PROTASIS

Restoring Trust in the Cyber Space:
A Systems Security Proposal

D2.5: CyberSecurity Repository†

Abstract: This report summarizes the effort put by the PROTASIS project in the Cybersecurity Repository: a platform to allow simplified access to research on cybersecurity.

Contractual Date of Upload
Actual Date of Upload
Deliverable Security Class
Editor
Contributors
Quality Assurance

April 2017
October 2017
Public
Davide Quarta
All PROTASIS partners
Stefano Zanero, H. Bos
The **PROTASIS** consortium consists of:

<table>
<thead>
<tr>
<th>Organization</th>
<th>Role</th>
<th>Country</th>
</tr>
</thead>
<tbody>
<tr>
<td>FORTH</td>
<td>Coordinator</td>
<td>Greece</td>
</tr>
<tr>
<td>POLIMI</td>
<td>Beneficiary</td>
<td>Italy</td>
</tr>
<tr>
<td>VU</td>
<td>Beneficiary</td>
<td>The Netherlands</td>
</tr>
<tr>
<td>RUB</td>
<td>Beneficiary</td>
<td>Germany</td>
</tr>
<tr>
<td>TELEFONICA</td>
<td>Beneficiary</td>
<td>Spain</td>
</tr>
<tr>
<td>F-Secure</td>
<td>Beneficiary</td>
<td>Finland</td>
</tr>
<tr>
<td>MIT</td>
<td>TC Partner</td>
<td>USA</td>
</tr>
<tr>
<td>UCSB</td>
<td>TC Partner</td>
<td>USA</td>
</tr>
<tr>
<td>UIC</td>
<td>TC Partner</td>
<td>USA</td>
</tr>
<tr>
<td>Columbia University</td>
<td>TC Partner</td>
<td>USA</td>
</tr>
<tr>
<td>Northeastern University</td>
<td>TC Partner</td>
<td>USA</td>
</tr>
<tr>
<td>Stony Brook University</td>
<td>TC Partner</td>
<td>USA</td>
</tr>
<tr>
<td>Stevens</td>
<td>TC Partner</td>
<td>USA</td>
</tr>
<tr>
<td>Boston University</td>
<td>TC Partner</td>
<td>USA</td>
</tr>
</tbody>
</table>
# Table of Contents

1 Table of Contents .................................................................................................................. 3
1 Introduction .................................................................................................................................. 4
2 Cybersecurity repository ........................................................................................................... 5
  2.1 Authentication and browsing ............................................................................................... 5
  2.2 Projects .................................................................................................................................. 5
  2.3 Publications ........................................................................................................................... 6
  2.4 Files ....................................................................................................................................... 7
  2.5 Data & Code .......................................................................................................................... 7
  2.6 Wiki ........................................................................................................................................ 7
  2.7 User messages ....................................................................................................................... 9
3 Technical details ......................................................................................................................... 11
  3.1 Deployment environment ....................................................................................................... 11
  3.2 Security considerations ......................................................................................................... 11
  3.3 Authentication framework ..................................................................................................... 11
    3.3.1 Views ............................................................................................................................... 12
4 Feeding the repository ............................................................................................................... 13
  4.1 Manually ............................................................................................................................... 13
  4.2 Automatically ........................................................................................................................ 13
5 Future developments ................................................................................................................ 17
1 Introduction

This report describes the “Cybersecurity Repository” platform prototype, built by the PROTASIS consortium as D2.5.

This platform will allow to keep track of research outcomes of the project (and, if adopted, of other European cybersecurity research projects), and act as a receptacle for all the related informations (including data, artifacts, code...).

The source will be available at https://github.com/Protasis/cs_repository.

Figure 1: The repository homepage (not logged in)
2 Cybersecurity repository

2.1 Authentication and browsing
Once logged in, the user will see all the resources they have access to. If the user is part of the staff group, he will be also able to modify the resources via the administration interface.

Figure 2: The home page of the repository when logged in

2.2 Projects
A project is a “unit” to keep related material together. Papers, code and data can be added. Moreover, a link to the internal wiki is generated to allow discussion and a free space to keep less structured material (i.e., teaching material).
2.3 Publications

The following items can be added as a publication:

- papers
- deliverables
- reports

More item types can easily be added in future versions of the application code.

In Figure 4 a paper is shown: the ID for vulnerability advisories can also be added to a publication where available. Citations in bibtex format can be added to the project to allow easily exporting citations.
2.4 Files

When an entry needs a particular file associated (i.e., the pdf version of a publication), it can be stored in a file object, or an external url can be specified. This is handled transparently and helps avoiding duplication of documents and using permanent repository links where already available.

2.5 Data & Code

Publications and projects can also have several associated data and code packages. These are stored as multiple entries each one with their own set of access permissions to allow for fine grained access management.

2.6 Wiki

Projects can also have a related wiki in which material and discussions can be uploaded. The url for the related wiki is automatically generated for each entry, but the wiki entry itself is manually generated. We plan to release a plugin for the Django wiki software as a helpful additional tool in a next release of the code.
Figure 5: The wiki for the SysSec project with an entry

Articles are written in markdown (https://daringfireball.net/projects/markdown/) a simple plaintext formatting syntax that will be automatically (and securely) converted to HTML.
2.7 User messages

To foster communication and collaboration, the repository also provides a way for users of the platform to communicate via private messages.
Figure 7: User messages interface
3 Technical details

In this section we will describe some technical details of the platform. Since the code of the project is written in a high-level language with a widely used web framework, the code is pretty self-explanatory. We will limit the description to critical design choices.

3.1 Deployment environment

The repository is based on Python 2.7, Django 1.10 with several extensions, and the objects are stored using MySQL.

The characteristics of Django that make it desirable to use as web framework to base the repository prototype on are the following:

- widely used, active community
- can be used to prototype a web service quickly
- many plugins and modules exists to expand the basic functionalities included in the framework

The repository has been developed using virtual environments and pip, a python modules repository (a requirements.txt is present in the git repository to ease deployment).

Moreover different linters (tools to ensure the “cleanliness” of the code) have been used during the development process, ensuring the code responds to the python community guidelines.

3.2 Security considerations

Given that some of the material included in the repository could be sensitive, some considerations about the security of the repository must be made.

The django framework is a mature and relatively solid project (i.e., last year only a few vulnerabilities were discovered: http://www.cvedetails.com/vulnerability-list/vendor_id-10199/product_id-18211/Djangoproject-Django.html)

Nevertheless also the django extensions we used could contain vulnerabilities. Hopefully this can be mitigated with a proactive approach to security. Since all the modules we are using are published on github, the status of the security of these projects will be possible to track when the security alerts feature will be rolled out by github. (see http://www.zdnet.com/article/github-aims-to-make-coding-more-automated/).

3.3 Authentication framework

The out-of-the-box functionality offered by django is not flexible enough to handle fine grained permissions for objects. We developed a “framework” to allow easily adding new “authenticable” objects. Defining a authenticable models is done by adding the AuthMixin as a mixin for the desired model.
All the *accessible* objects linked to a model are accessible using the “*get_accessible*” class method, which will return all the accessible objects, and if the parameter “unlinked” is True it will return all objects that are *spurious* (i.e., they are not linked with any project/publication).

The “*accessible_rel*” method will help getting all the *accessible* authenticable objects linked with the current model (i.e., publications, data, code...).

### 3.3.1 Views

Access to the views is regulated via the “*check_group_access*” and “*check_data_access*” decorators, that will check for permission on the decorated views and will pass the object to the decorated view or raise a 404 if it is not possible to “fetch” the object.
4 Feeding the repository

Feeding the repository can happen in two different ways, manually and automatically.

4.1 Manually

Any object can be added via the administration interface.

![Repository administration interface](image)

Figure 8: Repository administration interface

4.2 Automatically

To automatically add objects, a custom django management command can be created, this is needed because of how django handles access to the application models.

To programmatically create content we can for example scrape a web resource: to show this potential, we developed an example that scrapes the deliverables of the SysSec project from its website publications page. The code implementing this example can be found inside the repository in the file: `collabtool/management/commands/scrape_deliverables.py`.
Figure 9: Scraping with beautifulsoup and adding the deliverable

The scraped object will be saved and will be accessible to any user having read permissions for the syssec group.
At this point the publications haven’t been added to the project so they will result as unsorted publications available to the user: we need to create then a Publication object (that is a generic container for publications), link it with the deliverable, and then add the Publication model to the project.

This is the only part that requires some more glue logic. This code example will be included in the git repository to ensure that the process is as clearly and easily understandable as possible.

```
# now add to the syssec project
content_type = ContentType.objects.get(
    app_label='collabtool', model='Deliverable')
project = Project.objects.get(title__exact='syssec')
publication = Publication(
    content_type=content_type, object_id=d.id)
publication.save()
project.publication.add(publication)
project.save()
```

The final results can be seen in Figure 12.
Figure 12: The scraped deliverables in the SysSec project
5 Future developments

The software is going to be evolved further throughout the PROTASIS project. In particular, in the next release we will implement the following features – for which the code and the data models are already planned in the current version of the project:

- staff access management can be further restricted by extending django-admin to employ the very same authentication framework we developed. This simple modification will make possible to grant write access with an extremely fine-grained granularity.
- automatically parse the PublicationBase model to export a bibtex file if none has been uploaded manually
- parse automatically bibtex (i.e. use the bibtexparse python module) to automatically parse and insert a publication from a bibtex file
- parse advisory IDs and link to the advisory (i.e., CVE->[NVD...], MS->Microsoft, ABB→ABB)

Other desirable features that we are evaluating to add in the future include:

- SSO support
- user comments
- statistics for projects/papers/authors
- REST APIs
- export feeds (xml...)