Pillaging DVCS Repos

Adam Baldwin

INTRODUCTION

Distributed Version Control Systems (DVCS) including Git, Mercurial (HG), and Bazaar (BZR) are becoming increasingly popular and also a convenient method of deploying updates to web applications. DVCS track revision history and other information about the repository inside a meta directory. In some cases the meta directory is left inside the web root of the website unprotected. This information could be very valuable to an attacker or as part of a penetration test.

OBJECTIVE

The goal of this research was to identify methods in which information could be remotely extracted from the exposed meta directory and recreate as much of the repository state as possible.

REPOSITORY IDENTIFICATION

The first step in our process was to remotely and reliably identify the presence of an exposed DVCS meta directory. It was found that each DVCS we tried had a predictable set of file names, paths and contents. Using this information we created a set of regular expressions to identify a positive match.

DVCS	File	Regular Expression
GIT	.git/HEAD	^ref: refs/'
HG	.hg/requires	^revlogv1'
BZR	.bzr/README	^This\sis\sa\sBazaar'

A discovery plugin called findDVCS.py was created for the W3AF (Web Application Attack and Audit Framework) framework. This plugin uses the above regular expressions for repository identification.

PREDICTABLE FILES

The next step in extracting repository data is to download files with predictable file locations. Some of these are critical in extracting useful information from the repository. Below is a list of some of the predictable files. Some are left out just because we don't care about them.

GIT

File	Notes
.git/HEAD	Used to identify the repository and repository ref's
.git/config	Lists branches and remotes that may provide other targets to attack.
.git/index	"sorted list of path names, each with permissions and the SHA-1 of a blob object" [0]
.git/logs/HEAD	Current HEAD reference
.git/hooks/*	applypatch-msg.sample,pre-applypatch.sample,commit-msg.sample,pre-commit.sample,post-commit.sample,pre-rebase.sample,post-receive.sample,preparecommit-msg.sample,post-update.sample,update.sample Some hooks may contain sensitive information, depending on how they are used. Just remove .sample and see if they exist.

HG

File
.hg/00changelog.i
.hg/dirstate
.hg/requires
.hg/branch
.hg/branchheads.cache
.hg/last-message.txt
.hg/tags.cache
.hg/undo.branch
.hg/undo.desc
.hg/undo.dirstate

File
.hg/store/00changelog.i
.hg/store/00changelog.d
.hg/store/00manifest.i
.hg/store/00manifest.d.hg/store/fncache
.hg/store/undo

BZR

File
.bzr/branch-format
.bzr/branch/branch.conf
.bzr/branch/format
.bzr/branch/last-revision
.bzr/branch/tags
.bzr/checkout/conflicts
.bzr/checkout/dirstate
.bzr/checkout/format
.bzr/checkout/merge-hashes
.bzr/checkout/views
.bzr/repository/format
.bzr/repository/pack-names

RECONSTRUCTING THE REPOSITORY

Once we have obtained the predictable files inside of the repositories we can start trying to reconstruct as much of the repository as possible. 100% recovery of the repository is not always possible. In those cases there is still information that can be obtained and will be detailed in the next section.

GIT

1. Get the ref from .git/HEAD

2. Get the object that the ref points to. These objects are stored in the .git/objects directory in the form of a sha1 hash. The first two characters in the sha value is the directory and the remainder is the filename.

   Example:
   If we had a ref of *32dfb09ddd7ccaf90d5e8f24b6d79d23b92816fb*
   the file we would attempt to download would be
   .git/objects/32/dfb09ddd7ccaf90d5e8f24b6d79d23b92816fb

3. Using the git ls-files --stage command we download each object reference

   Example: (truncated for brevity)

   100644 c3c996e3b8a5579d534bb2ada3ee2cde0a8eb6bd 0 blog/.htaccess
   100644 49403ecc2d8a343da95ad8d354b4f16a73f094d9 0 blog/index.php
   100644 d31195ab0e695f8b894f3875c57cefc227aa3af5 0 blog/license.txt
   100644 c4897a991a5db2ec1738af1b862d26f639a745c9 0 blog/readme.html

   So we would try and download the following objects

   .git/objects/c3/c996e3b8a5579d534bb2ada3ee2cde0a8eb6bd
   .git/objects/49/403ecc2d8a343da95ad8d354b4f16a73f094d9
   .git/objects/d3/1195ab0e695f8b894f3875c57cefc227aa3af5
   .git/objects/c4/897a991a5db2ec1738af1b862d26f639a745c9

4. Not all references can be retrieved this way. Because of how git structures its object database some of these files are put into pack files. As far as we could tell the pack file names could not easily be determined.

5. One last technique can potentially grab a bit more of the object directory. Using error messages created by the “git log” command. An error like the below is simply parsed and the object reference downloaded until we can’t make any more progress.

   error: Could not read 058ef249880a92b335acb3ecb80cb729d4e90be9

fatal: Failed to traverse parents of commit 32dfb09ddd7ccaf90d5e8f24b6d79d23b928

6. Finally the best part, we get to try and restore some actual data files. We do this by simply trying to “git checkout” for each file mentioned in “git ls-files” output.

HG

HG is quite different than git in how it stores its data files on disk. Because it doesn’t use some hash value for filenames it has to deal with encoding normalization for filenames due to different platforms. [1] To handle this we simply use the encodefilename function of the mercurial python library instead of having to understand it completely.

Once we have each file downloaded we can use the “hg revert filename” command to get a raw copy extracted from the repository and onto disk.

To completely understand this process take a look at the source available in the DVCS Pillage Toolkit as mentioned below.

BZR

To identify what files to download with bazaar we use the integrity checking command bzr provides “bzr check.” Using error messages presented to us we can determine exactly what files we need to download and recreate the repository. Eventually all files will be downloaded and the bzr check command will have happily extracted the data for us.

Some sites appear to block downloading the format files in branch, repository, and checkout. If this is the case what you can do is simply fake it. Sometimes putting the following values in allow for progress to be made.

.bzr/branch/format
  Bazaar Branch Format 7 (needs bzr 1.6)

.bzr/checkout/format
  Bazaar Working Tree Format 6 (bzr 1.14)

.bzr/repository/format

OTHER INFORMATION

In the case that you were not able to extract actual raw files from the repository you still have the potential to get a lot of useful data. Here are some thoughts on what you could do in this situation.

- Downloadable Files / File Listing
  In all three situations the index files or dirstate provides an nice list of what files are included in the repository. Here are some ideas of what to look for.

  - .sql
  - .tmp
  - *.tar / *.tar.gz / *.zip
  - .bak
  - .old
  - .crt / .pem

- Old Revisions
  Sometimes people put confidential information into repositories but then decide it was a bad idea. Checking older revisions of files

  GIT:
  ```
  git log --follow -p FILENAME
  ```

  HG:
  ```
  hg log FILENAME
  hg diff -r 10 -r 20 FILENAME
  ```

  BZR:
  ```
  bzr log
  ```
  [2]

- Email Addresses
  Logs contain some good data, included in those are often email addresses or usernames.

DVCS PILLAGE TOOLKIT

To automate the most of the explained techniques in this paper a toolkit called the DVCS Pillage Toolkit was created and is hosted on github. Each repository has its own unique tool to pillage and pwn the repository.

You can visit the project page here
https://github.com/ngenuity/DVCS-Pillage

or clone it using git

```
git clone git@github.com:ngenuity/DVCS-Pillage.git
```

Please submit any issues or feature requests via the github issue tracker.

CONCLUSION

While 100% extraction of a repository is not always possible it was determined that information leaked from an exposed DVCS meta directory can be very valuable to an attacker.

REFERENCES

[0] - http://www.kernel.org/pub/software/scm/git/docs/user-manual.html#the-index
[1] - http://mercurial.selenic.com/wiki/EncodingStrategy
[2] - http://doc.bazaar.canonical.com/latest/en/user-reference/log-help.html

Author:
Adam Baldwin, Chief Pwning Officer
nGenuity Information Service

adam_baldwin@ngenuity-is.com
@adam_baldwin