be difficult, while at the same time, it must be easy for a contributor to be promoted, in order to preserve the all-can-edit policy of open wiki systems. Going with the weighted promotion procedure described above, this apparent conflict can be resolved by gradually increasing the requirement for promotion as the need to promote only high-quality contributors increase. In effect, instead of requiring a majority decision, the yes-votes should be above a variable threshold that can be anything from 1% to 100%.

At some point, depending on the weights given to each level, the threshold will be so high, that even if all selected users from the lowest level are colluding, they will not be able to control the promotion review decision. In order to control the promotion review, colluding authors would have to be promoted to higher levels in sufficient numbers to control the review. In order for colluding authors to get to these higher levels, an additional effort would be required by the users promoted to higher levels. In the case of a sybil-attack, where colluding authors are controlled by one small group of people, possibly just one individual, the effort required by the group of attackers would be increased proportionally to the ratio of attackers to accounts.

When selecting a set of users to perform a review, it will happen that some of the selected users will fail to cast their vote. Initially, this is insignificant, if the threshold is evaluated based on the theoretical maximum number of votes. If, however, the number of votes cast is too low, there is a chance that the votes do not represent the opinion of the community. If this happens, the vote should fail on account of unreliability.

3.4.3 Demotion Model

If for some reason a malicious user is detected, or a user starts to make contributions of insufficient quality for the user's current integrity level, a demotion of the user is necessary. The original secure wiki model didn't explicitly specify a policy for demotion, although a policy

similar to the promotion policy is a likely choice. The thesis describing the first prototype implementation is not much help in this respect either, since it doesn't document the specific policy used for demotion.

If the promotion policy is perfect, no demotion of users is needed. However, the promotion policy relies on necessary assumptions, e.g. that users are assumed to continue writing at the quality of their previous contributions, and assumptions are not guarantees. Mistakes happen, honest users can change behavior and attackers can make a high number of high-quality contributions, only to turn and compromise the system.

In the case of mistakes and users who change behavior in a negative way, any demotion policy will work, as long as it accurately determines the consensus of the community. The primary problem that the demotion policy must solve is the demotion of malicious users that have been promoted. Using a weighted policy will allow the demotion of malicious users from a level where the number of malicious users have compromised the integrity of the level. This requires that the threshold used is set sufficiently, such that the colluding users at the lowest level cannot control the demotion procedure.

If the highest level is compromised it may be difficult to perform a demotion review. In this case the community have no option but to throw the towel and call on the administrators of the system to clean up the level in a dictatorial manner. The same applies if any level is compromised to the point where any demotion will be followed by a subsequent and immediate promotion of the demoted user. Given a good management of the promotion policy and a quick detection of malicious users, this should not happen.

4 Design

Many wiki system implementations can easily be found on the Internet. Most of these implementations are provided under a license that would allow non-commercial use and modification of their source code and distribution of the resulting product. The implementations are often made using PHP or Java. PHP is a popular language used by many open-source web-projects, in part due to the ease with which a php-enabled webserver can be set up and the number of people capable of writting PHP means that the number of contributors to a project can easily grow large enough to keep the project going. Similar argument can be made for the Java variant. The problem with PHP, from a commercial point of view is that to distribute the project, the source code itself must be distributed. Commercial wikis written in java can be packaged into web-archives (WAR) and distributed in binary form. Since commercial wikis do not provide source code that can be modified, using a commercial wiki is not an option.

Most popular wiki implementations are large and complex and any extension made to these wikis would be subject to the design-decisions made by the original implementation. JamWiki, as used by Sander and Følsgaard and Ludwigs has no support for plugins due to it's lack of a plugin-architecture. Creating the necessary plugin architecture in JamWiki is a complex task that requires a great amount of work as demonstrated by Følsgaard and Ludwigs. In contrast, The mediawiki software, developed by the Wikimedia foundation as part of their efforts to manage and advance Wikipedia, does have a very effective plugin-architecture that could allow a plugin to abort the storing of a new article revision. The mediawiki plugin-architecture also enables plugins to add a new tab next to the read/edit/history tabs, which could be used to place links to promotion/demotion pages for each article. Despite these obvious benefits, previous experience with the mediawiki implementation have shown that the API is unstable [13] which

would break any extension relying on the old version of the API.

Creating a base wiki implementation, for the purpose of having a stable API and a plugin-architecture that is capable of supporting the actions needed by the secure wiki model, will require an extra effort, but this effort may equal the effort that could be needed if an existing wiki implementation needs to be modified to support the secure wiki model. The argument for using an existing wiki implementation would be that a plugin could be created for this wiki, that could be deployed to any installation. If the source is modified, even by a little, this argument fails and it makes little difference if the whole system is made from scratch.

Ultimately, creating a base wiki implementation is the approach chosen, which will allow continued testing and development of the secure wiki model plugin. When a stable and effective plugin has been identified, the plugin can be rewritten in the prober language, using the target-wiki APIs. Working based on an existing implementation would only decrease the number of wikis, incompatible with the development version, by one.

4.1 Base Wiki System

This base wiki system will be based on Java as the implementation language, partly because of the ease of development when using java, partly because of the chosen plugin framework described in Section 4.1.2.

4.1.1 Execution Environment

An important thing to establish is the execution environment that the system will be running in. Defining the characteristics of this environment allows the system to be set up on any platform that can provide this environment. The environment that will be used is shown in Figure 6 and described in the following. When using java to create a web-based application,

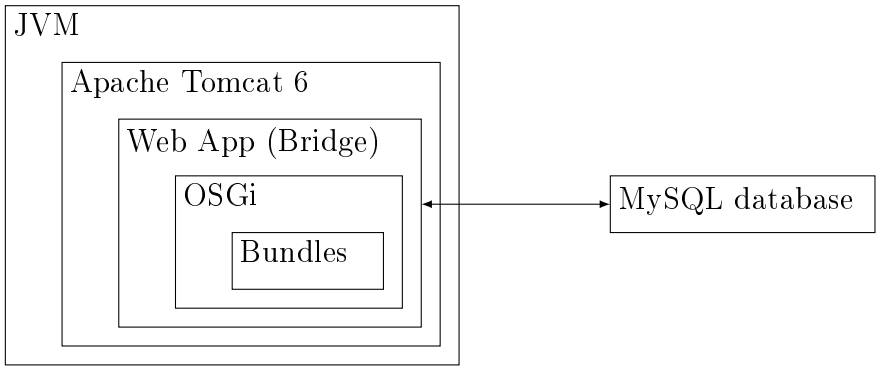


Figure 6: Execution environment

a java application server is needed to run the system. If done right, the choice of application server is irrelevant. Any java web-application should be able to run on any standard java application server.

The Apache Tomcat 6 application server is an easily accessible and popular application server available for most modern operating system. In addition, configuration of Apache Tomcat is well documented and has an active community that can provide support when the written documentation comes short. The lack of documentation and difficulties in downloading the Dynamic Module server (DM-server) chosen by Følsgaard and Ludwigs, combined with the

apparent lack of benefits over the Apache Tomcat means that the DM-server was discarded as a candidate for the task of application server.

The choice of using the Apache Tomcat server for development was made because it only requires a JVM installation and that it functions very well with the chosen plugin-framework. The choice of plugin framework is described in Section 4.1.2 and it should be apparent from this section that the Tomcat server is a good fit with the chosen plugin-framework.

In addition to the Apache Tomcat application server, the base system requires a database. The database chosen to be used is MySQL, due to the availability, stability and popularity of this particular database system. The use of Tomcat and MySQL means that this system can be set up on most major platforms including Windows, Linux and Mac. Since connections to MySQL is made through the network, it makes no difference to the implmentation whether the database system is located on the same physical machine as the application server or on a different machine.

4.1.2 Plugin Framework

In the work Følsgaard and Ludwigs made in converting JamWiki to a plugin-based implementation, they have shown that the *Open Services Gateway Initiative* (OSGi) framework is an effective and powerful framework on which a modularized web-application can be based. OSGi is also the backing framework used by Eclipse, a successful java-based IDE supported by an open-source community as well as large cooperations such as IBM and Oracle, both of which are active contributors [7]. Based on the OSGi-framework specification, a number of implementations of the OSGi-framework exists. Among the most noticeable is the Eclipse Equinox, the Knopflerfish and the Apache Felix implementations.

With the intentions of creating an easy-to-deploy system, the approach where Tomcat will be running in an OSGi-environment (see section 2.6.4) will require modifications to existing tomcat installations or the use of a dedicated server such as the DM-server used by Følsgaard and Ludwigs[13]. The opposite approach, where the OSGi-environment will be running in a Web-container managed by Tomcat will allow for the creation of a WAR file that can be deployed to most applications servers. A number of issues arise when embedding OSGi in an application server. OSGi-frameworks needs to be started, in order for them to work. Usually the OSGi-framework is started and instructed to start everything else. Running in a Tomcat-container the OSGi-framework needs to be embedded in a web-application and started with the application. In addition, most OSGi-frameworks provide a management console, enabling the user to start and stop bundles. This console is accessed using std-in and std-out, which, when running in a Tomcat-container, is not immediately available. For the Equinox framework, a web-application is available for download that will bridge the gap between the Tomcat container and the OSGi-environment. Knopflerfish provides no such bridge, nor were documentation on how to create it found. The Apache Felix implementation does not have a finished OSGi-bridge, but has sufficient documentation and example code to create a customized bridge. In addition, a number of OSGi bundles are available from Apache Felix, one of which, the WebConsole, enables web-based management of an OSGi-framework.

Even though the extra bundles could be used in any OSGi-framework, the ability to easily create a customized OSGi-bridge means that the Apache Felix OSGi-framework is the frame-

[13]From their thesis it is clear that the application-server part of the DM-server is a tomcat running inside the OSGi-framework.

work chosen for the base wiki system. By creating a proof-of-concept, the combination of Apache Felix, Apache Tomcat 6 and available support bundles was tested to ensure that the Apache Felix OSGi could be successfully embedded in Apache Tomcat 6 and managed through the webconsole. The proof-of-concept also had custom-made bundles intended to test the exporting and importing packages as well as the use of services from other bundles. Based on the successful evaluation of the proof-of-concept, Tomcat and Felix were chosen.

4.1.3 System Architecture

In its most basic form, an OSGi-bridge is a web-application whose only function is to start the OSGi-framework and perform the initial configuration of the framework, such that the relevant OSGi bundles are started and their functionality activated. A little more advanced bridge may maintain the framework by starting, restarting and stopping bundles added to, modified in or deleted from a preconfigured watch-directory.

The content of bundles work together to form a working program. Bundles export java-packages for other bundles to use or they can register services, such that other bundles can access and invoke, knowing only the interface provided by the service, not the source of the implementation. The ability of bundles to export both individual classes that can be used directly, assuming the proper java-imports are used, and services, where the caller has no need for importing classes from the bundle that implements the service, means that a bundle containing interfaces can be used to define services, while other separate bundles, implementing the services, can be made by independent third parties.

Conceptually, the whole system can be considered to be packaged in a single bundle. From this all-inclusive bundle, a number of things can be moved to a separate bundle. The first thing to be moved will be all interfaces that defines the API boundary between the different components in the system. Secondly, Storing data can be done using many different approaches. The most common of these is storing data in text-files on disk or using one of the many types of databases. Regardless of which approach is chosen, the system should be able to abstract away from the specific type of storage. Separating the data-access in its own bundle will allow an administrator to use a data storage of his own choosing. Being a java web-application, the Model-View-Control pattern is the recommended way of creating an application. Using the MVC pattern, it is relatively easy to separate the frontend of the system into a separate bundle. This would also allow administrators to use a different frontend-bundle with a different theme for their wiki. What is left in the conceptual all-inclusive bundle will be the core of the system. A number of additional core features, e.g. article-formatting, could be placed in separate bundles, but this is not deemed necessary at the moment. These considerations leads to the architecture shown in Figure 7. The bundles are described further below:

OSGi-bridge The OSGi-bridge is the web application that is installed in the application server. The bridge will be responsible for starting the OSGi-framework and managing bundles that are added to, updated in and removed from a location on disk. The management consists of loading, reloading and unloading the bundles. The bridge itself will not contain any wiki-system related functionality.

API Bundle To enable the system to use different sources of data, depending on what is available, the core bundle will not perform data access directly but through an implementation

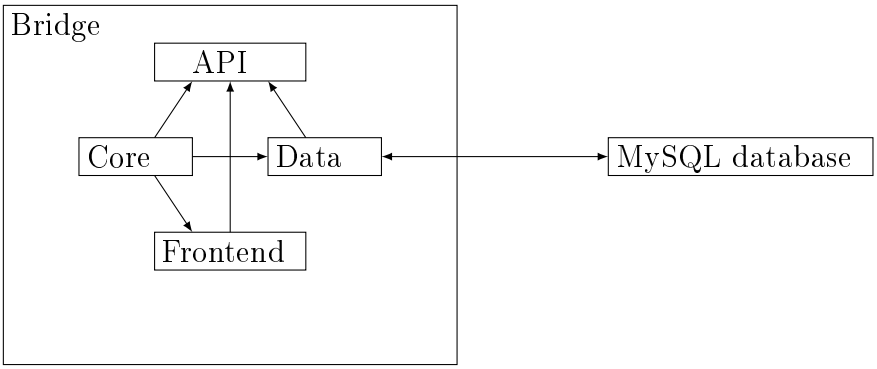


Figure 7: OSGi-bridge bundle structure

defined by another bundle. Providing this implementation to the core bundle will be done using an OSGi-service.

To register and retrieve an OSGi-service, the service must be registered as an instance of a class or interface, known by both the registering and retrieving bundles. For this purpose an API bundle will be used and contain the interfaces that describe how one part of the system can interact with another. The API bundle also contains interfaces describing the model objects used as parameters in these interfaces.

Core Bundle The core system functionality is to be handled by a core bundle that is responsible for the control-flow of the wiki system. The responsibilities of the core will be to receive requests for pages in the system, retrieve data and process it and then forward the result to the frontend. Both the frontend and the data-backend is handled by other bundles. Communication with the data bundle is done using an OSGi-service, referred to as the data service. Communication with the frontend is done through model-objects from which the frontend will get the data to be displayed.

The functionality handled by the core bundle is:

Authentication As user identity is tied to the operation of the base system wiki, the core bundle needs to ensure that the user have been logged in and authenticated in order for the system to allow the user to perform most of the functions in the base system. This is similar to other wikis, where every action requires some form of identity. Most wikis allow anonymous users to be identified by their IP, but the added complexity needed to handle anonymous users in the base system makes this undesirable. Requiring users to have a registered identity is not a conflict with the *all can edit* policy of a wiki, as long as registering an account is unrestricted.

Article Requests The core system will receive requests to view articles and will use the data service provided by the data-bundle to retrieve the markup of the requested article. The core bundle will then format the article to html format and forward the html to the frontend bundle for display.

Editing and Storing Articles Requests to edit articles will be handled by the core in much the same way as requests for articles. The core must retrieve the article markup using the data service and forward this to the frontend. When the frontend returns a new revision,

the core bundle shall use the data service to store the new revision. When editing, the core bundle needs to be aware of the secure wiki model plugin's need to abort a storage request.

Reverting Revisions In order to revert a revision, the revision previous to the revision being reverted must be retrieved using the data service. This revision must then be formated as if it is being edited, and forwarded to the edit-frontend. The user can then modify the result before storing the markup of the previous revision as the new revision.

User Contributions Reviewers participating in an secure wiki model review will benefit from this list to determine the quality of a user's contributions. however, since most wikis provide such a list, the list is a core feature. The core bundle will retrieve the user's contributions using the data service and forward these to the frontend for display.

Article History Identifying the changes made to an article is the core of the classic wiki soft-security scheme. The core will be responsible for retrieving information about article revisions and forward this information to the frontend for display.

Revision Diffs Part of the history feature is to identify the changes made between two revisions. The core will retrieve the two revisions that are to be compared and, using the third party library java-diff, calculate their differences. These differences will then be passed to the frontend along with the original markup.

Keyword Search Searching is highly dependent on the data storage format. The core system will therefore use the data service to search the data for given keywords. When the data service returns matching articles, the core bundle must forward these to the frontend.

Data Bundle Data for the base system will be stored in a MySQL database. The data service implementation will therefore target this database. Using a separate bundle for the data service would allow anyone to switch to use text-files or other data-sources simply by replacing the data-bundle with another implementation.

Frontend Bundle Functionality (backend) and appearance (frontend) of a system does not require a tight coupling. The functional part of the system can perform preprocessing of input and retrieve relevant data from the data-source, pack the processed data into model-objects and pass these to the frontend to be displayed. With a clearly defined API of the model objects as well as a clearly defined API for the interaction with the backend, the frontend can be isolated into a separate bundle, allowing administrators of a running system to change the theme of the wiki simply by changing the front-end bundle.

4.1.4 Plugin API

A requirement by the secure wiki model plugin is to be able to inject content into pages. The base system will support this need by using a viewmodel that will be passed to a presentation-plugin to be manipulated before being sent to the frontend and converted to html to be displayed to the reader. The presentation plugin will only have one method, that will receive a viewmodel

to be manipulated. The viewmodel will have a number of methods, allowing the presentation-plugin to inject content into various locations in the page.

In addition to injection of content into pages, the secure wiki model plugin needs to be notified of certain events. The events that the secure wiki model plugin needs to be notified of is before and after a new revision of an article is stored. Before storing an article, the secure wiki model plugin must be called to perform the *can_edit*-test and return the result. After a revision has been stored, the secure wiki model plugin must be notified to update the IL of the newly created revision. To support this, the base system will have an integrity plugin type and be aware that these plugins must be called before and after the storing of a revision and that the return value of the pre-store function indicates if the storing of the revision should be aborted. Although the plugin-API have been designed with the secure wiki model in mind, the API does not have any specialized functions that a more generic plugin wouldn't have. The call made to the plugin before the storing of a new revision can be considered a pre-store-event and the second call is the post-store-event. Both of these are events that could be relevant to any generic plugin.

4.1.5 Database Design

Storing data can be done using the file system, or a database. The advantage of a database compared to storing data in files is the optimized search-routines that are used when recalling articles and searching for keywords using the search-function to be implemented in the base system. Figure 8 displays the domain model, on which the database design is to be based. Each time an article is stored, a new revision is created. Since an article includes revision history, an article consists of many revisions. The text text being displayed being the latest revision. Each revision is created by a single user.

Figure 8: Base system domain model

Article Requesting an article is done using a url entered in the browsers address bar or by clicking a link to a url. The url will therefore serve as the primary means to retrieve an article. As will be apparent in the discussion of revisions below, the url is not suitable to be the primary key. For the primary key, a dedicated integer-id will be used. The dynamic nature of articles, means that a timestamp, indicating when the article were last modified should also be stored such that it can be displayed with the article. In addition, the markup of the article is needed.

Revision The latest revision made for a given url is the current article being displayed. This means that revisions must include all information that articles requires. Creation of a revision is done by a specific user. This user must be tied to the revision in order to display this user on the article history page. When creating the new revision, the user can give a short description of the changes made or can mark the changes as minor. The storing a revision must store this edit-summary and if the revision should be marked as a minor-edit.

User Users have a unique username and a password that must be checked when the user log in. Since the username is a string, and that it may change, the primary id of a user will be an integer id.

Database Tables These considerations suggests the creation of a table for users and a table for revisions. Articles being selected revisions will be created as a view of those selected revisions. In order to use the mysql text-search function on the article markup and url, The fulltext key is used on the markup and url fields in the revisions table and by extension on the article view.

Users table

uid	Integer	Primary Key, auto_increment
username	Text	
password	Text	
created	Timestamp	Default CURRENT_TIMESTAMP

Revisions table

rid	Integer	Primary Key
markup	Text	Fulltext key
url	Text	Fulltext key
uid	Integer	Foreign key to Users
minor_edit	Integer	
edit_summary	Text	
modified	Timestamp	on update CURRENT_TIMESTAMP

Articles view

aid	Integer	Primary Key
markup	Text	Fulltext key
url	Text	Fulltext key
modified	Timestamp	

4.1.6 Markup Language

Wikis use a special markup language that allows users to input plaintext into the system, which is then transformed into formated text when displayed to readers. The transformation relies on special formating-control sequences contained in the text to identify which parts of the text that is to be formated in bold and italic or made to a headline etc. Large wikis have an extensive syntax for formating text, but the core of the formating features are the same. At minimum, the markup language of a wiki needs to support formating headlines and paragraphs to allow for a structured text. In addition formating text as bold and italic will allow authors to emphasize certain words and phrases in the text. These formating features will be implemented in the base system. In addition, ordered and unordered lists are often used in text and will be implemented as well.

For the purpose of allowing readers to jump from one article to a related article, links between pages are needed. The base system will have support for formating links in text to clickable links.

For ease of maintenance and consistency, some wikis use templates to display common components of an article. This can range from templates that display a single character (e.g. λ, that would otherwise require knowledge of the html-encoding of the character) to info-boxes that displays information in a way consistent across multiple pages (in terms of layout and order

of info-fields). Such templates takes parameters that can be used to customize the information displayed by the template. This is however too advanced to be implemented in the base system.

The syntax used by mediawiki is the same as the one used by jam-wiki. Other wiki implementations have a similar syntax, but with both minor and major differences depending on the implementation. The markup language used in the base system will be a subset of the one used by mediawiki. Specifically, the formating will include

Headlines Mediawiki uses a number of equal-signs on either side of a piece of text to indicate various levels of headlines. This is ranging from `==X==` for a level 2 headline to `======X======` for a level 6 headline.

Bold Mediawiki uses three single-quotes before and after text that is to be formated as bold, e.g. `'''This is bold'''`

Italic Mediawiki uses two single-quotes before and after text that is to be formated as italic, e.g. `''This is italic''`

Bold & Italic To have both bold and italic, the two are combined to give five single-quotes before and after text, e.g. `'''''This is bold and italic'''''`

Links Formating a link is done using two square-brackets, e.g. `[[Link]]`.

Lists For each line in a list, the line must be prefixed with `*` for an unordered list and `#` for an ordered list.

4.1.7 System Management

A number of third-party bundles are available as support for system-functionality, such as logging, and for OSGi-framework management. Most significant is the Webconsole-bundle which allows web-based management of the OSGi-framework. The bridge will load and unload bundles to and from the OSGi-framework, but to activate or deactivate the functionality provided by these bundles, the bundles must be started or stopped. In addition to being able to start and stop bundles, the webconsole can be extended to provide Configuration services, log-output, system information, a command-line shell for direct OSGi-framework management and other more or less useful services.

The log-output and system-information will be useful for debugging problems, the shell is nice to have when troubleshooting bundle-dependency issues, but otherwise replaced by the webconsole functionality. Parameters to the base system will be entered through the webconsole configuration service interface. This gives administrators a webbased input method for parameters, removes the risk of having malformed input files and automates the setting of input parameters in the base system.

4.2 Secure Wiki Model Plugin

Creating a generic plugin architecture would require plugins to be notified for each event that happens in the wiki system. In this context, an event could be post-login, pre-edit, pre-save, post-save, pre-view-article etc. Depending on the plugin-functionality, different events may be relevant.

Implementing a generic plugin architecture, can be done by registering the plugin to listen for each individual event or the plugin can register to listen for all events and only act on the relevant events. The definition of truly generic-plugin API is out of scope for the current project. The base system will therefore use an approach where a plugin registers as a plugin of a given type and the base system will then invoke the plugins where relevant. The system will use a plugin-mediator that allows the base system to retrieve registered plugins. This approach is similar to that of a generic plugin architecture. The difference between this and a generic plugin architecture is limited to the system knowing the type of plugin that is relevant to call at a specific point in the control-flow. To convert it to a generic plugin architecture, the plugin-type must be removed and all plugins called instead. Other than this, the plugin architecture is similar to that of a generic plugin architecture, although the number of events are limited.

As shown in the analysis, the secure wiki model plugin must be called before a new article revision is stored, to determine if the contributor is allowed to store it and when an article has been stored, in order to update the article IL if needed. In addition, the secure wiki model plugin has a need to inject content into the pages displayed to the user. Primarily, the injection of the article IL into article pages, to inform the user of the articles IL, but it also needs to inject an IL-selector into the edit-page, allowing the user to select the appropriate article IL before saving. The result of this selection must then be passed along to the secure wiki model plugin when the plugin is notified that a new revision has been stored.

4.2.1 Architecture

The secure wiki model plugin is constrained by the plugin-API defined by the base system. The base system provides two plugin-types that can be used. A presentation plugin used to modify the presentation of data and an integrity plugin that can veto storage of new revisions and is notified when a new revision have been stored.

by implementing an integrity plugin, the secure wiki model plugin can evaluate the *can_ edit* predicate and decide if the storing of the article should proceed. The subsequent post-storing-event can then be used to assign the proper IL to the new revision. In order for the post-storing-event to assign an IL, the request to store the revision needs to include the corresponding IL in the request. In order for the IL to be included in the request, the form that is submitted must be altered to include this input. Adding the IL parameter to the form will be done using the presentation plugin. The presentation plugin will include the means to place content on the edit-page inside the form, allowing an extra input field to be added.

Using the same presentation plugin, the article display page can be modified to include the article IL. The presentation plugin will also be used to inject menu items into the menu, to provide access to the specific features of the secure wiki model plugin, such as requesting a review and viewing the active reviews in the system.

When requesting a review and when evaluating a finished review, a number of parameters are needed. When starting a review, the number of users that should participate is needed. When evaluating, the weights of each vote and the threshold to use is needed. From the webconsole, these parameters can be entered into the configuration service, which will then update the secure wiki model plugin.

Secure Wiki Model Overview For the secure wiki model to be useful, the secure wiki model plugin must provide an overview of the running reviews and provide the means for users to make an informed decision and indicate their decision to the system. This will be accessible from the navigation-menu.

Receiving a Vote The secure wiki model plugin must provide reviewers with a means to cast a vote. When a vote is received, the secure wiki model plugin must update the review with the reviewer's decision. Reviewers will be provided with links on the overview page to the reviews they can vote on.

Requesting a Review Starting a review is a manual task. From the navigation-menu, users will have access to pages where they can request a promotion and demotion review of themselves or a different user.

Processing a Review When a review ends, defined by the end-timestamp of the review, the votes must be counted and the result evaluated according to the policy specified in Sections 4.3 and 4.4 using the parameters configured by the administrator.

4.2.2 Database Design

Since the secure wiki model plugin is a plugin, the base system database cannot be modified to suit the needs of the plugin, but must be extended to support the needs of the plugin. Figure 9 shows the extended domain model.

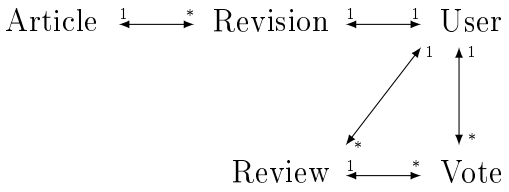


Figure 9: Secure wiki model domain model

Article The secure wiki model plugin needs to associate an IL with articles. Since articles are a subset of revisions, no additional information is needed.

Revision Articles, being a subset of all revisions, must be associated with an IL. For this reason, each revision will be associated with an IL. In stead of storing one IL for the article, an IL will be stored for each revision. This allows the IL to be displayed when viewing previous revisions. Since the *Revisions* table cannot be changed, a new table is needed. The table must contain a reference to the revision and the IL of that revision.

User Like the *Revisions* table, the *Users* table cannot be changed. Again a new table is used, where there is a reference to the *Users* table and the QCV of the user. To support both document and user levels, both a document-QCV and a user-QCV will be stored.

Review Reviews are used to determine the consensus in the community on whether a user should be promoted or demoted. For referencing the review, an integer id is needed. To distinguish between promotion and demotion reviews, a type, indicating if the review is for promotion or demotion, is needed. To identify the user, for which the review is about, a reference to the user must be stored. In addition, reviews must be ended after a certain period of time, so a start and end time is needed. For audit purposes, storing a reference to the user who requested the review as well as the target QCV will be needed. When requesting a promotion or demotion the user can enter a reason for the request. This information should also be stored with the review.

Vote Votes indicate the decision by individual users. Each vote will have a unique integer id, reference the review for which the vote is related and the value of the vote (yes/no/unanswered). In addition, since secret voting is only a requirement between users, a reference to the user who cast the vote can be stored in the table. Doing this means that an enforcement of the (reviewid,userid) pair being unique by the database makes it impossible for one user to vote more than once for any given review.

Database Tables These considerations leads to the creation of 4 additional tables.

Swmp_user table

uid	Integer	Primary Key, Foreign key to Users
doclevel	Integer	
userlevel	Integer	

Swmp_revision table

rid	Integer	Primary Key, Foreign key to Revisions
il	Integer	

Reviews table

rid	Integer	Primary Key, auto_increment
uid	Integer	Foreign key to Users
description	Text	
type	Integer	
targetqcv	Integer	
requestorid	Integer	Foreign key to Users
start	Timestamp	default CURRENT_TIMESTAMP
end	Timestamp	
processed	Integer	Default 0

Votes table

vid	Integer	Primary Key, auto_increment
rid	Integer	Foreign key to Reviews
uid	Integer	Foreign key to Users
vote	Integer	Default NULL

4.3 Promotion Policy

The policy to be used in this project will combine elements from both the original model and the first prototype implementation. From the original model, the idea of selecting a number of authors to perform the review will be used. This has the benefit that in large systems with many users, it is unrealistic that everyone will constantly participate in reviews. Reducing the number of reviewers to a statistically representative number will reduce the voting frequency of the individual users without affecting the final decision of the vote and hopefully increase the willingness to participate. In addition, it will also allow the system to calculate a participation percentage, which can be used to determine if a sufficient percentage of possible votes were cast. In addition, selecting a set of reviewers, prevents an attacker from using all colluding malicious users to influence the review, in the manner described in Section 3.4.2.

Votes cast by users in each level will be given weights and the number of votes multiplied by their respective weights will be the score of the vote. The promotion review success criteria will be based on a threshold of yes-votes that differs for each level. For any vote to succeed, the number of no-votes cannot exceed the number of yes-votes, if the review is to result in promotion. The threshold of a vote is based on the maximum score of a vote, allowing a 30% threshold without the number of no-votes exceeding the number of yes-votes.

For a 30% threshold, only 30% of the reviewers needs to cast a vote. For lower thresholds, even fewer reviewers needs to cast their vote. To ensure that the decisions are valid, a sufficiently large number of people needs to participate in the review. Participation percentage should be checked for, when evaluating a vote, but otherwise does not affect the promotion/demotion policy.

As evidenced by the original model, handling the promotion to the two highest integrity levels becomes tricky, as the three levels required by the normal procedure does not exists. The original model compensates by requiring a unanimous decision. Since the integrity of the highest levels are the most important, reducing the strength of the promotion policy would be undesirable. In order to use the common-case promotion policy for the highest integrity levels, one additional author-only integrity level can be introduced. The original secure wiki model uses five integrity levels. Using level 0 to 4 for documents and authors and adding a level 5 for authors only, would allow the common-case policy to be used for promotion of authors from level 3 to level 4. The author-only integrity level would still write documents at level 4, but stricter policies can be used to protect this level, that may require considerable effort to get to, i.e. not something that everybody can do.

The original secure wiki model [12] used the policy Π_1. The policy defined in this thesis will be referred to as Π_2. Like Π_1, the levels involved in a review, are levels L_i, L_{i+1} and L_{i+2} from where a number of users, defined as r_i, will be randomly selected to perform the review. For each level, Λ_{R_i} is used to denote the set of the selected reviewers.

In the event that a given level does not contain any users, the level above will be used to supply the users needed at the empty level. The two primary causes for empty levels is when all users at a level has been promoted and during the bootstrapping phase of the system, before the levels have been populated. The strategy of selecting users from levels above empty levels has the effect that in order to bootstrap the system, only a single user is needed at the top level. This user should be one of the administrators of the system and can therefore be trusted not to act maliciously. In addition, the user will be included in most reviews during the bootstrapping phase and have a significant amount of control on the result of the review, which again suggest that this user should be an administrator of the system.

Using $\delta_j(\text{a:}\mathbb{A})$ as the decision of reviewer $j \in \Lambda_{R_i}$ on the review of author $a \in \mathbb{A}$ and with each vote weighed according to the level of the reviewer who cast the vote, the level score can be expressed as shown in (7) where $\mathcal{W}_i$ is the weight of the level.

$$\mathcal{S}_i(a) = \sum_{j \in \Lambda_{R_i}} \delta_j(a) \cdot \mathcal{W}_i \tag{7}$$

The maximum score that a level can produce is the number of selected reviewers multiplied with the weight of the level. This is expressed in (8).

$$\mathcal{S}_i^{\max}(a) = |\Lambda_{R_i}| \cdot \mathcal{W}_i \tag{8}$$

The threshold that must be reached is denoted using τ_i. In contrast to Π_1, where the threshold were an integer threshold, which the number of positive votes should exceed, τ_i will, in the context of Π_2 be a percentage, denoting the percentage of the maximum score to exceed. The common case for the promotion policy can then be specified as shown in (9).

$$\begin{aligned} \mathcal{D}(a) = \mathcal{S}_i(a) + \mathcal{S}_{i+1}(a) + \mathcal{S}_{i+2}(a) \geq \\ \left(\mathcal{S}_i^{\max}(a) + \mathcal{S}_{i+1}^{\max}(a) + \mathcal{S}_{i+2}^{\max}(a)\right) \cdot \tau_i \\ \text{for } i \in \{0, 1, \ldots, |\mathbb{I}| - 2\} \end{aligned} \tag{9}$$

For the special case of $i = |\mathbb{I}| - 1$, only two levels can be included in the review, but is otherwise parallel to the common case. This is expressed by (10).

$$\begin{aligned} \mathcal{D}(a) = \mathcal{S}_i(a) + \mathcal{S}_{i+1}(a) \geq \left(\mathcal{S}_i^{\max}(a) + \mathcal{S}_{i+1}^{\max}(a)\right) \cdot \tau_i \\ \text{for } i = |\mathbb{I}| - 1 \end{aligned} \tag{10}$$

If it is deemed necessary additional author-only levels could be used. Since (9) and (10) are defined in relation to $|\mathbb{I}|$, the definition of the common and special case conditions would remain the same.

4.4 Demotion Policy

The demotion policy will be similar to the promotion policy. However, using a different threshold for demotion allows administrators to make demotion easy or hard compared to promotion. Doing this has an effect on the motivation of users to do work. If promotion requires a significant effort, but demotion can happen if a contributor makes a single bad action, it can hurt the motivation of contributors as their efforts may be wasted if they make a simple mistake, but for the same reason, an attacker will have a problem keeping the level of the attacker's colluding 'malicious users. If demotion is more difficult, contributors will be more willing to make an effort as they know that they will not risk being demoted. This, however, also makes it easier to keep malicious users at their repective levels. The amount of negative work to perform to be demoted therefore a balance between ensuring the motivation of contributors and demoting malicious users who tries to blend in the crowd.

The condition for the general case is largely the same as for promotion, except the threshold to use can be set independently. The threshold for demotion is denoted τ_i^{dem}. The condition for demotion is given by (11).

$$\begin{aligned} \mathcal{D}(a) = \mathcal{S}_i(a) + \mathcal{S}_{i+1}(a) + \mathcal{S}_{i+2}(a) \geq \\ \left(\mathcal{S}_i^{\max}(a) + \mathcal{S}_{i+1}^{\max}(a) + \mathcal{S}_{i+2}^{\max}(a)\right) \cdot \tau_i^{\text{dem}} \\ \text{for } i \in \{0, 1, \ldots, |\mathbb{I}| - 2\} \end{aligned} \tag{11}$$

For demotion of a user at level $L_{|\mathbb{I}|-1}$, where only two levels ($L_{|\mathbb{I}|-1}$ and $L_{|\mathbb{I}|}$) can participate in the review, the condition becomes that shown in (12).

$$\mathcal{D}(a) = \mathcal{S}_i(a) + \mathcal{S}_{i+1}(a) \geq \left(\mathcal{S}_i^{\max}(a) + \mathcal{S}_{i+1}^{\max}(a)\right) \cdot \tau_i^{\text{dem}} \quad \text{for } i = |\mathbb{I}| - 1 \tag{12}$$

For demotion of a user at the author-only level, $L_{|\mathbb{I}|}$, where only the same level $L_{|\mathbb{I}|}$ can participate in the review, the condition becomes that shown in (13).

$$\mathcal{D}(a) = \mathcal{S}_i(a) \geq \left(\mathcal{S}_i^{\max}(a)\right) \cdot \tau_i^{\text{dem}} \quad \text{for } i = |\mathbb{I}| \tag{13}$$

If a malicious user is at level L_i, the chances are that more malicious users are at level L_i. The inclusion of level L_i in a demotion review that may demote a user from level L_i to L_{i-1} can therefore potentially include malicious colluding author protecting the user who are subject to a demotion review. It is believed, however, that this will not have a significant effect on the demotion review. Should enough malicious colluding authors exist at level L_i, such that they can control a demotion review, they would also have the power to control a promotion review that could reestablish the QCV of any demoted user. In such a case, not only is the demotion policy violated, but the entire system is in trouble.

5 Implementation

As described by the design section, the secure wiki system have been implemented using Java and targeted to a java web application server. For each of the bundles that have been created, a bundle manifest must also be created. The content of this manifest is, in part, extracted from the existing build configuration and, where necessary, explicitly encoded in the configuration. This allows the manifest to be created on-the-fly during the build process. For the entire project, Apache Maven has been used to control the life-cycle of the build process. This means that everything from dependency management to compilation, packaging and deployment have been controlled using maven. Using maven has the benefit that the project can be compiled from source and have dependencies downloaded automatically as part of the build-process. For this project, however, two third-party bundles have been created from source (see section 5.7) and therefore not generally available for download, these have been deployed to a private maven repository during the development of the project. Another effect of using maven to build the project, is that bundle dependencies, that are not available as a bundle, can be embedded into the finished bundle at build-time, removing the need to convert those dependencies to bundles. This also makes it easier to change the version to depend on.

5.1 OSGi-Bridge

The OSGi-Bridge component is the physical web application to be deployed on the server. The bridge itself is very simple and contains no wiki functionality, but contains the OSGi-framework used to handle the bundles that make up the wiki system. When the application is deployed and the application server initializes the application, the bridge will create and start an instance of the OSGi-framework. The framework used is the Apache Felix OSGi-framework version 4.

The framework and its dependencies are contained as jar-files in the lib-folder of the bridge application.

For the purpose of framework management, the bridge starts a thread that runs concurrent to the framework. The thread monitors a directory on disk, where bundles can be added to and removed from. The thread repeatedly calculates a hash for each of the bundles in the directory to identify changes to bundles. When a bundle is added to the directory, the bundle is installed in the framework. When a bundle is changed on disk, the thread will reload the bundle. When a bundle is deleted from the directory the thread will stop and remove the bundle from the OSGi-framework.

For the purpose of being able to restart the OSGi-framework, the thread will initially consult the file *init.bundles* located in the watch-directory to determine which bundles should also be started after installation. The file contains references to other *.bundles*-files or bundles that should be started.

When building the brigde, all bundles required for the wiki to run is automatically copied to the bundle-watch directory to be started when the bridge starts. The bundles include both the specific wiki-bundles described below but also the web-console bundle and all of its dependencies and extensions. The inclusion of the web-console bundle in the bridge allows manual management of the OSGi-framework.

5.2 API

The API bundle contains all the interfaces that define the boundary between system components. In addition to defining constants for use by all parts of the system, the API bundle defines a number of models and services.

5.2.1 Models

The API bundle defines interfaces describing models used to communicate information between the Data, Core and Frontend bundles. The primary models used by the system is the user model and article models. In addition to this, the history, contribution and diff features each have a model used to transport a collection of data from the data bundle to the core and onto the frontend.

User Model The user model is defined by the interface `WikiUserModel` which defines methods for retrieving the username and the userid. In addition, methods, to get and set properties related to the user object, are defined.

Article Models Articles have a model for a missing article (`WikiNoArticleModel`) containing only the url and title of the missing article. For articles that are to be formated and displayed, there is an article model (`WikiArticleModel`) that provides the formated html to the frontend, along with methods that provide the information needed when editing articles. For use in the history view, a revision model (`WikiRevisionModel`) exists, that provides the id of the revision as well as convenience methods for determining if the revision is a new page or has been marked as a minor edit. In addition, methods for retrieving the edit-summary, contributor name and revision size are defined.

View Model The view model is defined by the interface `WikiViewModel` and is used as a parameter to the presentation plugin. After each presentation plugin have had a chance to modify the view model, it is passed on to the front end, where it is used to construct the content of the pages shown to the user. The view model defines seven points on the page where data can be added.

- **Head:** Used to add javascript and stylesheets to the head of the page.
- **Navigation:** Used to include links to other pages in the navigation. The frontend implementation is free to decide the location of the navigation on the page.
- **Action menu:** Used to include links to page actions in the action menu. The frontend implementation is free to decide the location of the action menu on the page.
- **Before the content:** Used to add content before the start of the page content.
- **Next to the content:** Used to add content that floats next to the page content.
- **After the content:** Used to add content that follows the page content.
- **Form:** Used to add content to the the primary html-form of a page. At the moment, this is only used on the edit pages.

5.2.2 Services

The API bundle defines a number of services, that are used to share functionality between bundles. The data service (`DataService`) and formatting service (`ArticleFormatterService`) are used to perform actions directly, while the lookup service (`LookupService`) and plugin service (`PluginService`) functions as mediators between the different parts of the system.

Data Service The `DataService` interface is used to define an OSGi-service used by the core bundle to access data from the data storage. The interface defines functions to get information needed by the core bundle as well as the storing of new article revisions. For the purpose of the secure wiki model plugin, which relies on an SQL datasource, functions to select, insert, update and delete has been added in an `SQLDataService` interface, which extends the normal data service interface.

Formating Service The `ArticleFormatterService` interface is used to define an OSGi-service used to format the article markup into html. The formatter has been separated into an OSGi-service such that other bundles may use this, if needed. This service could easily be moved to a separate bundle, but so far this has not been necessary.

Lookup Service The `LookupService` interface defines an OSGi-service that functions as the mediator between different parts of the system. Strings defined by one bundle can be looked up by another bundle using the same defining key and get the value registered for that key.

Plugin Service The `PluginService` interface is used to define and OSGi-service and used by plugins to register themselves as well as the core to access all registered plugins. The plugin service differentiates between integrity plugins and presentation plugins allowing the core to retrieve only the relevant plugins.

5.2.3 Plugins

The API bundle also defines the interfaces describing the plugin-API, which allows the presentation and integrity plugin to be called.

Presentation Plugin The `PresentationPlugin` interface has a single method, *execute*, that is called with a view model as parameter. The presentation plugin can then call methods on the viewmodel to manipulate its content.

Integrity Plugin The `IntegrityPlugin` interface defines two functions. A function that is called before the storing of an article revision and one that is called afterwards. The pre-store function allows the secure wiki model plugin to perform its *can_ edit* check, and indicate if the storing of the article should be aborted. The second is used to update the database with the integrity level of the new revision.

5.3 Data

The data bundle is responsible for managing the data source. The data source used by the data bundle is a MySQL database. The Data bundle provides and implementation of the `SQLDataService` defined by the API bundle. Because the data bundle is responsible for fetching article and user information, the data bundle contains an implementation of the article and user models. Because the system is running in an application server, the database connection is defined by the web-application in the context file of the web application. The file must be installed in the applicaiton server by the administrator of the system prior to the installation of the wiki web application. The content of this file and it installation is documented in Appendix B. Creation of the database used as the datastorage backend is documented in Appendix E.

5.4 Core

The core bundle contains the functionality that handles the control part of the Model-View-Control pattern that java web applications are encouraged to use. For each major feature in the base system, the core has a controller (servlet). These are described in the following.

5.4.1 Authentication

The authentication servlet (`AuthenticationServlet`) is the part of the system that handles the security of user creation and identification. When creating a user, the servlet will protect the password using a SHA-256 hash algorithm and store the hash instead of the users password. When authenticating a user, the password provided by the user will be protected in the same way and compared with the value stored in the database. The implementation does not guard against automated user creation. Methods to achieve such protection are well documented and in use in several places [17, 2, 16, 10, 18]. In a production ready system, such prevention should

be implemented, but the lack thereof is of no consequence to the current implementation of the secure wiki system in terms of it being a proof-of-concept system for the secure wiki model. The security of the user-creation process is assumed. The solutions for ensuring, that this assumption is valid, exists and the implementation of these solutions are left for future work.

5.4.2 Article

The article servlet (`ArticleServlet`) is responsible for decoding the requested url to extract the identifier of the article. Using the url, the article is retrieved from the database and the markup of the article is formated, using the formating service. Having formated the article markup into xhtml, the article model is forwarded to the frontend for display.

5.4.3 Edit

When a users wishes to edit an article, the edit servlet (`EditServlet`) will fetch the article markup using the data service and forward this to the frontend where it will be presented to the user for editing. When the user has modified the markup, the edit servlet will be called to either create a preview of the modified markup or to store a new revision. To create a preview, the modified markup is formated into xhtml and added to an article model sent to the edit-page. The modified markup is forwarded to the edit-page as well to be displayed to the user for further editing.

The edit servlet is aware of the integrity plugin and calls this before creating a new revision. If the integrity plugin does not indicate a problem, the revision will be stored using the data service and the integrity plugin is notified that the revision has been stored.

5.4.4 Revert

The revert servlet (`RevertServlet`) is called with a parameter indicating the revision to undo. Using the data service, the revision previous to the revision to undo is retrieved and forwarded to the edit-page to be edited and stored as a new revision. No attempts are made to identify the specific changes between the revisions and revert them from later revisions. This has the unfortunate side-effect that reverting a revision will revert all revisions after it. This is acceptable for the moment, as most revisions are either reverted quickly or not at all [1].

5.4.5 Contribution

The contribution servlet (`ContributionServlet`) serves simply as input validation and mediator between the data service and the frontend. The contribution servlet uses the data service to retrieve the contributions made by the username given to it as input and forwards the model object it receives directly to the frontend.

5.4.6 History

The history servlet (`HistoryServlet`) is a mediator between the data service and the frontend. Retrieving the history of an article is handled by the data service which returns a history object, which is then passed on to the frontend.

5.4.7 Diff

The diff servlet (`DiffServlet`) is responsible for calculating the diff between two article revisions. The diff servlet fetches the revisions using the data service. Using the third-party library java-diff, which is also used by JamWiki, it calculates the differences between the two revisions and transforms the output from java-diff to a diff model object that is forwarded to the frontend.

5.4.8 Search

The search servlet (`SearchServlet`) performs input validation and passes the keywords on to the data service to perform the search. The data service returns a list of articles, which are passed on to the frontend.

5.4.9 Formating Service

The formating service (`WikiCoreAntlrFormatterService`) is a frontend to an ANTLR-generated parser that reads the article markup. The formating service then uses the parser output to generate the xhtml that corresponds to the markup-input. The creation of the parser is an automated process, based on an input grammar to the ANTLR parser-generation tool. Since maven is used to control the build-process of the project, the generation of the parser has been incorporated into the build-process.

5.5 Frontend

The frontend implementation consists primarily of JSP files. To prepare the web application to use the JSP files contained in the frontend bundle, the frontend bundle activator uses the functionality of the Equinox JSP Jasper and Equinox JSP HTTP-helper bundles to register each JSP file as a servlet having the same name as that of the JSP file. In addition to the JSP files, a number of resources, such as images, javascript files and css-stylesheets, needs to be registered as resources to be available for direct download as is necessary when browsers renders the generated html of the jsp-pages.

The frontend is implemented using a master-layout which then includes the content specific jsp pages. Using the viewmodel, the master layout can determine the correct content file to include along with the content to include in the head, navigation and action menu and before, after and next to the content. The frontend bundle also contains a presentation plugin implementation used to modify the viewmodel for each page that is displayed. This presentation plugin is required since the navigation and action menu are part of the master layout and changes to the these are out of reach of the content-specific pages.

The decision to structure the frontend in this manner is partly due to page maintenance being easier when only one file defines the layout and partly because of the needs of plugins to create content-specific pages. If the layout were to be embedded into every page, plugins would not be able to seamlessly integrate into any frontend theme used by the system, unless separate plugin frontend implementations were made for every theme available to the base system.

5.6 Secure Wiki Model Plugin

The secure wiki model plugin consists of an integrity plugin, that implements the static access control, a presentation plugin for displaying article ILs and a number of servlets and JSP-files that implements the dynamic part of the secure wiki model. The presentation plugin is also responsible for modifying the navigation menu such that users can access the review-related pages supplied by the secure wiki model plugin bundle. In addition, the presentation plugin is used to modify the edit-page to include an IL-selector used when saving a new revision.

5.6.1 Review

The review servlet (`ReviewServlet`) is used to display the details of a review. In case the servlet is not instructed to display a valid review, the servlet will display a list of ongoing reviews. The servlet also handles the registration of a vote cast by a reviewer.

5.6.2 Requesting a Review

The request-review servlet (`RequestReviewServlet`) will display a form that allows users to request either a promotion or a demotion review to begin. When the servlet receives the request to start a review, it will randomly select a number of users and assign these as reviewers for the review. The number of users that are randomly selected are defined by r_i of the policy and the value set using the configuration service interface of the web console. Reviews are created with an expiration date of two weeks.

5.6.3 Processing a Review

To process finished reviews, the implementation uses a `ProcessReviewWorker` class that implements the functionality that evaluates the result of all expired reviews as specified in Sections 4.3 and 4.4. Depending on the type and level of the specific review, the worker will use the condition defined by (9), (10), (11), (12) or (13) to evaluate the review. The worker will, if the evaluation condition indicates that the review is successful, promote or demote the relevant user depending on the type of the review. The worker is scheduled to run once every hour, thereby concluding any finished review.

5.7 Modifications of Third Party Components

A few of the components required to implement the secure wiki system was either not available in bundle form or contained errors that were critical in relation to the project. Fixing these problems have created artifacts not generally available. As described above, the bridge application handles http-requests and forwards these requests to the servlet registered to handle that request. Using the MVC pattern, servlets will commonly include and/or forward a request to another servlet or JSP page which in turn can do the same. The Apache Felix OSGi-framework has built-in support for this, but a critical bug in the system meant that the process would fail. In order for the system to function as intended it was necessary to patch the system and create a working bundle. Creation of the patched bundle is documented in Appendix C.

In order to load JSP-pages from bundles, they need to be registered as as a servlet with the OSGi-framework. The apache Felix framework does not directly support jsp-pages. As part of the development of the Eclipse Equinox OSGi-framework, the Eclipse foundation have

developed the code that extends an OSGi-framework with JSP support. The code relies on the Apache Tomcat JSP-engine and therefore fits nicely with the use of the Apache Tomcat application server. OSGi being a standard means that bundles intended for one framework will work just as good in other frameworks. This means that the Apache Felix framework can be extended with JSP support using the bundle created by the Eclipse foundation. This bundle is generally available as a bundle, but a dependency of the bundle is not. The source code for the missing classes and dependencies of those are generally available through the Eclipse foundation CVS server. The source code have therefore been downloaded, compiled and packaged into a bundle which is then installed in the OSGi-framework. Creation of the bundle is documented in Appendix D.

6 Evaluation

The policy, Π_2, defined in this thesis has been defined using variables. The value of these variables are intended to be adjusted to fit the specific needs of the specific wiki system that uses the policy Π_2 with the secure wiki model. However, in order to evaluate the effectiveness of Π_2, a number of assumptions, regarding a specific system and the environment, in which it exists, must be made.

In the following, the values for each of the variables will be discussed and an estimate of a value will be used to compare Π_2 with Π_1.

6.1 Evaluation Scenarios

In the following, two scenarios will be used as basis for the evaluation. The first is a basic model of a wiki system. The community of this wiki consists of levels with 100 users in each. Although this even distribution of users is unlikely, the basic model has the advantage that the number of reviewers can be translated directly into percentages.

In real-life systems, everyone can register an account and thus get to the first level. A subset of these users will have the skills and motivation to get to the next level. A subset of these users will have the skills and motivation to get to the next level and so on. If equating an edit in Wikipedia, with an amount of work that the user is willing to make, user-contribution statistics for Wikipedia [19], which groups contributors based on the number of edits they have made, can be used to estimate a grouping of users based on the amount of work they will make. This of course speaks nothing of the quality of the individual edits or whether the edits were a correction of a typo, or the addition of a large section. However, a user with many edits is more likely to have written much and of reasonable quality, compared to users with a low number of edits, which is not to say that low-edit users write content of low quality.

The statistics [19] covers Wikipedia from the beginning in 2001 to September 2008. Table 1 shows the data for the last month of available data, which is representative for the last year of available data (see Appendix A where the statistics have been partially reproduced).

X	1	5	20	100	250	1000	2500	10000
Per Month	68.3%	21.3%	7.2%	2.1%	1.0%	0.1%	0.0%	0.0%
Total	64.8%	23.1%	7.8%	2.3%	1.2%	0.5%	0.2%	0.1%

Table 1: Percentages of active authors with $\geq$ X edits to Wikipedia, September 2008 [19]

Although the distribution of users in terms of their edits to Wikipedia cannot be translated directly into an integrity level user distribution, what the statistics does show is that there is a pyramid shaped distribution of users in terms of how much work they put into the creation and maintenance of content in Wikipedia. This suggests that a more real-life scenario is a scenario where the user distribution has a pyramid shaped distribution.

Assuming a user-distribution that matches the one given by the *total*-row in Table 1, the relative level sizes can be calculated. Table 2 shows the level size ratio and a normalization of these for easier comparison. From this table, an estimate of a realistic user-distribution can be

i	Original	Normalized
0	64.8 : 23.1 : 7.8	8.3 : 2.96 : 1
1	23.1 : 7.8 : 2.3	10 : 3.4 : 1
2	7.8 : 2.3 : 1.2	6.5 : 1.9 : 1
3	2.3 : 1.2 : 0.5	4.6 : 2.4 : 1
4	1.2 : 0.5 : 0.2	6 : 2.5 : 1
5	0.5 : 0.2 : 0.1	5 : 2 : 1

Table 2: Ratios of level sizes, based on Table 1

determined and be used as the basis for the second scenario. A fair approximation to the two first rows of the table is the ratio 9:3:1. The remaining levels can be approximated with the ratio 6:2:1. Since the lower levels are the ones that will be attacked first, these are of higher interest. The second scenario used in the evaluation will therefore have user distribution where levels are 3 times larger than the next. Specifically, $|\Lambda_i| = 3 \cdot |\Lambda_{i+1}|$. Given this size-ratio of levels, and arbitrarily choosing $|\Lambda_{i+2}| = 50$, it follows that $|\Lambda_{i+1}| = 150$ and $|\Lambda_i| = 450$.

In order to evaluate the policies Π_1 and Π_2, both policies will be simulated to determine the probability of control, given all possible distributions of malicious users, ranging from none to all users behaving maliciously. The simulation will, based on a set of given values for the variables used to define Π_1 and Π_2 simulate reviews for each possible combination of malicious users. For each combination, the review is simulated 100 times, in order to find the probability of controlling a review.

The work/probability plots are based on the results of these simulations. A simulation results in a value for the probability of control for each combination of malicious users. The work/probability plots described in the next section and used throughout the evaluation-section are created based on the results of these simulations, using the user-combination/work mapping in (17).

6.2 Estimation of Work

The number of malicious users (x,y,z) at levels L_i, L_{i+1} and L_{i+2} required to violate the policy Π_2 is an important metric for the security of Π_2. However, given the fact that multiple combinations, numbering in the thousands, give the same probability of controlling a review and cannot be ordered, it is practically infeasible to document the characteristics, in terms of x,y,z, of the combinations that has a given probability of controlling a review.

The reason for the importance of the number of malicious users is because they give an indication of the effort needed by an attacker in order to violate the given policy. Using the fact that some of these combinations requires less work to achieve than others, and that attackers

will target the cheapest combinations that has a given probability, the combination of users can be translated into the work required to achieve that specific combination, using the work-function in (17). The work required to get a given combination gives a significant characteristic for combinations (x,y,z) and can be used to order these. In addition, work/probability gives a good measure for the security of the system, given an upper bound on the amount of work an attacker can perform.

Using $|\Lambda_{Rz_i}|$ to denote the set of malicious users that are selected from level L_i to participate in a review, an attacker will be able to control a review if the condition in (14) is fulfilled.

$$\begin{aligned} |\Lambda_{Rz_i}| \cdot \mathcal{W}_i + \left|\Lambda_{Rz_{i+1}}\right| \cdot \mathcal{W}_{i+1} + \left|\Lambda_{Rz_{i+2}}\right| \cdot \mathcal{W}_{i+2} \geq \\ \left(\mathcal{S}_i^{\max}(a) + \mathcal{S}_{i+1}^{\max}(a) + \mathcal{S}_{i+2}^{\max}(a)\right) \cdot \tau_i \end{aligned} \tag{14}$$

Given the random selection of users, it can be assumed that in order to control x% of the reviewers from a given level, an attacker must control x% of the level. The amount of work needed to get a given combination of malicious users z_i, z_{i+1}, z_{i+2} included in a review is therefore not limited to the promotion of z_i, z_{i+1} and z_{i+2} users to levels L_i, L_{i+1} and L_{i+2} respectively, but to the users needed to have z_i, z_{i+1}, z_{i+2} users selected to perform the review.

The work required to have one user promoted from level L_i to level L_{i+1} is denoted by α_i. In order for an attacker to have a malicious user added to the system and then promoted to level L_i is then

$$\sum_{j=0}^{i-1} \alpha_j \tag{15}$$

To have x users at level L_i, the attacker will have to make an effort x times greater.

$$\sum_{j=0}^{i-1} \alpha_j \cdot x \tag{16}$$

In order to get a given x, y, z combination of users at levels L_i, L_{i+1} and L_{i+2} respectively, the effort required is

$$\sum_{j=0}^{i-1} \alpha_j \cdot x + \sum_{j=0}^{i+1-1} \alpha_j \cdot y + \sum_{j=0}^{i+2-1} \alpha_j \cdot z \tag{17}$$

Using (17) as a mapping between combinations of malicious users and the work required to achieve that combination allows the creation of work/probability plots, that shows the work needed by an attacker in order to achieve a given probability of controlling a review. As attackers must be assumed to target the cheapest combinations of users, the important parameter becomes the work required to achieve a given probability and not the specific combinations that give the same probability.

The work-function in (17) is defined based on the values of α_i. The value of α_i for all i can therefore affect the security of the policy. If using a linear work-function where the elemental work required to be promoted from one level to the next is constant, i.e. that $\alpha_0 = \alpha_1 = \ldots = \alpha_{|\mathbb{I}|-1}$, a work/probability plot can look like shown in Figure 10.

When looking at the work/probability plots, the important things to consider is the left and right edges of the colored areas. These areas will be referred to as the *areas of vulnerability* for Π_1 and Π_2 respectively. The left edge of an area indicates the minimum amount of work an attacker is required to perform in order to have a given probability of controlling a review.

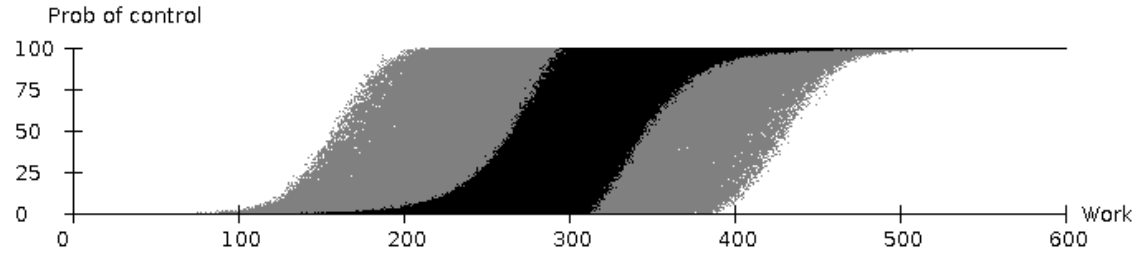


Figure 10: Plot of policy security for L_1 review using linear work-function.
Π_1 (in gray): $|\Lambda_i| = [100, 100, 100]$. $r_i = [20, 20, 20]$. $\tau_i = [10, 10, 10]$.
Π_2 (in black): $|\Lambda_i| = [100, 100, 100]$. $r_i = [20, 20, 20]$. $\mathcal{W}_i = [1, 2, 4]$. $\tau_i = 50\%$.

This is important because smart attackers will attempt to control a combination of malicious users that requires the least amount of work, i.e. a combination that results in a work value that lies on the left edge. The right edge of an area indicates the maximum amount of work that an attacker will have to do in order to have the given probability of control, even when using an undirected brute force approach. Any work beyond the right edge is guaranteed to increase the probability of control.

To get a sense of the work parameter, it should be noted, that in order to control 25% of the three levels included in a review, the amount of work required, when using the linear work-function is $25 \cdot (1) + 25 \cdot (1+1) + 25 \cdot (1+1+1) = 25 + 50 + 75 = 150$. Controlling 25% of each of the levels is difficult to impossible, depending on the size of the system. The larger the system, the less feasible it will be to control 25% of the levels. In addition, when comparing this to Figure 10, it can be seen that this has little probability of controlling a Π_2 review and only a 50% chance of controlling a Π_1 review.

From Figure 10, it can be seen that an attacker must perform 100 units of work, in order to have even a small probability of controlling a Π_1 review. For Π_2, the necessary work is close to 200. The left edge for Π_1 also shows that a smart attacker will require a little more than 200 units of work to have 100% probability of controlling Π_1 reviews. The corresponding value for Π_2 is around 300.

Figure 11 shows a work/probability plot of the same simulation, but using a different work-function. For this plot, it is assumed that it requires progressively more work to be promoted to the next level, i.e. $\alpha_i > \alpha_{i-1}$. The specific α_i-values used for the work-function are $\alpha_0 = 1$, $\alpha_1 = 2$, and $\alpha_2 = 3$.

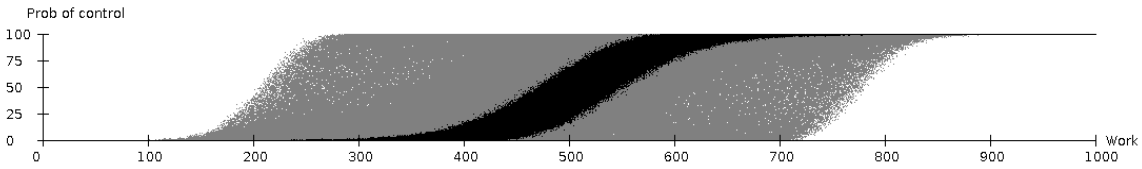


Figure 11: Plot of policy security for L_1 review using increasing work-function.
Π_1 (in gray): $|\Lambda_i| = [100, 100, 100]$. $r_i = [20, 20, 20]$. $\tau_i = [10, 10, 10]$.
Π_2 (in black): $|\Lambda_i| = [100, 100, 100]$. $r_i = [20, 20, 20]$. $\mathcal{W}_i = [1, 2, 4]$. $\tau_i = 50\%$.

From Figure 11, it can be seen that for a Π_1 review, an attacker must perform 100 units of work and 150 before the attacker will have any significant probability of control. For an attacker to have 100% probability of control, the attacker must perform around 275 units of

work. For this amount of work, an attacker has at most 2% probability of control for a Π_2 review. Full control for a Π_2 review is not achieved until having performed around 575 units of work.

Figures 10 and 11 illustrates the effect of changing the work-function. Increasing the value of α_i increases the maximum value of the work-function and therefore the horizontal axis of Figure 11 is extended to match this. In addition, the work required to get a given combination of users is increased. The increase of α_2 from 1 to 3, while keeping $\alpha_0 = 1$ means that the combinations, with users in the higher levels, requires more work to achieve. These combinations are also the ones that have a higher probability of control. The more work required for higher probabilities means that the shape of the area of vulnerability for Π_2 shifts further to the right, for higher probabilities of control, relative to the lower probabilities of control. The fact that Π_2 moves further right shows that Π_2 is more susceptible to changes in the work-function than Π_1.

In a real-life system, the α_i parameter is not something that can be configured, but is implicitly defined as a consequence of the requirements imposed by the community. If a community has a low standard for promotion, α_i will be correspondingly low. If the community on the other hand requires a high standard for promotion α_i will be correspondingly high. For each level, a different opinion of the level of standard may exists which will cause α_i to differ for each i. Administrators can attempt to affect α_i by defining guidelines, but in the end has little direct control of α_i.

In the following, the work-function used will be the linear work-function described above. Without loss of generality the effort needed to be promoted from level L_i to L_{i+1} is assumed to be 1 as other values simply results in a scaling of the result.

6.3 Estimation of Number of Reviewers

The number of reviewers that participate in a review (r_i) has an effect on the robustness of the review in terms of reaching the decision of the reviewers. The number of reviewers from each level to include in a review can be set in different ways. The simplest approach is to define r_i as a percentage of the users at level L_i. Assuming a pyramid-shaped distribution of users in the level-hierarchy, this will result in a pyramid-shaped distribution of users in the three consecutive levels included in the review. An alternate approach will be to fix the value of r_i to a constant value, such that the number of users included in a review is the same for each level L_i, L_{i+1} and L_{i+2}.

The relative size between Λ_{R_i}, $\Lambda_{R_{i+1}}$, $\Lambda_{R_{i+2}}$ will affect the influence each level has on the review decision. When $r_i = r_{i+1} = r_{i+2}$, the influence each level has on the review decision is determined by the weights given to the votes at each level. When $r_i > r_{i+1} > r_{i+2}$, Level L_i will have a greater influence on the review decision than the weight $\mathcal{W}_i$ would otherwise suggest. With more users at the lowest level, an attacker will have more malicious users included in the review and have a greater influence on the review decision.

Figure 12 shows the simulation of an L_0 to L_1 review for Π_1 and Π_2, using a pyramid shaped user-distribution with $r_i = 20\%$ of $|\Lambda_i|$, equal weights and linear work-function with no cost of user registration requiring only a simple majority. The obvious problem with this configuration is that the left edge of the area of vulnerability for Π_2 is a vertical line up from work=0. This means that an attacker only needs to register, but not promote, malicious colluding users to control a review. The reason for this is the large number of reviewers from L_0, which in itself can account for the score needed to reach the simple majority needed to perform a successful

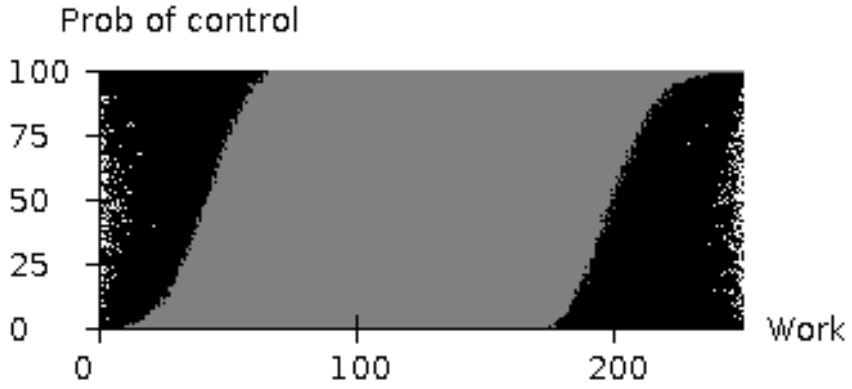


Figure 12: Plot of policy security for L_0 review using linear work-function and equal weights.
Π_1 (in gray): $|\Lambda_i| = [450, 150, 50]$. $r_i = [90, 30, 10]$. $\tau_i = [45, 10, 5]$.
Π_2 (in black): $|\Lambda_i| = [450, 150, 50]$. $r_i = [90, 30, 10]$. $\mathcal{W}_i = [1, 1, 1]$. $\tau_i = 50\%$.

review. For an L_0 review, the problem is further complicated by the zero entry cost into level L_0. Π_1 on the other hand still requires enough users at level L_1, to get a simple majority in the L_1 vote. This fact is the reason that the Π_1 area has a left edge starting from a work of 10 up to a work of 66 for 100% probability of control. The right edge of the Π_2 area is also a vertical line. This is caused by the fact that an attacker can perform the work needed to control 100% of the users in levels L_1 and L_2, and still not have the users needed to exceed the 50% threshold. Having partial control of levels L_1 and L_2 in an L_0 Π_1 review, will result in a probability of control of the review, giving Π_1 its right edge in the figure.

Figure 12 illustrates the fact that differences in the relative size between Λ_{R_i}, $\Lambda_{R_{i+1}}$, $\Lambda_{R_{i+2}}$ will have a measurable effect on the security of Π_2, compared to Π_1, where size-differences in Λ_{R_i}, $\Lambda_{R_{i+1}}$, $\Lambda_{R_{i+2}}$ is insignificant. In order to improve the security of Π_2, a number of approaches can be taken. Raising the threshold, τ_i, can increase the score needed to have a successful review and therefore will increase the number of users needed at levels L_{i+1} and L_{i+2}. However, for the current case, the threshold must be above 70% before for this approach to have an effect. Another approach will be to use the weights to shift the power from the lower to the upper levels.

Figure 13 shows a simulation using the same configuration as that of Figure 12, except for the weights which have been modified to shift power away from the lowest level, such that each level accounts for a third of the maximum score. Since only the weights are different, the area of vulnerability for Π_1 is the same as that of Figure 12. For Π_2, the change of weights significantly increases the effort required by an attacker, to violate Π_2. This increase in effort is sufficient to increase the security of Π_2 to a level slightly better than the security of Π_1. What is not immediately obvious from the figure is that the right edges of Π_1 and Π_2 are roughly in the same location in the plot. Given a stable user-distribution, the number of reviewers can be chosen as a fixed fraction of the level-size and the relative strength of each level can be corrected using the weights. Using the weights to correct issues caused by size-differences can, however, be difficult, if the level-sizes are unstable. Another approach is to use an equal number of reviewers at each level, and only use the weights for controlling the relative power between levels.

This approach is taken in Figure 14, which shows the result of a simulation using the same

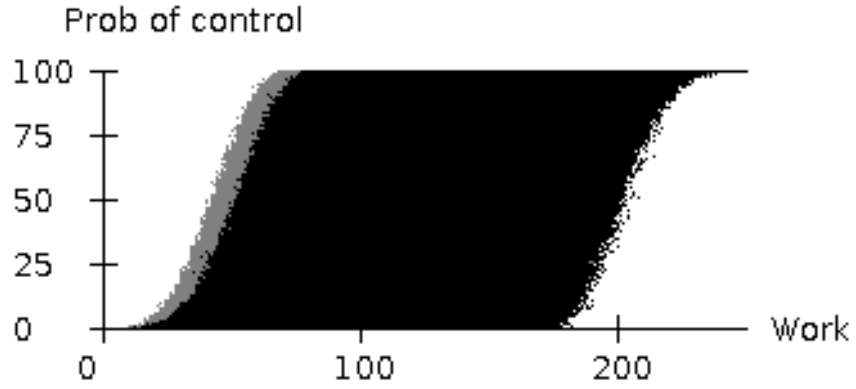


Figure 13: Plot of policy security for L_0 review using linear work-function.
Π_1 (in gray): $|\Lambda_i| = [450, 150, 50]$. $r_i = [90, 30, 10]$. $\tau_i = [45, 15, 5]$.
Π_2 (in black): $|\Lambda_i| = [450, 150, 50]$. $r_i = [90, 30, 10]$. $\mathcal{W}_i = [1, 3, 9]$. $\tau_i = 50\%$.

configuration as that of Figure 12, except the number of reviewers are equal. Using equal

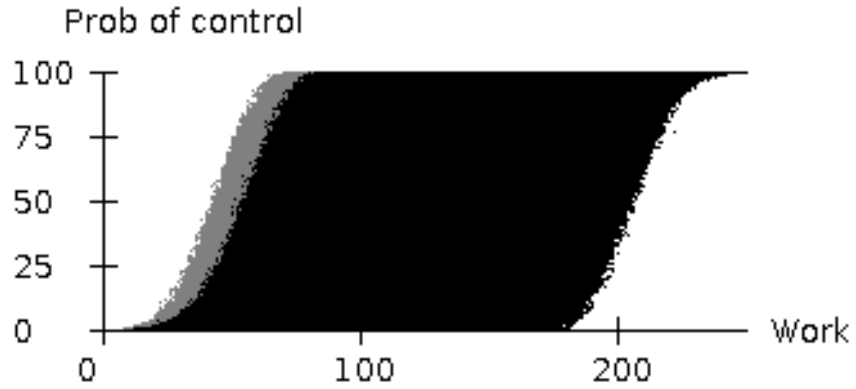


Figure 14: Plot of policy security for L_0 review using linear work-function and equal weights.
Π_1 (in gray): $|\Lambda_i| = [450, 150, 50]$. $r_i = [10, 10, 10]$. $\tau_i = [5, 5, 5]$.
Π_2 (in black): $|\Lambda_i| = [450, 150, 50]$. $r_i = [10, 10, 10]$. $\mathcal{W}_i = [1, 1, 1]$. $\tau_i = 50\%$.

weights and equal number of reviewers effectively means that each level controls one third of the total score. This effect is the same as the one obtained by using the weights in Figure 13, however, this distribution is guaranteed to be stable, regardless of fluctuations in the sizes of the levels. With a stable number of reviewers in each level, the power-ratio between levels can be guaranteed to be the same as the ratio between the weights. This means that the weights can easily be modified to strengthen the power of one or more levels, relative to the others.

As shown by Figures 12, 13 and 14, it is possible to use a non-equal number of reviewers, if the weights are adjusted accordingly. However, in the following, the number of reviewers will be equal, i.e. $r_i = r_{i+1} = r_{i+2}$, which will eliminate the problem caused by size-differences.

6.4 Estimation of Level Weights

The weights ($\mathcal{W}_i$) used by Π_2 are there to control the difference in power between the reviewers included in a review. As shown above, using the weights, a level can be given more or less

influence on the outcome of the review and thereby be used to increase or decrease the security of Π_2. As demonstrated by Figures 12, 13 and 14, using weights that increases the influence of the highest level significantly improves the security of Π_2. The simulation illustrated in Figure 15 uses a user-distribution with an equal number of users at each level, along with an equal number of reviewers from each level and using equal weights. This figure can be seen to be

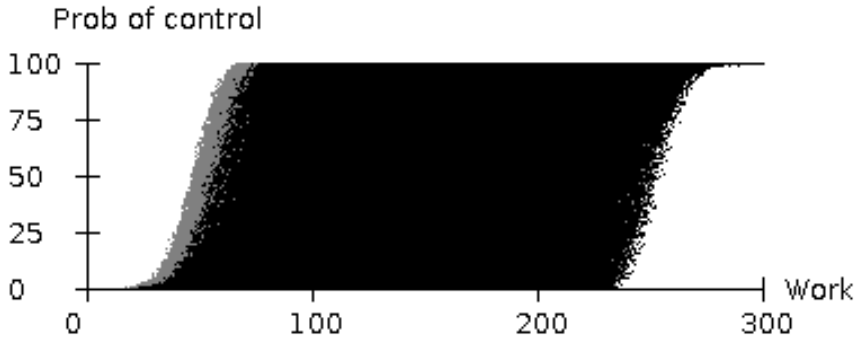


Figure 15: Plot of policy security for L_0 review using linear work-function.
Π_1 (in gray): $|\Lambda_i| = [100, 100, 100]$. $r_i = [20, 20, 20]$. $\tau_i = [10, 10, 10]$.
Π_2 (in black): $|\Lambda_i| = [100, 100, 100]$. $r_i = [20, 20, 20]$. $\mathcal{W}_i = [1, 1, 1]$. $\tau_i = 50\%$.

similar to Figure 14, except for the amount of work needed, due to the difference in level size. Like in Figure 14, the right edges of Π_1 and Π_2 in Figure 15 are located in the same location in the plot. For Π_2, an attacker needs to perform 34 units of work, in order to have a 5% probability of controlling a review. For 100% probability of control, the work required is 73. The corresponding numbers for Π_1 is 29 and 66.

As illustrated by Figure 12, giving the lowest level greater influence than the highest level will significantly reduce the security of the policy. For this reason, $\mathcal{W}_i$ should be defined such that $\mathcal{W}_i < \mathcal{W}_{i+1}$.

Figure 16 is based on the same simulation as that of Figure 15, but shows an L_1 to L_2 review. The difference between the L_0 and L_1 review is that there is a cost on controlling

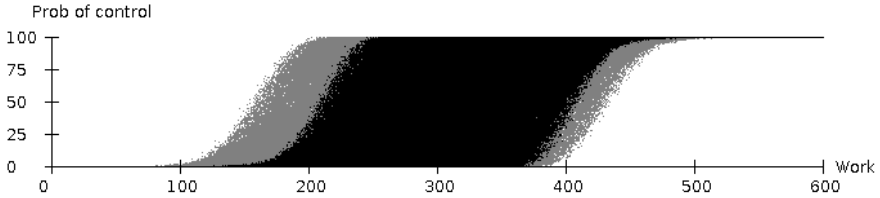


Figure 16: Plot of policy security for L_1 review using linear work-function.
Π_1 (in gray): $|\Lambda_i| = [100, 100, 100]$. $r_i = [20, 20, 20]$. $\tau_i = [10, 10, 10]$.
Π_2 (in black): $|\Lambda_i| = [100, 100, 100]$. $r_i = [20, 20, 20]$. $\mathcal{W}_i = [1, 1, 1]$. $\tau_i = 50\%$.

users at all levels. This extra cost, relative to the L_0 review in Figure 15, is clearly seen as a right-shift of the location of the areas of vulnerability. For the configuration used in Figure 16, an attacker must perform 113 units of work to have a 5% probability to control a Π_1 review and 202 to have 100% probability of control. For Π_2, the corresponding values are 166 and 245. From these values, it can be seen that the left edge of Π_2 shifts further right, than Π_1.

For both Π_1 and Π_2, the width, in terms of work, of the areas of vulnerability increases, but the Π_1 area increases the most. The right-edge of Π_1 moves further right because of the possibility of an attacker wasting effort on compromising L_1, without gaining anything, when the attacker has already partially compromised L_2 and L_3, which is the most expensive attack.

In order to have weights increase with levels, the simplest definition is $\mathcal{W}_i = i$. Figure 17 displays the effort required by an attacker to violate Π_2 when using $\mathcal{W}_i = i$. The configuration

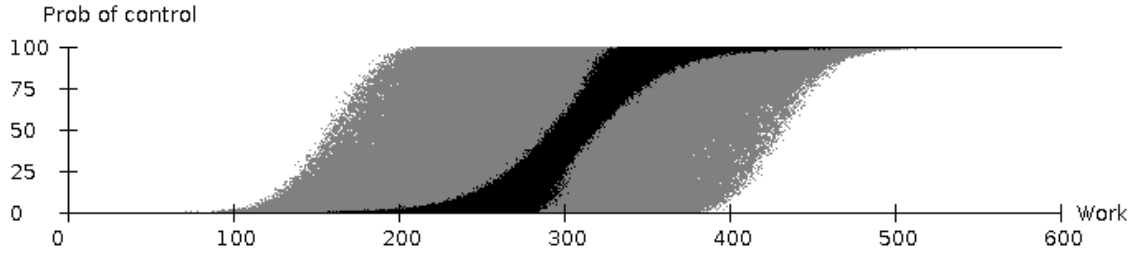


Figure 17: Plot of policy security for L_1 review using linear work-function.
Π_1 (in gray): $|\Lambda_i| = [100, 100, 100]$. $r_i = [20, 20, 20]$. $\tau_i = [10, 10, 10]$.
Π_2 (in black): $|\Lambda_i| = [100, 100, 100]$. $r_i = [20, 20, 20]$. $\mathcal{W}_i = [1, 2, 3]$. $\tau_i = 50\%$.

is equal to that used in Figure 16, except for the weights. This means that the Π_1 area is equal to that of Figure 16. For Π_2 an attacker must perform 217 units of work in order to have a 5% probability of controlling a review and 328 to have 100% probability of control. This represents a significant improvement in security, even though the area of vulnerability is relatively thin. This low width means that there will be little difference in the effectiveness of a brute force attack and an attack directed at the left edge.

Using the weight $\mathcal{W}_i = i$ has the consequence that the relative power ratio between the highest and lowest level decreases as i increases. Table 3 shows the relative power ratio between the highest and lowest level for $i <= 5$. The problem is that, given enough levels, the relative

i	power ratio	low-high-ratio
0	0:1:2	
1	1:2:3	1:3.0 (1:3)
2	2:3:4	1:2.0 (2:4)
3	3:4:5	1:1.6 (3:5)
4	4:5:6	1:1.5 (4:6)
5	5:6:7	1:1.4 (5:7)

Table 3: Power-ratio for votes when $\mathcal{W}_i = i$

power ratio would asymptotically approach a 1:1 ratio, which is undesirable. To mitigate this situation, the weights can be increased, such that they maintain a fixed power-ratio for all reviews, independent of the placement in the level-hierarchy. This power-ratio can be achieved by defining $\mathcal{W}_i = 2^i$, effectively doubling the weight of the previous level as i increases and fixing the power-ratio to 1:2:4. Figure 18 displays the effort required by an attacker to violate Π_2 when using $\mathcal{W}_i = 2^i$. The configuration is equal to that used in Figure 16, except for the weights. For Π_2 an attacker must perform 192 units of work in order to have a 5% probability of controlling a review and 294 to have 100% probability of control. This is less than that of Figure 17, but still an improvement over Figure 16.

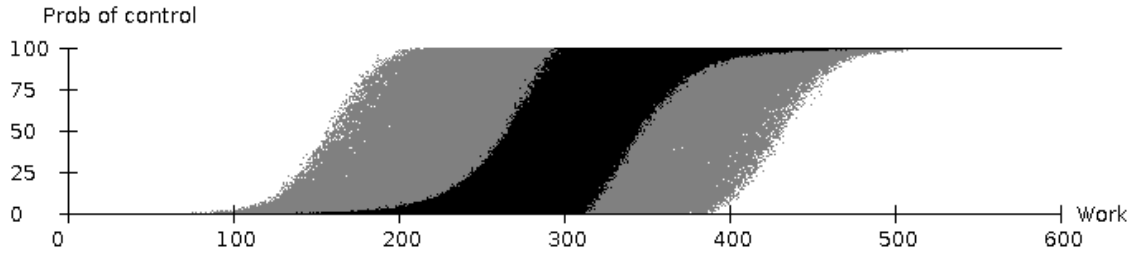


Figure 18: Plot of policy security for L_1 review using linear work-function.
Π_1 (in gray): $|\Lambda_i| = [100, 100, 100]$. $r_i = [20, 20, 20]$. $\tau_i = [10, 10, 10]$.
Π_2 (in black): $|\Lambda_i| = [100, 100, 100]$. $r_i = [20, 20, 20]$. $\mathcal{W}_i = [1, 2, 4]$. $\tau_i = 50\%$.

Arguments for both cases can be made, depending on the specific system in which Π_2 is used. Using $\mathcal{W}_i = i$, the difference, in terms of work, between the left and right edge is relatively low. In contrast, using $\mathcal{W}_i = 2^i$ is significantly wider, but at the expense of having a left edge further left, i.e. an attacker requires less effort to violate it. It is believed that the difference is insignificant compared to the effort that an attacker will have to make. For Π_2 it holds that the greater the number of users in each level, the greater the complexity of determining the left edge of the system and the more likely it is that an attacker will use a brute-force approach to violating the policy. If the system manages to keep the values of the parameters secret, it would further complicate the calculations of the left edge. A best-effort brute-force approach is believed to follow a a path from 0% to 100% that lies between the left and right edges. Based on this, the greater difference between the left and right edge will help protect the system against attackers that are using a brute-force approach to violating the system.

In the following, $\mathcal{W}_i = 2^i$ will be used as this shifts power to the higher levels ensuring that their influence in reviews are required for a review to succeed.

6.5 Estimation of Vote Threshold

The threshold value (τ_i) is the indicator of the level of consensus that is needed in order for a review to succeed. Having a low threshold means that only a low number of reviewers needs to vote to promote a user, which makes it easier to be promoted. On the other hand, a high threshold requires the reviewers to have a high degree of consensus which helps reach the correct decision before promoting a user. The threshold value τ_i can be estimated using the estimated values for r_i and $\mathcal{W}_i$ and the review condition for Π_2 stated in (9). To estimate the effect of varying τ_i, the definition of $S_i(a)$) in (7) can replace the term $S_i(a)$ in (9). By replacing the sums over reviewers in Λ_{R_i}, $\Lambda_{R_{i+1}}$ and $\Lambda_{R_{i+2}}$ with x, y, z respectively and isolating z, the resulting equation is that in (18).

$$z = \frac{\left(S_i^{\max}(a) + S_{i+1}^{\max}(a) + S_{i+2}^{\max}(a)\right) \cdot \tau_i - \mathcal{W}_i \cdot x - \mathcal{W}_{i+1} \cdot y}{\mathcal{W}_{i+2}} \tag{18}$$

A plot of (18) will show the combination of malicious users that is needed to reach the threshold and control the review. In addition, any combination of users that falls above the plane will exceed the threshold and therefore also control the review. Using 100 reviewers at each level, where x, y and z are malicious, the plot can be interpreted in percentages, such that $x = 25$ indicates 25% of $|\Lambda_{R_i}|$. The plot in Figure 19 shows the combination of malicious users needed

to control a review using $\tau_i = 0.5$. The plot shows that given a specific value for τ_i, the

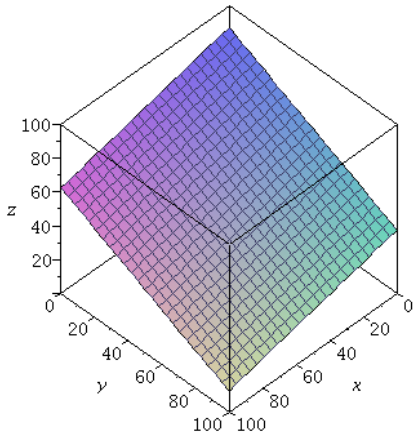


Figure 19: Plot of (18) showing the combinations of colluding reviewers needed to control a review when using $\tau_i = 0.5$

combination of users needed to control a review, forms a plane. The gradient of the plane is dependent on the value of the weights, but it clearly shows how a change in one level can be accounted for in the other levels.

For the 50% threshold and weights 1, 2 and 4, Figure 19 shows that even if an attacker controls both lower levels, (x and y), the attacker still needs control of 13 (12.5 to be precise) of the 100 users at level L_{i+2}. The figure also shows that 88 (87.5) of the 100 users at level L_{i+2} can control a review without the help of the lower levels. Controlling all of the lowest level, but none at the middle level requires an attacker to control 63 (62.5) of the 100 users at level L_{i+2}. Similarly, controlling all of the middle level, but none at the lowest requires an attacker to control 38 (37.5) of the 100 users at level L_{i+2}.

For a review from level $L_{|\mathbb{I}|-1}$ to $L_{|\mathbb{I}|}$, the review condition is that of (10). Rewriting this in the same way as was done with (9) gives the result in (19).

$$y = \frac{\left(S_i^{\max}(a) + S_{i+1}^{\max}(a)\right) \cdot \tau_i - \mathcal{W}_i \cdot x}{\mathcal{W}_{i+1}} \tag{19}$$

Since (19) only involves two levels, the result of ploting (19) is a line, not a plane. The gradient of the line is dependent on the weights used and also shows how a change in one level can be accounted for in the other level. The plot of (19) for various values of τ_i is shown in Figure 20. Like Figure 19, the line represents the combinations that reach the threshold with anything above exceeding it.

From figures such as Figure 19 and Figure 20, the required number of malicious users needed by an attacker to control a review can be found. Having decided a good threshold value, the value can be used in a simulation to see the result of using this value.

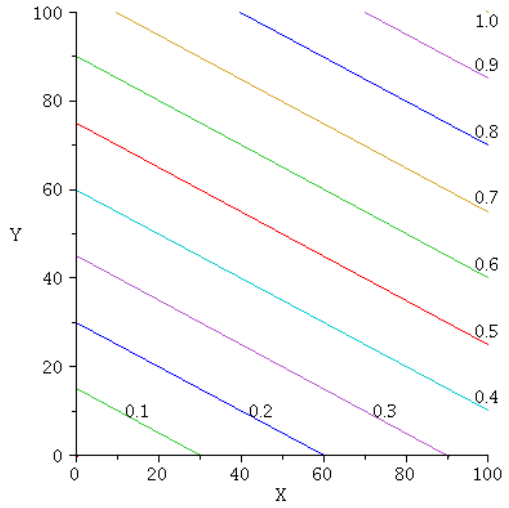


Figure 20: Plot of (19) showing the combinations of colluding reviewers needed to control a review when using $\tau_i = 0.5$

Figures 21, 22 and 23 shows the result of three different simulations. The simulations are based on the same configuration, where, for $i \in \{1, 2, 3\}$, the level size $|\Lambda_i| = 100$, the number of reviewers $r_i = 20$ and having the weights double to shift power to the higher levels, i.e. $\mathcal{W}_i = 2^i$. The simulations differ only in the value used for the threshold.

Figure 21 uses $\tau_i = 50\%$. In order for an attacker to have a 5% probability of controlling a Π_2

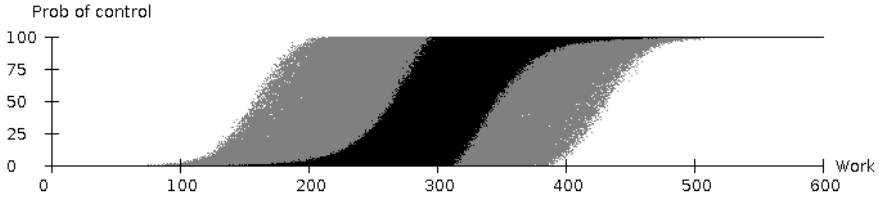


Figure 21: Plot of policy security for L_1 review using linear work-function.
Π_1 (in gray): $|\Lambda_i| = [100, 100, 100]$. $r_i = [20, 20, 20]$. $\tau_i = [10, 10, 10]$.
Π_2 (in black): $|\Lambda_i| = [100, 100, 100]$. $r_i = [20, 20, 20]$. $\mathcal{W}_i = [1, 2, 4]$. $\tau_i = 50\%$.

review, the attacker must perform a minimum of 192 units of work. To have a 100% probability of control the work required is at least 294. The worst-case, i.e. right edge, approach requires the attacker to perform 312 units of work in order to get a 1% probability of control. The worst-case effort needed for a 95% probability of control is 413. For a Π_1 review, 5% probability of control requires a minimum of 110 units of work. To have a 100% probability of control the work required is at least 207. The worst-case approach requires the attacker to perform 389 units of work in order to get a 1% probability of control. The worst-case effort needed for a 95% probability of control is 480.

Figure 22 uses $\tau_i = 60\%$. In order for an attacker to have a 5% probability of controlling a Π_2 review, the attacker must perform 262 units of work. To have a 100% probability of control the work required is 356. The worst-case approach requires the attacker to perform 355 units of work in order to get 1% probability of control. The worst-case effort needed for 95% probability of control is 462. For a Π_1 review, 5% probability of control requires a minimum of 141 units of

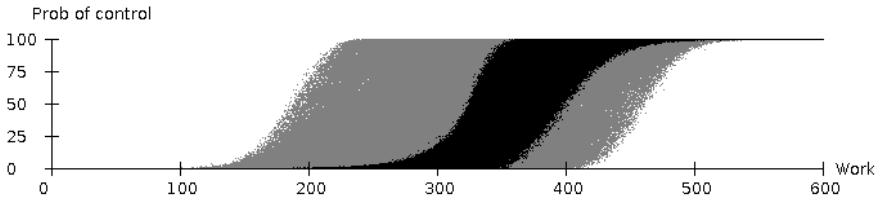


Figure 22: Plot of policy security for L_1 review using linear work-function.
Π_1 (in gray): $|\Lambda_i| = [100, 100, 100]$. $r_i = [20, 20, 20]$. $\tau_i = [12, 12, 12]$.
Π_2 (in black): $|\Lambda_i| = [100, 100, 100]$. $r_i = [20, 20, 20]$. $\mathcal{W}_i = [1, 2, 4]$. $\tau_i = 60\%$.

work. To have a 100% probability of control the work required is at least 235. The worst-case approach requires the attacker to perform 417 units of work in order to get a 1% probability of control. The worst-case effort needed for a 95% probability of control is 505.

Figure 23 uses $\tau_i = 75\%$. In order for an attacker to have a 5% probability of controlling a

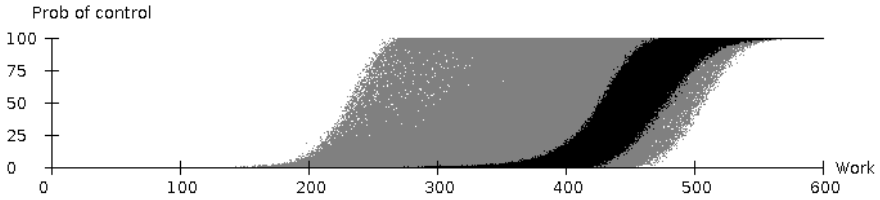


Figure 23: Plot of policy security for L_1 review using linear work-function.
Π_1 (in gray): $|\Lambda_i| = [100, 100, 100]$. $r_i = [20, 20, 20]$. $\tau_i = [15, 15, 15]$.
Π_2 (in black): $|\Lambda_i| = [100, 100, 100]$. $r_i = [20, 20, 20]$. $\mathcal{W}_i = [1, 2, 4]$. $\tau_i = 75\%$.

Π_2 review, the attacker must perform 358 units of work. To have a 100% probability of control the work required is 466. The worst-case approach requires the attacker to perform 430 units of work in order to get 1% probability of control. The worst-case effort needed for 95% probability of control is 531. For a Π_1 review, 5% probability of control requires a minimum of 186 units of work. To have a 100% probability of control the work required is at least 263. The worst-case approach requires the attacker to perform 456 units of work in order to get a 1% probability of control. The worst-case effort needed for a 95% probability of control is 546.

From these figures it can be seen how the changes in the threshold value effectively shifts the area of vulnerability to the right, indicating an increase in the amount of work needed to violate Π_2. Comparing the values of the right and left edges on the figures show that the area of vulnerability for Π_2 gets thinner as the threshold value is increased. The thinner area of vulnerability means that an attacker, using an undirected brute force approach, is more likely to perform work close to the optimal approach. Given the high amount of work required when following the left edge, it is not a problem that the width of the area decreases, since the left edge should be sufficiently secure.

In addition to Π_2 moving to the right, so does Π_1. From the figures, it can be seen that Π_2 moves significantly further right than Π_1, which illustrates how significantly the security of Π_2 increases compared to the increase in security of Π_1. Effectively, this means that Using Π_2, the requirement for consensus between reviewers can be relaxed without sacrificing security in relation to Π_1.

6.6 Required Number of Reviewers for Π_2

The concept of reviews introduced by the secure wiki model is the biggest issue when attempting to incorporate the secure wiki model into a running wiki with many users. The fact that users will have to perform a review, which is an action not usually done, may not be something many users will participate in. Therefore, the lower the number of reviewers needed to perform a review, the better chances of success for the secure wiki model.

For Π_2 any number of reviewers will suffice to have a working system. For each reviewer removed from the set of selected reviewers will reduce the work required by an attacker since the attacker will need to control fewer users in the system. This means that each reviewer removed reduces the security of Π_2. The security of Π_2 would worsen gradually and it is therefore difficult to set an absolute value for a threshold indicating when the security is insufficient.

The proposed policy Π_2 is intended to replace the original secure wiki model policy Π_1. In order for this replacement to be acceptable, the security of Π_2 cannot be worse than that of Π_1. This suggests that the lower limit of the number of reviewers used in a Π_2 reviewer is set at the number of reviewers that reduces the security of Π_2 to that of Π_1, for otherwise equal parameters.

To find this this lower limit of reviewers, a number of simulations were run, using varying values for the number of reviewers. Based on this trial-and-error approach, the results shown in Figures 24, 25 and 26 was found. For all three figures, the number of reviewers in for the Π_2

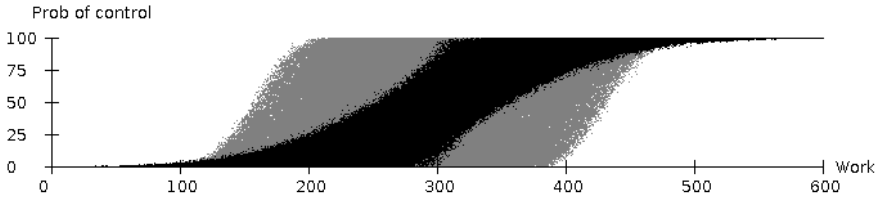


Figure 24: Plot of policy security for L_1 review using linear work function.
Π_1 (in gray): $|\Lambda_i| = [100, 100, 100]$. $r_i = [20, 20, 20]$. $\tau_i = [10, 10, 10]$.
Π_2 (in black): $|\Lambda_i| = [100, 100, 100]$. $r_i = [5, 5, 5]$. $\mathcal{W}_i = [1, 2, 4]$. $\tau_i = 50\%$.

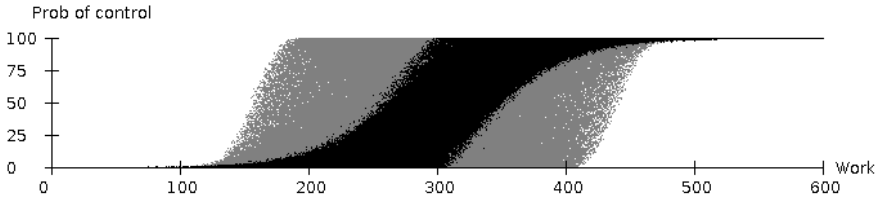


Figure 25: Plot of policy security for L_1 review using linear work function.
Π_1 (in gray): $|\Lambda_i| = [100, 100, 100]$. $r_i = [40, 40, 40]$. $\tau_i = [20, 20, 20]$.
Π_2 (in black): $|\Lambda_i| = [100, 100, 100]$. $r_i = [10, 10, 10]$. $\mathcal{W}_i = [1, 2, 4]$. $\tau_i = 50\%$.

review is a quarter of those used in the Π_1 review. From the figures, it can be seen that the left edge of the area of vulnerability for Π_2 initially follows the left edge of the area of vulnerability for Π_1 but after this initial common edge, the left edge of Π_1 increases quickly. At no time is the left edge of Π_2 to the left of Π_1, however, it is as close as it can be. From this, it can be seen that Π_2 can maintain the same level of security as Π_1 using only a quarter of the reviewers

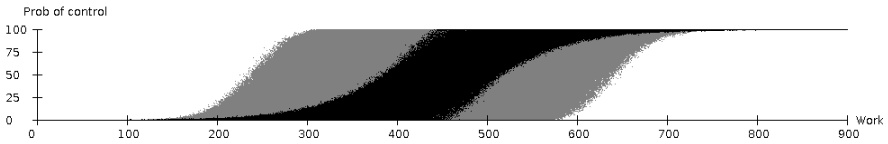


Figure 26: Plot of policy security for L_1 review using linear work function.
Π_1 (in gray): $|\Lambda_i| = [225, 150, 100]$. $r_i = [45, 30, 20]$. $\tau_i = [23, 15, 10]$.
Π_2 (in black): $|\Lambda_i| = [225, 150, 100]$. $r_i = [12, 8, 5]$. $\mathcal{W}_i = [1, 2, 4]$. $\tau_i = 50\%$.

needed by Π_1. In addition, the security of Π_2 is still significantly better than Π_1 for the higher probabilities of control.

7 Future Work

The secure wiki model, as described in this thesis, is a good and easy to use model. The model is simple to implement and can easily be incorporated into existing wiki systems without having to rewrite the base code of the wiki. The implementation has been made with the experiences of the previous implementations in mind and therefore does not use the document-review procedure as defined by the secure wiki model, but is limited to user-reviews. In the implementation, document reviews have been replaced with the ability of individual users to define the appropriate IL of documents which they can edit. In the future, the implementation should be evaluated to determine if the IL selection by individual users is sufficient to keep a stable and correct IL for documents. If this evaluation finds that the document reviews are necessary, a method for document reviews, that addresses the issues explained in Section 2.5.1, should be implemented and evaluated.

The implementation focused on the secure wiki model and made the assumption that the user-creation process is secure. In order to use the implementation in a hostile environment, this assumption must be ensured to be valid. Various methods for ensuring this assumption, such as captchas [16] exists and should be implemented in the future.

Other wiki-related functionality have also been implemented only to the extend it was necessary. Most notable is the missing feature of locking articles from being edited simultaneously by multiple contributors, which will lead to one contributor's contribution being overwritten by the second contributor. The secure wiki model plugin, as implemented in the proof-of-concept system, has the features defined by the secure wiki model and is capable of managing the IL of documents and QCV of users. The secure wiki model does not depend on users being able to communicate, but having user pages, user talk pages and article discussion pages can greatly aid in the administration of the wiki system and the secure wiki model. In general, such pages are implemented using namespaces, e.g. user, talk and discussion, with articles located in the main namespace, which is usually nameless. In the future, the base system should be extended to support namespaces, such that user pages, talk pages and discussion pages can be implemented and can be used by contributors to discuss changes to articles and the merits for promotion/demotion of users. Finally, the syntax allowed by the article markup language is not as flexible and comprehensive as that of existing wiki implementations, e.g. Wikipedia has support for template inclusion and the inclusion of images. This means that the parser used by the base system to read the article markup should be updated to support a more flexible and comprehensive syntax in order to be on par with other wiki systems and the base system

must be extended to allow upload of images.

The secure wiki model requires contributors to spend time reviewing contributions in stead of researching and writing content. In a standard wiki system, without the secure wiki model, contributors would spend time correcting vandalism. For the secure wiki model to be worth the effort, the time needed to manage the model must be less than the time otherwise used to correct vandalism. The time spend managing the secure wiki model should be evaluated and compared to the time spend correcting vandalism in wiki systems suffering from various degrees of vandalism. This evaluation should show the characteristics of wiki systems where the secure wiki model can be used to reduce or eliminate vandalism.

8 Conclusion

This thesis have addressed issues with trustworthiness of content in wiki systems caused by the open nature of these systems. To this end, this thesis has described the secure wiki model, along with previous implementations of this model. The design-decisions made by the previous models have been analyzed and the knowledge gained has been used to propose an alternative policy for use with the secure wiki model. This thesis has also presented an analysis of a generic wiki system and an analysis of the secure wiki model in the context of this generic wiki system. Based on this, a proof-of-concept prototype has been designed and implemented.

The secure wiki model assigns integrity levels to users (QCVs) and articles (ILs)), and defines the procedures for managing these. The ILs for articles are managed by individual contributors when editing articles, while integrity levels for contributors are managed through promotion and demotion reviews. In this thesis, a new policy, Π_2, has been proposed. The new policy changes the promotion and demotion review-policy from a vote between levels to a weighted vote between users that makes the higher levels stronger than the lower levels, in terms of deciding the outcome of a review.

Where most wiki security schemes are reactive, the secure wiki model is proactive in that it limits contributors' ability to contribute to certain articles. Ensuring the integrity of articles in wiki systems by means of limiting the freedom of contributors to edit any article is in principle a conflict with the *all can edit* policy of wiki systems, but the practice of limiting contributors' ability to edit articles is already being used in existing wiki security schemes. The anti-vandalism protection measures employed by Wikipedia includes semi- and full-protection, which limits new or non-administrator contributors from editing protected articles. The approach to integrity that the secure wiki model has taken therefore does not conflict with a pragmatic view of the open nature of wiki systems. The implementation of the base wiki system and the secure wiki model plugin, using the new policy to evaluate reviews of contributors, have shown that the secure wiki model requires few changes to the user-interface of existing wikis.

The proposed policy Π_2 has been evaluated to determine appropriate values for the variables used in the definition of the policy and the work required to violate it. The evaluation of Π_2 has shown that the policy is highly configurable, giving the policy the ability to be configured to both strict and relaxed security requirements, without being vulnerable to attackers, allowing it to be tailored to most systems despite their varying needs. It has been shown that the new policy can achieve the same level of security with significantly less reviewers. This reduced number of reviewers is beneficial in motivating contributors to review, since they will be asked to do so less frequently than when using Π_1. This can help small systems with a low number

of contributors.

As outlined in Section 7, the implementation still requires the implementation of a number of wiki-related features, as well as the improvements of the other wiki-related features, before the system is on par with existing wiki systems. The lack of these features, however, does not affect the operation of the secure wiki model plugin, which does not rely on these. Although a number of features still needs to be implemented, the system does meet the definition of a wiki and can be set up and function as one.

References

[1] B. Thomas Adler, Luca de Alfaro, and Ian Pye. Detecting wikipedia vandalism using wikitrust, 2010.

[2] Luis Von Ahn, Manuel Blum, Nicholas J. Hopper, and John Langford. Captcha: using hard ai problems for security. In *Proceedings of the 22nd international conference on Theory and applications of cryptographic techniques*, EUROCRYPT'03, pages 294–311, Berlin, Heidelberg, 2003. Springer-Verlag.

[3] Tim Berners-Lee and Daniel Connolly. Hypertext markup language (html). http://tools.ietf.org/html/draft-ietf-iiir-html-00, 1993. [Online; accessed 21-February-2012].

[4] K. J. Biba. Integrity considerations for secure computer systems (technical report mtr-3153). *The Mitre Corporation*, 1977.

[5] Peter Denning, Jim Horning, David Parnas, and Lauren Weinstein. Wikipedia risks. *Commun. ACM*, 48:152–152, December 2005.

[6] John R. Douceur. The sybil attack. In *Revised Papers from the First International Workshop on Peer-to-Peer Systems*, IPTPS '01, pages 251–260, London, UK, 2002. Springer-Verlag.

[7] Eclipse.org. All eclipse foundation members. http://www.eclipse.org/membership/showAllMembers.php. [Online; accessed 23-February-2012].

[8] Michal Feldman, Christos Papadimitriou, John Chuang, and Ion Stoica. Free-riding and whitewashing in peer-to-peer systems. In *Proceedings of the ACM SIGCOMM workshop on Practice and theory of incentives in networked systems*, PINS '04, pages 228–236, New York, NY, USA, 2004. ACM.

[9] Christopher Følsgaard and Mark Ludwigs. Support for integrity module as plug-in in an existing wiki. Master's thesis, Technical University of Denmark, 2011.

[10] Google.com. recaptcha. http://www.google.com/recaptcha. [Online; accessed 23-February-2012].

[11] JamWiki. Jamwiki — jamwiki java wiki engine. http://jamwiki.org/wiki/en/Special:History?topicVersionId=18518&topic=JAMWiki, 2011. [Online; accessed 19-November-2011].

[12] C. Jensen. Security in wiki-style authoring systems. In *Proceedings of the Third IFIP International Conference on Trust Management (IFIPTM'09)*, pages 81–98, West Lafayette, Indiana, U.S.A., June 2009.

[13] M. Mihaila. Addressing the cold start problem in the wikipedia recommender system through content-based filtering. Master's thesis, Technical University of Denmark, DTU Informatics, 2011.

[14] osgi.org. OSGi Alliance. http://www.osgi.org/About/HomePage. [Online; accessed 26-December-2011].

[15] Poul Sander. Security in wiki-style systems. Master's thesis, Technical University of Denmark, DTU Informatics, 2009.

[16] Luis von Ahn, Manuel Blum, Nicholas Hopper, and John Langford. The official captcha site. http://www.captcha.net/. [Online; accessed 23-February-2012].

[17] Luis von Ahn, Manuel Blum, and John Langford. Telling humans and computers apart automatically. *Commun. ACM*, 47:56–60, February 2004.

[18] Wikipedia. Log in / create account — wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=Special:UserLogin&campaign=ACP2&type=signup&returnto=Main+Page. [Online; accessed 23-February-2012].

[19] Wikipedia. Wikipedia:editing frequency — wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/Wikipedia:Editing_frequency, 2008. [Online; accessed 23-February-2012].

[20] Wikipedia. Wikipedia:about — wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=Wikipedia:About&oldid=466855878, 2011. [Online; accessed 20-December-2011].

[21] Wikipedia. Wikipedia:bureaucrats — wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=Wikipedia:Bureaucrats&oldid=463277316, 2011. [Online; accessed 20-December-2011].

[22] Wikipedia. Wikipedia:wikiproject vandalism studies/study1 — wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=Wikipedia:WikiProject_Vandalism_studies/Study1&oldid=403633901, 2011. [Online; accessed 29-December-2011].

[23] Wikipedia. Category:wikipedia anti-vandal bots — wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/Category:Wikipedia_anti-vandal_bots, 2012. [Online; accessed 19-February-2012].

[24] Wikipedia. Wikipedia:patrols — wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/Wikipedia:Patrols, 2012. [Online; accessed 19-February-2012].

[25] Wikipedia. Wikipedia:user access levels — wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=Wikipedia:User_access_levels&oldid=474727213, 2012. [Online; accessed 19-February-2012].

Appendix

A Wikipedia author activity

The edit-frequencey statistics shown below are a reproduction of a list retrieved from Wikipedia itself [19]. The statistics where written in 2008 by the user *Dragons flight*[14], an administrator of the English Wikipedia since 2005[15]. In addition, the user claims to be the lead scientist, Robert Rohde, from Berkeley Earth Surface Temperature[16]. The claim seems to be valid and overall these facts lend credibility to the statistics.

The statistics below reproduce the last 13 entries from the statistics. Table 4 and 6 shows the number of authors with the more than the specified number of edits, respectively, per month and in total. Table 5 and 7 shows the corresponding percentages of active authors, respectively, per month and in total.

Table 4: Editors with $\geq$ X edits in that month

Year-Month	1	5	20	100	250	1000	2500
2008-9	132487	41393	13971	4127	1780	266	61
2008-8	130163	42610	15019	4391	1893	286	55
2008-7	130200	43298	15095	4392	1928	278	53
2008-6	131446	42547	14759	4260	1819	271	39
2008-5	146932	45835	15360	4449	1943	275	49
2008-4	150529	46782	15386	4520	1974	303	48
2008-3	152438	47820	15847	4750	2080	302	56
2008-2	145151	45346	14903	4411	1903	263	50
2008-1	144786	45404	15141	4566	2036	324	41
2007-12	132555	42123	14403	4207	1782	256	36
2007-11	143631	44667	14835	4339	1836	255	38
2007-10	151003	47362	15601	4563	1944	269	33
2007-9	141835	45284	15429	4490	1915	240	30

[14] http://en.wikipedia.org/wiki/User:Dragons_flight
[15] http://en.wikipedia.org/wiki/Wikipedia:Requests_for_adminship/Dragons_flight
[16] http://berkeleyearth.org/about-us/

Table 5: percentage of active editors with ≥ X edits in that month

Year-Month	1	5	20	100	250	1000	2500
2008-9	68,26	21,33	7,2	2,13	0,92	0,14	0,03
2008-8	66,95	21,92	7,73	2,26	0,97	0,15	0,03
2008-7	66,69	22,18	7,73	2,25	0,99	0,14	0,03
2008-6	67,36	21,8	7,56	2,18	0,93	0,14	0,02
2008-5	68,39	21,33	7,15	2,07	0,9	0,13	0,02
2008-4	68,57	21,31	7,01	2,06	0,9	0,14	0,02
2008-3	68,27	21,42	7,1	2,13	0,93	0,14	0,03
2008-2	68,46	21,39	7,03	2,08	0,9	0,12	0,02
2008-1	68,2	21,39	7,13	2,15	0,96	0,15	0,02
2007-12	67,85	21,56	7,37	2,15	0,91	0,13	0,02
2007-11	68,53	21,31	7,08	2,07	0,88	0,12	0,02
2007-10	68,4	21,45	7,07	2,07	0,88	0,12	0,01
2007-9	67,79	21,64	7,37	2,15	0,92	0,11	0,01

Table 6: Editors with ≥ X edits. Cumulative all-time

Year-Month	1	5	20	100	250	1000	2500	10000
2008-9	2021613	721512	242923	72119	38051	14245	6666	1727
2008-8	1969294	704787	237862	70821	37340	13944	6517	1674
2008-7	1920020	687910	232248	69294	36589	13644	6362	1617
2008-6	1871225	670554	226322	67677	35768	13325	6198	1561
2008-5	1821382	653433	220667	66215	35025	13051	6054	1503
2008-4	1761683	633571	214654	64604	34202	12738	5900	1438
2008-3	1699030	612626	208545	63012	33418	12418	5756	1379
2008-2	1634116	591340	202058	61315	32535	12093	5558	1323
2008-1	1572530	570938	196139	59795	31769	11771	5408	1267
2007-12	1511339	551325	190335	58164	30958	11432	5227	1208
2007-11	1459721	533975	184909	56673	30212	11138	5066	1157
2007-10	1400439	514534	179139	55132	29416	10836	4895	1101
2007-9	1335976	493363	173011	53507	28544	10493	4729	1051

Table 7: Percentage of active editors with ≥ X edits. Cumulative all-time

Year-Month	1	5	20	100	250	1000	2500	10000
2008-9	64,82	23,13	7,79	2,31	1,22	0,46	0,21	0,06
2008-8	64,73	23,17	7,82	2,33	1,23	0,46	0,21	0,06
2008-7	64,7	23,18	7,83	2,33	1,23	0,46	0,21	0,05
2008-6	64,69	23,18	7,82	2,34	1,24	0,46	0,21	0,05
2008-5	64,65	23,19	7,83	2,35	1,24	0,46	0,21	0,05
2008-4	64,56	23,22	7,87	2,37	1,25	0,47	0,22	0,05
2008-3	64,45	23,24	7,91	2,39	1,27	0,47	0,22	0,05
2008-2	64,33	23,28	7,95	2,41	1,28	0,48	0,22	0,05
2008-1	64,19	23,31	8,01	2,44	1,3	0,48	0,22	0,05
2007-12	64,04	23,36	8,07	2,46	1,31	0,48	0,22	0,05
2007-11	63,94	23,39	8,1	2,48	1,32	0,49	0,22	0,05
2007-10	63,79	23,44	8,16	2,51	1,34	0,49	0,22	0,05
2007-9	63,6	23,49	8,24	2,55	1,36	0,5	0,23	0,05

B Project Setup

This will describe how to set up the execution environment needed to run the implementation of the wiki system described in this thesis. The development of the project was made on linux, but can also run on Windows. Where there is a choice, 64 bit programs are the recommended version. The files referenced in this section is available from the cd containing the implementation.

B.1 Java

Ensure that java is available on the machine. From a terminal (Linux) or command-prompt (Windows), execute *java -version*. If the program *java* cannot be found, install it using the steps below. The output from the program, should look like

Java version output on Windows

```
1 java version "1.6.0_26"
2 Java(TM) SE Runtime Environment (build 1.6.0_26-b03)
3 Java HotSpot(TM) Client VM (build 20.1-b02, mixed mode, sharing)
```

Java version output on Linux

```
1 java version "1.6.0_22"
2 OpenJDK Runtime Environment (IcedTea6 1.10.6) (6b22-1.10.6-0ubuntu1)
3 OpenJDK Client VM (build 20.0-b11, mixed mode, sharing)
```

If the java-version is below 1.6, java should be updated.

B.1.1 Ubuntu Linux

Using Ubuntu, openjdk-6 is installed with the operating system. If this has been removed, or for some reason is not present, install it using

```
1 sudo apt-get install openjdk-6-jre
```

Alternatively, to install the Sun/Oracle java 6, download it from http://java.com/en/download/linux_manual.jsp?locale=en and follow the instructions to install it. Issue *java -version* again to verify that java has been installed correctly.

B.1.2 Windows

To install java on Windows, download it from http://java.com/en/download/windows_manual.jsp?locale=en and follow the instructions to install it. Issue *java -version* again to verify that java has been installed correctly.

B.2 MySQL

To run the wiki system, a MySQL server must be set up.

B.2.1 Ubuntu Linux

Install the MySQL database server using

```
1 sudo apt-get install mysql-server
```

B.2.2 Windows

Download the MySQL server from http://dev.mysql.com/downloads/mysql/ and install it.

B.2.3 Setup

To create the database used for the base system and the secure wiki model plugin execute the following command (db.sql is the file in Listing 7. It is also included on the CD).

```
1 mysql -u root -p < db.sql
```

When prompted for the root password, enter it. To create the database user the wiki system will use to connect, log in to the database server using:

```
1 mysql -u root -p
```

Then create a dedicated user with full access to the created database using the command:

```
1 grant all on wiki.* to '<user>'@'\%' identified by '<password>';
```

Remember the values chosen for '<user>' and '<password>'.

B.3 Apache Tomcat

Apache Tomcat 6 is the application server chosen to run the wiki web application.

B.3.1 Ubuntu Linux

On ubuntu, The tomcat server can be installed using

```
sudo apt-get install tomcat6
```

The lib-folder will be /usr/share/tomcat6/lib/, the config-folder will be /etc/tomcat6/, the context-folder will be /etc/tomcat6/Catalina/localhost/ and the web-application directory will be /var/lib/tomcat6/webapps/.

B.3.2 Windows

Tomcat 6 for windows is available at http://tomcat.apache.org/download-60.cgi. Download the Windows Service installer and follow instructions to install. Assuming standard installation path, Relative to the base installation directory, usually C:\Program Files\Apache Software Foundation\Tomcat 6.0, the lib-folder will be \lib, the config-folder will be \conf, the context-folder will be \conf\Catalina\localhost and the web-application directory will be \webapps.

B.3.3 Setup

To connect to a MySQL database, the tomcat installation must be extended with a MySQL-driver. To do this, add mysql-connector-java-5.1.6.jar to the lib-folder. Create a file with the content shown in Listing 2 (it can also be copied from the cd). Change the username and password to match the credentials of the database user created above.

Listing 2: wiki.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<Context>
    <Resource name="jdbc/wiki" auth="Container" type="javax.sql.DataSource"
        username="<user>" password="<password>"
        driverClassName="com.mysql.jdbc.Driver"
        url="jdbc:mysql://localhost:3306/wiki"
        maxActive="15" maxIdle="3"/>
</Context>
```

Add the file to the context-folder of the tomcat installation. The file should be named *wiki.xml*.

B.4 Application Installation

To install the wiki web application, copy the the web-application *wiki.war* to the tomcat web-application directory, and start the tomcat server, if it is not running already. Wait for the tomcat to start itself and the wiki web application, then access the server on port 8080, using /wiki as the context path, e.g. http://localhost:8080/wiki.

C Patched Http Bridge Bundle

The following description assumes knowledge of maven, maven projects and their structure as well as access to a repository server, e.g. http://archiva.apache.org/, where the bundle will be deployed to. To create the apache felix http bridge-bundle used for the base wiki system, a

maven project should be created using the standard maven project structure, with source-files in src/main/java. The sources are located in multiple projects and should be fetched from the following sources:

- http://mirrors.dotsrc.org/apache/felix/org.apache.felix.http.api-2.2.0-project.tar.gz
- http://mirrors.dotsrc.org/apache/felix/org.apache.felix.http.base-2.2.0-project.tar.gz
- http://mirrors.dotsrc.org/apache/felix/org.apache.felix.http.bridge-2.2.0-project.tar.gz
- http://repo1.maven.org/maven2/org/osgi/org.osgi.compendium/4.2.0/org.osgi.compendium-4.2.0-sources.jar

The sources should be merged such that the following package structure is obtained.

```
org
org/apache
org/apache/felix
org/apache/felix/http
org/apache/felix/http/api
org/apache/felix/http/base
org/apache/felix/http/base/internal
org/apache/felix/http/base/internal/context
org/apache/felix/http/base/internal/dispatch
org/apache/felix/http/base/internal/handler
org/apache/felix/http/base/internal/listener
org/apache/felix/http/base/internal/logger
org/apache/felix/http/base/internal/service
org/apache/felix/http/base/internal/util
org/apache/felix/http/bridge
org/apache/felix/http/bridge/internal
org/osgi
org/osgi/service
org/osgi/service/http
```

The patch shown in Listing 3 should then be applied to the sources.

Listing 3: org.apache.felix.http.bridge forward patch

```
Index: src/main/java/org/apache/felix/http/base/internal/dispatch/ServletPipeline.java
===================================================================

--- src/main/java/org/apache/felix/http/base/internal/dispatch/ServletPipeline.java (revision 60)
+++ src/main/java/org/apache/felix/http/base/internal/dispatch/ServletPipeline.java (working copy)
@@ -88,7 +88,7 @@
         public void include(ServletRequest req, ServletResponse res)
             throws ServletException, IOException
         {
-            this.handler.handle((HttpServletRequest)req, (HttpServletResponse)res);
+            this.handler.handle(new RequestWrapper((HttpServletRequest)req, this.path), (HttpServletResponse)res);
         }
     }
```

```
13
14 @@ -103,6 +103,12 @@
15                 this.requestUri = requestUri;
16         }
17
18 + @Override
19 + public String getPathInfo() {
20 +  return getRequestURI();
21 + }
22 +
23 + @Override
24         public String getRequestURI()
25         {
26             return this.requestUri;
```

To build the bundle, add the following pom-file to the root of the project. The Distribution-Management section of the pom-file should be modified to match the repository server that will hold the snapshot of the bundle.

Listing 4: org.apache.felix.http.bridge pom.xml

```
1  <!--
2      Licensed to the Apache Software Foundation (ASF) under one
3      or more contributor license agreements. See the NOTICE file
4      distributed with this work for additional information
5      regarding copyright ownership. The ASF licenses this file
6      to you under the Apache License, Version 2.0 (the
7      "License"); you may not use this file except in compliance
8      with the License. You may obtain a copy of the License at
9
10     http://www.apache.org/licenses/LICENSE-2.0
11
12     Unless required by applicable law or agreed to in writing,
13     software distributed under the License is distributed on an
14     "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY
15     KIND, either express or implied. See the License for the
16     specific language governing permissions and limitations
17     under the License.
18 -->
19 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/ ↩
       XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven. ↩
       apache.org/maven-v4_0_0.xsd">
20
21     <modelVersion>4.0.0</modelVersion>
22     <parent>
23         <groupId>org.apache.felix</groupId>
24         <artifactId>org.apache.felix.http</artifactId>
25         <version>2.2.0</version>
26     </parent>
27
28     <name>Patched Apache Felix Http Bridge</name>
29     <artifactId>org.apache.felix.http.bridge</artifactId>
30     <version>${org.apache.felix.http.bridge.version}.patched-SNAPSHOT</version>
31     <packaging>bundle</packaging>
32
33     <properties>
```

```
34          <org.apache.felix.http.bridge.version>2.2.0</org.apache.felix.http.bridge.version>
35      </properties>
36      <build>
37      <resources>
38              <resource>
39                  <directory>${basedir}/src/main/resources</directory>
40              </resource>
41              <resource>
42                  <targetPath>META-INF</targetPath>
43                  <directory>${basedir}</directory>
44                  <includes>
45                      <include>LICENSE</include>
46                      <include>NOTICE</include>
47                      <include>DEPENDENCIES</include>
48                  </includes>
49              </resource>
50          </resources>
51          <plugins>
52              <plugin>
53                  <groupId>org.apache.maven.plugins</groupId>
54                  <artifactId>maven-compiler-plugin</artifactId>
55                  <version>2.3.2</version>
56                  <configuration>
57                      <source>1.6</source>
58                      <target>1.6</target>
59                      <fork>true</fork>
60                      <executable>/usr/lib/jvm/java-6-sun/bin/javac</executable>
61                      <compilerVersion>1.6</compilerVersion>
62                  </configuration>
63              </plugin>
64              <plugin>
65                  <groupId>org.apache.maven.plugins</groupId>
66                  <artifactId>maven-source-plugin</artifactId>
67                  <version>2.1.2</version>
68                  <executions>
69                      <execution>
70                          <id>attach-sources</id>
71                          <phase>verify</phase>
72                          <goals>
73                              <goal>jar-no-fork</goal>
74                          </goals>
75                      </execution>
76                  </executions>
77              </plugin>
78              <plugin>
79                  <groupId>org.apache.felix</groupId>
80                  <artifactId>maven-bundle-plugin</artifactId>
81                  <extensions>true</extensions>
82                  <version>2.0.0</version>
83                  <configuration>
84                      <instructions>
85                          <Bundle-Activator>
86                              org.apache.felix.http.bridge.internal.BridgeActivator
87                          </Bundle-Activator>
88                          <Export-Package>
89                              org.apache.felix.http.api;version=2.2.0,
```

```
                    org.osgi.service.http;version=1.2.0
                </Export-Package>
                <Private-Package>
                    org.apache.felix.http.base.*,
                    org.apache.felix.http.bridge.internal.*
                </Private-Package>
                <Import-Package>
                    javax.servlet.*,
                    *;resolution:=optional
                </Import-Package>
            </instructions>
        </configuration>
    </plugin>
  </plugins>
</build>

<dependencies>
  <dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>servlet-api</artifactId>
    <scope>provided</scope>
  </dependency>
  <dependency>
    <groupId>org.osgi</groupId>
    <artifactId>org.osgi.core</artifactId>
    <scope>provided</scope>
  </dependency>
  <dependency>
    <groupId>org.osgi</groupId>
    <artifactId>org.osgi.compendium</artifactId>
    <scope>provided</scope>
  </dependency>
</dependencies>
<!-- replace example.com below with the appropriate url-->
<distributionManagement>
  <repository>
    <id>internal</id>
    <name>Archiva Managed Internal Repository</name>
    <url>http://example.com:8080/archiva/repository/internal</url>
  </repository>
  <snapshotRepository>
    <id>snapshots</id>
    <name>Archiva Managed Internal Snapshot Repository</name>
    <url>http://example.com:8080/archiva/repository/snapshots</url>
  </snapshotRepository>
</distributionManagement>
</project>
```

Build and deploy the bundle using maven

Maven deploy command

```
mvn clean deploy
```

If no repository server is available, use *install* instead of *deploy*. The bundle will then have to be installed in the local repository on every computer used for development.

D Equinox JSP HTTP-Helper bundle

The following description assumes knowledge of maven, maven projects and their structure as well as access to a repository server, e.g. http://archiva.apache.org/, where the bundle will be deployed to. To create the Equinox JSP HTTP-Helper bundle used for the base wiki system, a maven project should be created using the standard maven project structure, with source-files in src/main/java. The sources are located in multiple projects and should be fetched from the following sources:

To build the JSP HTTP-Helper bundle, check out revision *v20070607* of *org.eclipse.equinox/server-side/bundles/org.eclipse.equinox.jsp.jasper* from *:pserver:anonymous@dev.eclipse.org:/cvsroot/rt*

The source were fetched using the following cvs command from a linux terminal.

Listing 5: CVS checkout command for JSP HTTP-Helper bundle sourcecode

```
cvs -Q -d :pserver:anonymous@dev.eclipse.org:/cvsroot/rt co -r v20070607 \
  org.eclipse.equinox/server-side/bundles/org.eclipse.equinox.jsp.jasper
```

To build the bundle, add the following pom-file to the root of the project. The Distribution-Management section of the pom-file should be modified to match the repository server that will hold the snapshot of the bundle.

Listing 6: org.eclipse.equinox.http.helper pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>org.eclipse.equinox.http</groupId>
    <artifactId>helper</artifactId>
    <description>Bundle for org.eclipse.equinox.http.helper ,created using maven, from sources found at :pserver:anonymous@dev.eclipse.org:/cvsroot/eclipse /equinox-incubator/org.eclipse.equinox.http.helper </description>
    <packaging>bundle</packaging>
    <name>org.eclipse.equinox.http.helper</name>
    <version>0.0.2-SNAPSHOT</version>
    <dependencies>
        <dependency>
            <groupId>javax.servlet</groupId>
            <artifactId>servlet-api</artifactId>
            <version>2.5</version>
        </dependency>
        <dependency>
            <groupId>org.osgi</groupId>
            <artifactId>org.osgi.compendium</artifactId>
            <version>4.2.0</version>
        </dependency>
        <dependency>
            <groupId>org.osgi</groupId>
            <artifactId>org.osgi.core</artifactId>
            <version>4.2.0</version>
        </dependency>
    </dependencies>
    <build>
```

```
28          <plugins>
29              <plugin>
30                  <groupId>org.apache.maven.plugins</groupId>
31                  <artifactId>maven-compiler-plugin</artifactId>
32                  <version>2.3.2</version>
33                  <configuration>
34                      <source>1.6</source>
35                      <target>1.6</target>
36                      <fork>true</fork>
37                      <executable>/usr/lib/jvm/java-6-sun/bin/javac</executable>
38                      <compilerVersion>1.6</compilerVersion>
39                  </configuration>
40              </plugin>
41              <plugin>
42                  <groupId>org.apache.maven.plugins</groupId>
43                  <artifactId>maven-source-plugin</artifactId>
44                  <version>2.1.2</version>
45                  <executions>
46                      <execution>
47                          <id>attach-sources</id>
48                          <phase>verify</phase>
49                          <goals>
50                              <goal>jar-no-fork</goal>
51                          </goals>
52                      </execution>
53                  </executions>
54              </plugin>
55              <plugin>
56                  <groupId>org.apache.felix</groupId>
57                  <artifactId>maven-bundle-plugin</artifactId>
58                  <extensions>true</extensions>
59                  <version>2.0.0</version>
60                  <configuration>
61                      <instructions>
62                          <Bundle-Name>Http Service Helper Bundle</Bundle-Name>
63                          <Bundle-SymbolicName>org.eclipse.equinox.http.helper</Bundle- ↩
                                SymbolicName>
64                          <Bundle-Description>${project.description}</Bundle-Description>
65                          <Bundle-Version>${project.version}</Bundle-Version>
66                          <Bundle-Vendor>Eclipse.org</Bundle-Vendor>
67                          <Bundle-Activator>org.eclipse.equinox.http.helper.Activator</Bundle- ↩
                                Activator>
68                          <Bundle-Copyright>Eclipse.org</Bundle-Copyright>
69                          <Import-Package>javax.servlet;version="2.5",javax.servlet.http;version="2.5", ↩
                                org.osgi.framework;version="1.3.0",org.osgi.service.http;version="1.2.0"</ ↩
                                Import-Package>
70                          <Export-Package>org.eclipse.equinox.http.helper</Export-Package>
71                      </instructions>
72                  </configuration>
73              </plugin>
74          </plugins>
75      </build>
76      <distributionManagement>
77          <!-- replace example.com below with the appropriate url-->
78          <repository>
79              <id>internal</id>
```

```
            <name>Archiva Managed Internal Repository</name>
            <url>http://example.com:8080/archiva/repository/internal</url>
        </repository>
        <snapshotRepository>
            <id>snapshots</id>
            <name>Archiva Managed Internal Snapshot Repository</name>
            <url>http://example.com:8080/archiva/repository/snapshots</url>
        </snapshotRepository>
    </distributionManagement>
</project>
```

Build and deploy the bundle using maven

Maven deploy command

```
mvn clean deploy
```

If no repository server is available, use *install* instead of *deploy*. The bundle will then have to be installed in the local repository on every computer used for development.

E Database SQL

The following script creates the database as described in the design section. The script creates tables for both the base system and the secure wiki model plugin.

Listing 7: Database creation SQL

```
-- MySQL dump 10.13 Distrib 5.1.58, for debian-linux-gnu (x86_64)
--
-- Host: localhost Database: wiki
-- ------------------------------------------------------
-- Server version 5.1.58-1ubuntu1

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8 */;
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, ↩
    FOREIGN_KEY_CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE=' ↩
    NO_AUTO_VALUE_ON_ZERO' */;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;

--
-- Current Database: `wiki`
--

/*!40000 DROP DATABASE IF EXISTS `wiki`*/;

CREATE DATABASE /*!32312 IF NOT EXISTS*/ `wiki` /*!40100 DEFAULT CHARACTER SET utf8 ↩
    COLLATE utf8_bin */;

```

```
26 USE ‘wiki‘;
27
28 --
29 -- Temporary table structure for view ‘articles‘
30 --
31
32 DROP TABLE IF EXISTS ‘articles‘;
33 /*!50001 DROP VIEW IF EXISTS ‘articles‘*/;
34 SET @saved_cs_client = @@character_set_client;
35 SET character_set_client = utf8;
36 /*!50001 CREATE TABLE ‘articles‘ (
37   ‘aid‘ int(11),
38   ‘markup‘ text,
39   ‘url‘ varchar(255),
40   ‘modified‘ timestamp
41 ) ENGINE=MyISAM */;
42 SET character_set_client = @saved_cs_client;
43
44 --
45 -- Table structure for table ‘reviews‘
46 --
47
48 DROP TABLE IF EXISTS ‘reviews‘;
49 /*!40101 SET @saved_cs_client = @@character_set_client */;
50 /*!40101 SET character_set_client = utf8 */;
51 CREATE TABLE ‘reviews‘ (
52   ‘rid‘ int(11) NOT NULL AUTO_INCREMENT,
53   ‘uid‘ int(11) NOT NULL,
54   ‘description‘ text COLLATE utf8_bin NOT NULL,
55   ‘type‘ int(11) NOT NULL,
56   ‘targetqcv‘ int(11) NOT NULL,
57   ‘requestorid‘ int(11) NOT NULL,
58   ‘start‘ timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
59   ‘end‘ timestamp NOT NULL DEFAULT ’0000-00-00 00:00:00’,
60   ‘processed‘ int(11) NOT NULL DEFAULT ’0’,
61   PRIMARY KEY (‘rid‘),
62   KEY ‘uid‘ (‘uid‘),
63   KEY ‘requestorid‘ (‘requestorid‘)
64 ) ENGINE=MyISAM AUTO_INCREMENT=1 DEFAULT CHARSET=utf8 COLLATE=utf8_bin;
65 /*!40101 SET character_set_client = @saved_cs_client */;
66
67 --
68 -- Table structure for table ‘revisions‘
69 --
70
71 DROP TABLE IF EXISTS ‘revisions‘;
72 /*!40101 SET @saved_cs_client = @@character_set_client */;
73 /*!40101 SET character_set_client = utf8 */;
74 CREATE TABLE ‘revisions‘ (
75   ‘rid‘ int(11) NOT NULL AUTO_INCREMENT,
76   ‘markup‘ text NOT NULL,
77   ‘url‘ varchar(255) NOT NULL,
78   ‘uid‘ int(11) NOT NULL,
79   ‘minor_edit‘ tinyint(3) NOT NULL DEFAULT ’0’,
80   ‘edit_summary‘ varchar(256) DEFAULT NULL,
81   ‘modified‘ timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE ↩
```

```
        CURRENT_TIMESTAMP,
  PRIMARY KEY (`rid`),
  KEY `url` (`url`) USING HASH,
  KEY `users` (`uid`),
  FULLTEXT KEY `article` (`markup`,`url`)
) ENGINE=MyISAM AUTO_INCREMENT=1 DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `swmp_revision`
--

DROP TABLE IF EXISTS `swmp_revision`;
/*!40101 SET @saved_cs_client = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `swmp_revision` (
  `rid` int(11) NOT NULL,
  `il` int(11) NOT NULL,
  UNIQUE KEY `docid` (`rid`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_bin;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `swmp_user`
--

DROP TABLE IF EXISTS `swmp_user`;
/*!40101 SET @saved_cs_client = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `swmp_user` (
  `uid` int(11) NOT NULL,
  `doclevel` int(11) NOT NULL DEFAULT '0',
  `userlevel` int(11) NOT NULL DEFAULT '0',
  UNIQUE KEY `userid` (`uid`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_bin;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `users`
--

DROP TABLE IF EXISTS `users`;
/*!40101 SET @saved_cs_client = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `users` (
  `uid` int(11) NOT NULL AUTO_INCREMENT,
  `username` varchar(48) NOT NULL,
  `password` varchar(128) NOT NULL DEFAULT '',
  `created` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
  PRIMARY KEY (`uid`),
  UNIQUE KEY `username` (`username`)
) ENGINE=MyISAM AUTO_INCREMENT=1 DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `votes`
```

```
137 --
138
139 DROP TABLE IF EXISTS `votes`;
140 /*!40101 SET @saved_cs_client = @@character_set_client */;
141 /*!40101 SET character_set_client = utf8 */;
142 CREATE TABLE `votes` (
143   `vid` int(11) NOT NULL AUTO_INCREMENT,
144   `rid` int(11) NOT NULL,
145   `uid` int(11) NOT NULL,
146   `vote` int(11) DEFAULT NULL,
147   PRIMARY KEY (`vid`),
148   UNIQUE KEY `voteid` (`rid`,`uid`),
149   KEY `uid` (`uid`)
150 ) ENGINE=MyISAM AUTO_INCREMENT=1 DEFAULT CHARSET=utf8 COLLATE=utf8_bin;
151 /*!40101 SET character_set_client = @saved_cs_client */;
152
153 --
154 -- Current Database: `wiki`
155 --
156
157 USE `wiki`;
158
159 --
160 -- Final view structure for view `articles`
161 --
162
163 /*!50001 DROP TABLE IF EXISTS `articles`*/;
164 /*!50001 DROP VIEW IF EXISTS `articles`*/;
165 /*!50001 SET @saved_cs_client = @@character_set_client */;
166 /*!50001 SET @saved_cs_results = @@character_set_results */;
167 /*!50001 SET @saved_col_connection = @@collation_connection */;
168 /*!50001 SET character_set_client = latin1 */;
169 /*!50001 SET character_set_results = latin1 */;
170 /*!50001 SET collation_connection = latin1_swedish_ci */;
171 /*!50001 CREATE ALGORITHM=UNDEFINED */
172 /*!50013 DEFINER=`wiki`@`%` SQL SECURITY INVOKER */
173 /*!50001 VIEW `articles` AS select `a`.`rid` AS `aid`,`a`.`markup` AS `markup`,`a`.`url` AS `url`,`a`.`modified` ↩
      ` AS `modified` from `revisions` `a` where (`a`.`rid` = (select max(`b`.`rid`) from `revisions` `b` where (`a` ↩
      `.`url` = `b`.`url`))) */;
174 /*!50001 SET character_set_client = @saved_cs_client */;
175 /*!50001 SET character_set_results = @saved_cs_results */;
176 /*!50001 SET collation_connection = @saved_col_connection */;
177 /*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;
178
179 /*!40101 SET SQL_MODE=@OLD_SQL_MODE */;
180 /*!40014 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS */;
181 /*!40014 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS */;
182 /*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
183 /*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
184 /*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
185 /*!40111 SET SQL_NOTES=@OLD_SQL_NOTES */;
186
187 -- Dump completed on 2012-03-02 0:41:28
```