

# Tools for Discovering Credit Card and Social Security Numbers in Computer File Systems

## Introduction

Financial risks can be enormous if the right information falls into the wrong hands. Because of identity theft concerns, more and more organizations are adopting security policies to better protect information that is considered private and personal, such as credit card and Social Security numbers. This information should be guarded while in use, and securely deleted when no longer needed.

While such a policy sounds simple, many organizations don't know if their systems contain private personal information and if they do, where this information is located. Some databases may leave behind "temporary" files that contain Social Security numbers believed to be deleted, or old spreadsheets with credit card numbers may be buried in an obscure or rarely used subdirectory. It is not always obvious if a file system still contains sensitive information even when users believe the sensitive data has been deleted, but with the proper tools, credit card and Social Security numbers can be located in computer file systems.

## Audience and Scope

This paper is intended for system administrators at the University of Michigan who are conducting audits to locate credit card numbers and United States Social Security numbers<sup>1</sup> on their systems. This document does not cover incident management, the investigation of compromised systems, secure data removal or the encryption of data. (See "Eraser" in the Resources section for information about a free secure data removal tool.)

## Overview and Evaluation of Discovery Tools

*Warnings and disclaimers regarding the discovery tools:*

Before taking any action you should determine whether you intend to use a tool to investigate a compromised system, or as part of an audit of systems not believed to be compromised.

In the case of a compromised system, it is critical that the system be handled in accordance with forensic industry best practices. Forensic investigation is beyond the scope of this document.

If you believe you have a compromised system, it is critical that you contact Information Technology Security Services (ITSS) in accordance with the incident response procedure before taking any action. E-mail [security@umich.edu](mailto:security@umich.edu) or visit <http://safecomputing.umich.edu/> for more information.

**Note:** *The use of tools that recursively traverse file trees, like many of the tools mentioned here, will destroy valuable evidence (file access times, at the very least) if special measures are not taken to protect it.*

*Also, be aware that all of these (and similar) tools are not always accurate. They could find information that is not being sought (false positive), and they could fail to discover information that is being sought (false negative). It is extremely important to understand that simply because a tool appears to have found nothing, it does not mean that the intended information is not present.*

*Additionally, none of these tools are designed to find information that has been obfuscated, encrypted, or intentionally hidden in any manner.*

The tools covered in this paper only attempt to find numbers encoded as bytes representing the ASCII digit characters “0” through “9”. Data encoded as binary coded decimal (BCD), encrypted/obfuscated in even the most naïve manner, hidden using steganography, or encoded in any one of the almost infinite number of other possible encodings will not be discovered by these tools.

Anyone conducting an audit with a goal of finding private personal or other sensitive information should:

- Determine whether they are dealing with a compromised machine and take appropriate action
- Attempt to understand what applications are in use on the target system
- Attempt to understand the application(s) and application data format(s) associated with known data on the target system
- Use the most appropriate tool for the type of information being sought and likely to be present
- Securely delete any files created by the discovery tools that may give an attacker information as to where sensitive information can be found before it can be properly disposed of
- Use multiple tools if needed
- Manually verify every match found by a tool
- Attempt manual application-specific searches using the application(s) associated with the data present
- Realize that, even after all of this, undiscovered sensitive information may still exist on the target system

While no discovery tool can always be accurate, the use of these tools can lower the cost of the discovery process making it more practical and is significantly better than doing nothing.

## **Cornell Spider**

Spider comes in two different versions, “Linux” and Windows. The Linux version is distributed as a compressed tar file containing the client and server Perl scripts. While it is called the Linux version, it is not Linux-specific; it runs on other UNIX-like operating systems such as FreeBSD and OpenBSD. The Windows version is distributed as a typical Windows installer package containing most of the functionality of the Linux version with some additional Windows functionality.

The Linux version requires the manual installation of a large number of Perl and other dependencies, though many of the dependencies are optional. (For example, if ‘unzip’ is not installed, spider will run but will not be able to search the contents of ZIP files.) The Linux version consists of a client script and

a server script (written in Perl) that must be executed from the shell command line. The server script receives file data from one or more client scripts, either run locally or over the network, and checks the data against regular expressions which match some common ASCII representations of credit card and Social Security numbers. The list of regular expressions can be edited and extended by the user.

The Windows version of Spider comes in an installer package. Unlike the Linux version, it does have a graphic user interface, but it lacks other features like the ability to decode some of the more obscure file compression formats.

Spider is particularly good at weeding out false positives by using a “magic number” style file identification (as used by the UNIX ‘file’ command) to detect and disregard files (GIF images, for example) which are unlikely to contain targeted data. The file identification methods are also used to detect compressed and encoded files that can be decompressed or otherwise decomposed before applying the regular expression engine. As a result this tool can find interesting information that will go unnoticed by other less sophisticated tools that do not understand archive and compression file formats. It also tends to generate fewer false positives than other tools.

Spider is not without limitations. The current production version does not find credit card numbers that are encoded as 16 unbroken ASCII digits (in its default configuration), and it does not attempt to validate any credit card numbers against the Luhn algorithm<sup>2</sup> to avoid false positives. In one real-world case at the University of Michigan, the tool failed to identify over 20,000 credit card numbers in one database that were detectable by other tools. Currently, there is a Windows-beta version of Spider that attempts to validate credit card numbers using the Luhn algorithm and finds numbers that do not contain dashes or spaces. However, this beta version was not used in writing this document.

Spider may have some bugs, but the tool is being actively used and maintained, and problems are usually addressed quickly.

#### Spider Benefits:

- Free and source code is available (for the Linux version, but not the Windows version )
- Supports compressed files searches
- Can feed file data to a central analysis server over a network in an encrypted manner
- Low false positives
- Logs each match and shows the unmatched data before and after each match
- Current beta version contains Luhn validation of credit card numbers with dashes and spaces and heuristic validation of social security numbers

#### Spider Limitations:

- More prone to false negatives (tends to err on the side of false negatives rather than false positives)
- Time-consuming installation procedure for the Linux version (many dependencies require manual installation)

## SENF

SENF is a command line tool written in Java. The use of Java makes it cross-platform, but also causes it to require the presence of a Java runtime environment. The tool does not tell you the location of matches

in a matching file, but it does let you set a threshold on the number of matches in a single file that must exist for the file to be reported.

**SENF Benefits:**

- Easy to install
- Can be run from the command line
- Can be run from removable media without a Java virtual machine installed

**SENF Limitations:**

- Only lists the number of matches for each file rather than the specific matches and surrounding data in the file
- High false positives (tends to err on the side of false positives)
- No source code available

## **EnCase Forensic**

EnCase Forensic (referred to simply as “EnCase” from here on) is a commercial product from Guidance Software (see the Resources section) that is primarily a general computer forensics tool. It supports a very large variety of features that are not necessary, in most cases, just to find credit card numbers and Social Security numbers.

For Social Security numbers, EnCase uses a regular expression engine like most free tools. EnCase does not add much over the free tools when it comes to finding Social Security numbers except for a slick GUI to perform the search and manually check the matches.

For credit card numbers, EnCase comes with a script (written using “EnScript”, EnCase’s embedded scripting language) that finds numbers (encoded with ASCII digit characters) which conform to the Luhn algorithm used to pre-validate credit card numbers. If a 13 to 16 digit credit-card-like number starting with valid issuer identifier digits<sup>3</sup> (e.g., “6011” for the Discover Card) is found by the script, it checks it using the algorithm. If it does not conform, then the number is not a credit card number, and it is ignored. If it does conform, then it is reported in the output. It is still possible for arbitrary data that looks like a credit card number to conform to the Luhn algorithm without actually being a valid credit card number, so manual verification is needed. EnCase is good at presenting the raw context near a match to make manual verification as easy as possible.

**EnCase Benefits:**

- Comes with a good script for finding credit card numbers
- Has a nice graphic user interface that integrates discovery automation with a hex and text dump browser for manual verification of results
- Commercial support
- Large user community

**EnCase Limitations:**

- Is not free (Costs around \$2000 with an additional \$3000 for training.)
- Does not come with any sophisticated methods or default regular expressions for finding SSNs
- Is somewhat complex and, as a result, has a steeper learning curve compared to other tools

## **FTimes**

FTimes is a system “baselining” and evidence collection tool which can be useful for discovering credit card and Social Security numbers using regular expressions. FTimes does not come with a default list of regular expressions for detecting credit card and Social Security numbers. (We provide a sample configuration file below in the “Using” section.)

Like Cornell Spider, FTimes also supports downloading its configuration from a network server.

### **FTimes Benefits:**

- Free and source code is available (for the Linux version, but not the Windows version.)
- Can feed file data to a central analysis server over a network in an encrypted manner (“integrity server” using SSL)

### **FTimes Limitations:**

- Relies solely on regular expressions (does not implement the Luhn algorithm)
- Does not decode and process the contents of ‘zip’ and other compressed or archive file formats
- High false positive rate

## **Using the Tools**

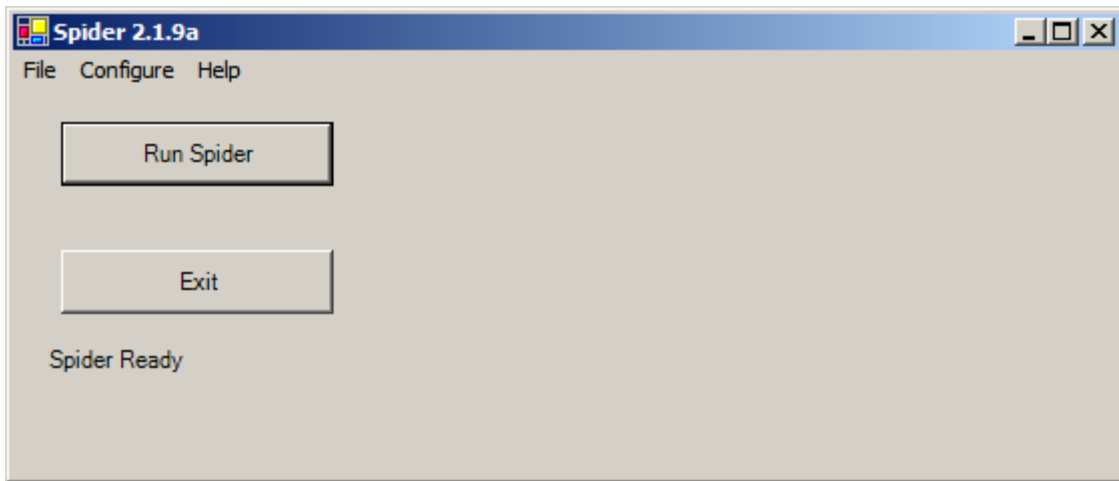
This section contains some simple “how to” information to help people get started with the tools described above.

### **Windows**

This section describes the use of credit card and Social Security number discovery tools available for Windows.

**Cornell Spider:** The Windows version of Cornell Spider can be downloaded from: <http://www.cit.cornell.edu/security/tools/>. After installation it must be run using an account with administrative privileges. (This can be done using “Run As...” in the explorer shell context menu.)

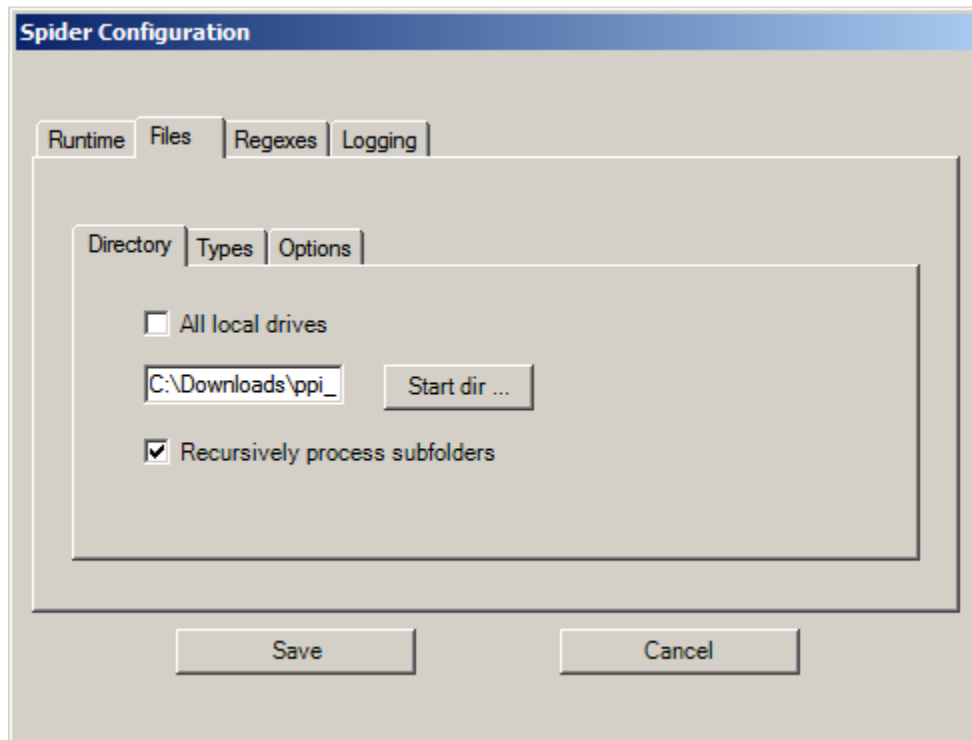
A full treatment of how to use Spider is beyond the scope of this document, but here is some basic information to get started.



**Figure 1: Main Spider Window**

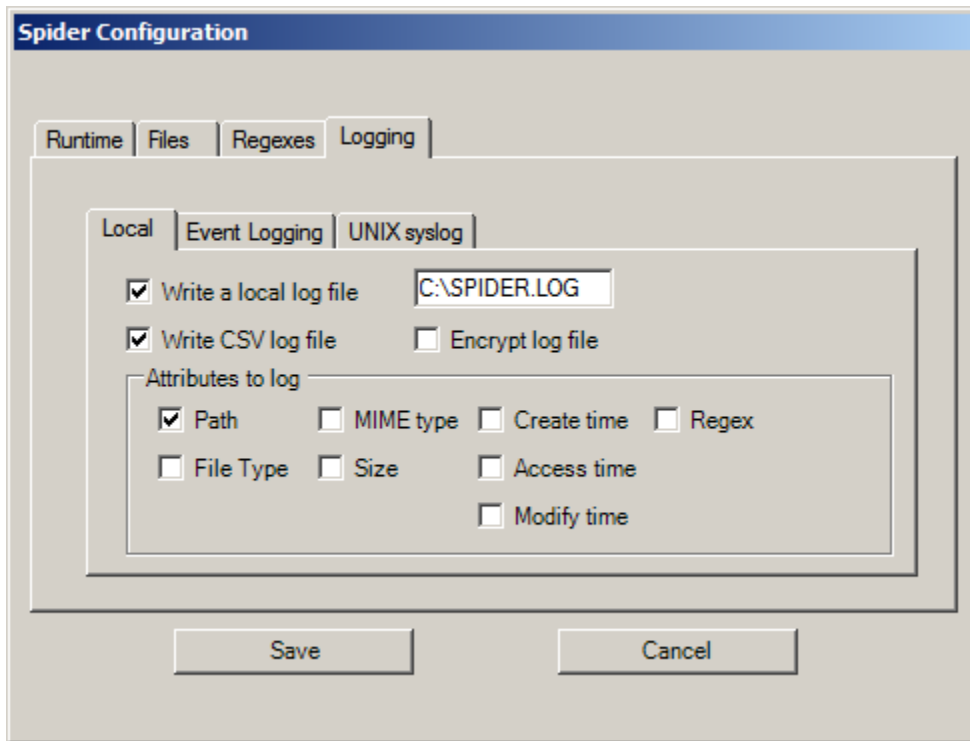
First, launch the program from the Start menu using an administrator account or “Run As” with administrator credentials.

If you want to scan only a subset of the files on the system, select Configure→Settings and click the files tab.



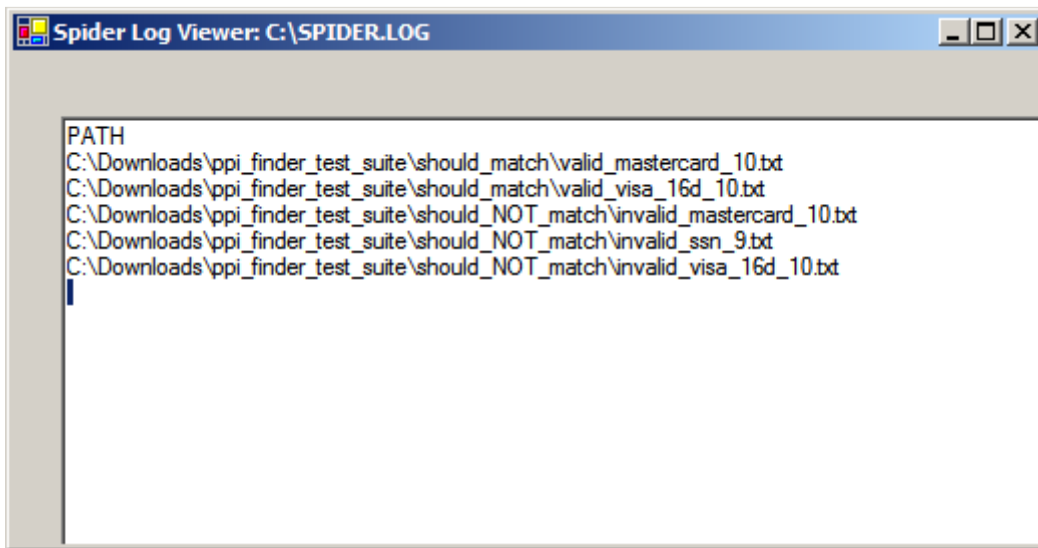
**Figure 2: Configuring the location for Spider to scan.**

You may also wish to change the location of the log file or other logging options. Some of the options are shown in Figure 3.



**Figure 3: Configuring Spider’s logging parameters in version 2.1.9a.**

When done configuring, click the “Run” button in the main window. To view the log file, use File→View Log.



**Figure 4: Viewing the Spider log file with the built-in viewer.**

**SENF:** SENF is written in Java and requires that a Java Runtime Environment (JRE) be available for the target machine. The JRE does not necessarily need to be installed on the target system.

**Note:** For those not wanting to install Java on their systems, the following self-contained instance of the JRE is an option: <http://home.arcor.de/blackwell/javahappiness.html>. Copy

*JavaRunner and SENF to a USB dongle, follow the JavaRunner instructions, and perform spot checks as desired.*

Run SENF as shown below using the ‘-p’ option to specify the directory to search. (Run it with ‘-h’ for a list of other options.)

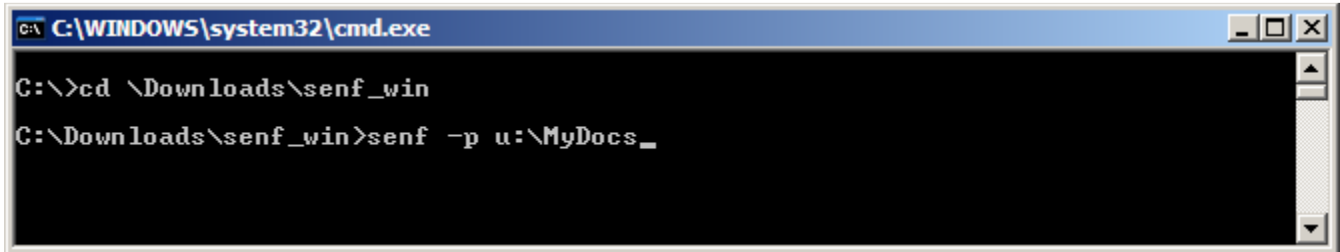


Figure 5: Invoking SENF From the cmd.exe prompt

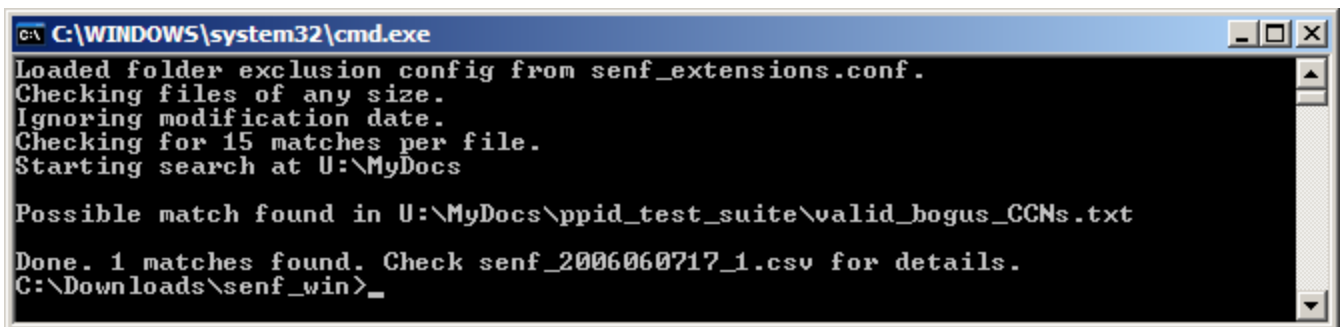


Figure 6: SENF execution messages.

SENF generates a .csv file containing information on the matches found. View this file using a text editor or by importing it into something that can parse ‘|’ character-delimited files.

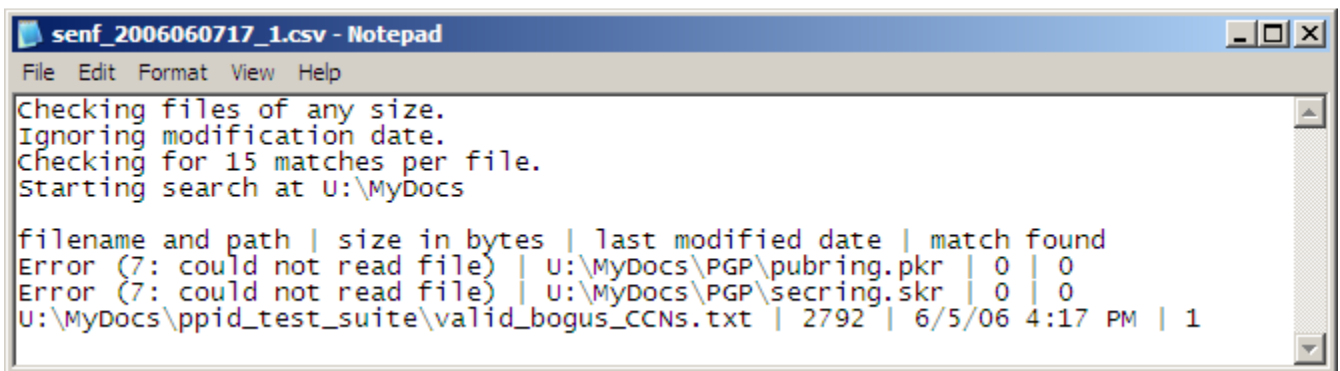


Figure 7: Log file output from SENF.

**EnCase:** Among other things, the commercial application EnCase Forensic can be used to discover credit card numbers. The procedure for doing this is:

1. Create a new case with File→New
2. Either:
  - a) Add individual files
    - a. Click the Cases tab

- b. Click the Entries 2nd level sub-tab
  - c. Click the Home 3rd level sub-tab
  - d. Drag-and-drop files/folders onto Entries item (not tab) in the list below the tabs
- b) Perform media acquisition (see EnCase documentation)

After this step the UI should look something like this:

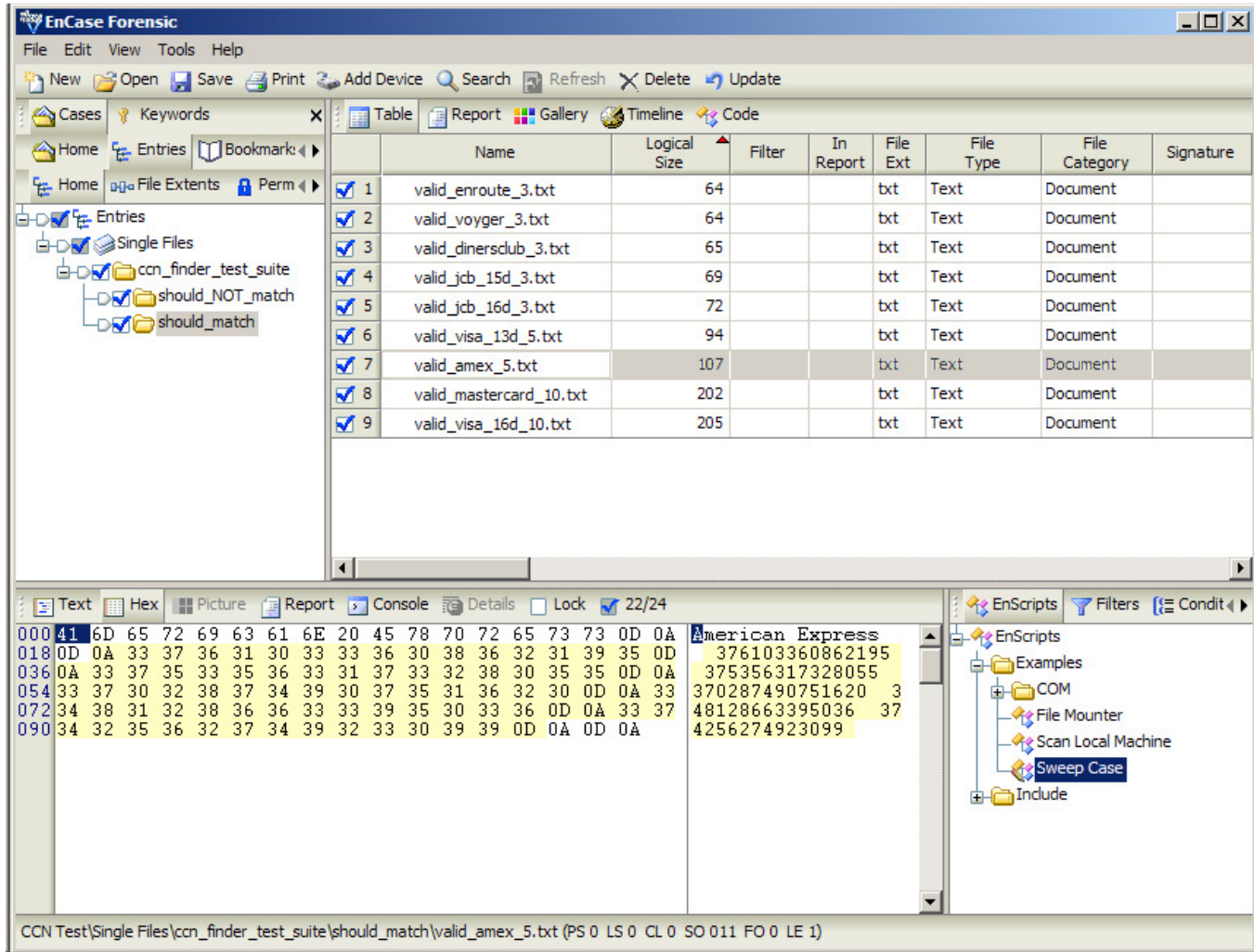
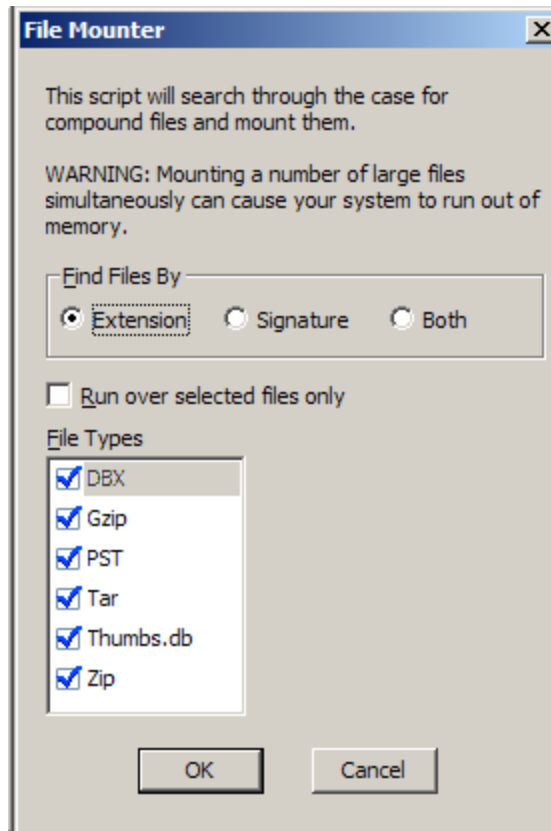


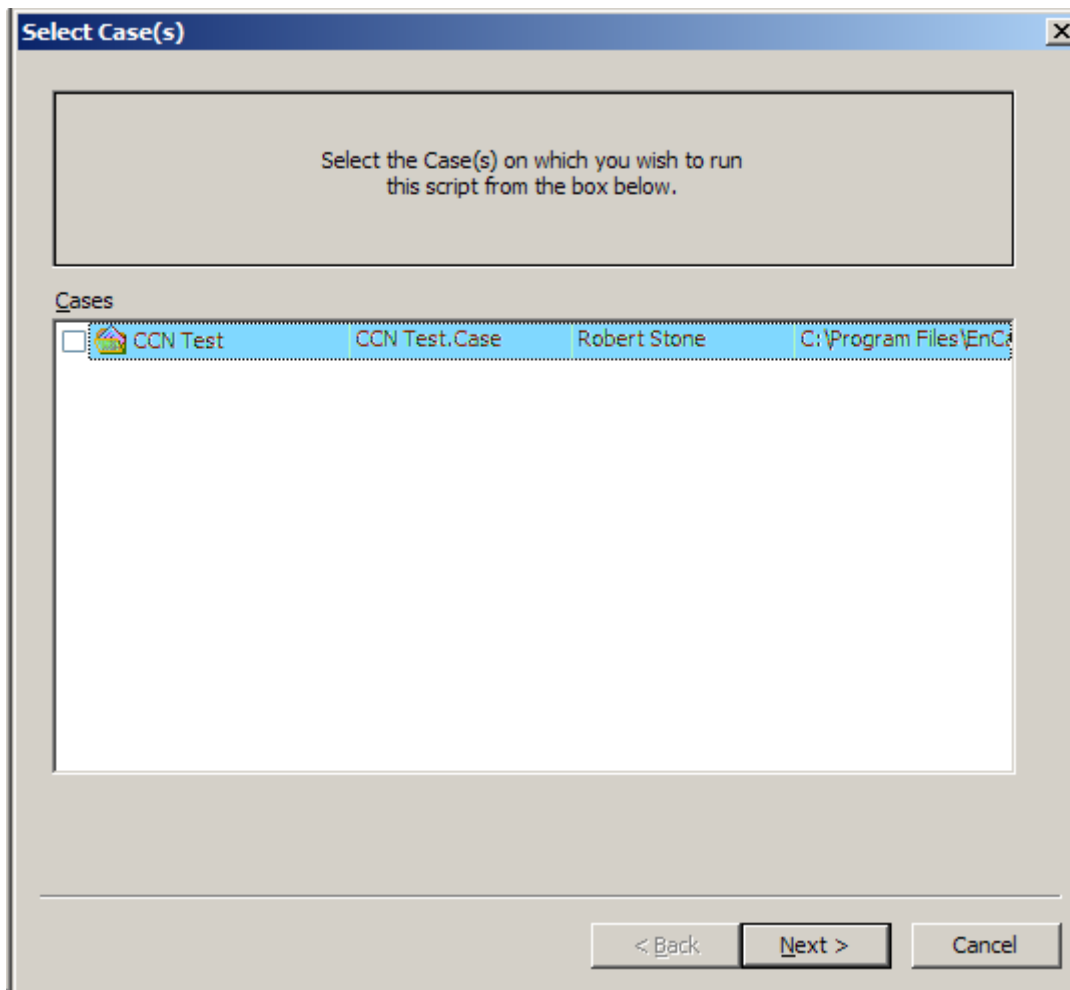
Figure 8: Selecting files to search in EnCase.

3. Click the check box next to the Entries list item under the Cases→Entries→Home tab. All files should be selected.
4. Right click on File Mounter and select Run...



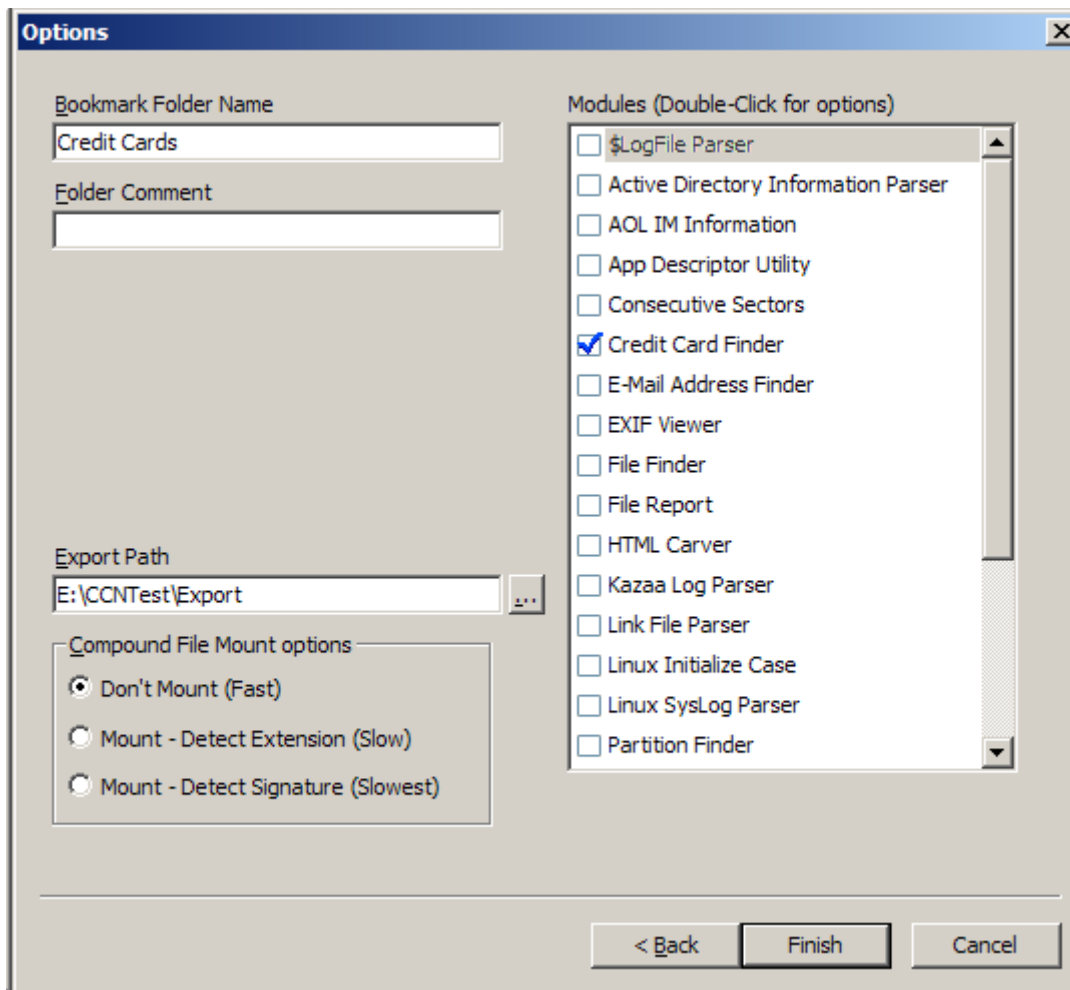
**Figure 9: Selecting archive files to extract in EnCase.**

5. Make sure all of the file types are checked and click OK. If any archive files in one of the supported formats are present they should be mounted as folders.
6. In lower right corner, right click on Enscripts→Examples→Sweep Case and select "run".
7. Check the case(s) to search and click next.



**Figure 10: Selecting the case(s) to search for credit card numbers using EnCase.**

8. Clear all checkboxes except Credit Card Finder.



**Figure 11: Select only the Credit Card module.**

9. Click Finish.
10. Click the Search Hits tab under the Cases top level tab.
11. Expand the Single Files item.
12. Select “Credit Cards 1”.

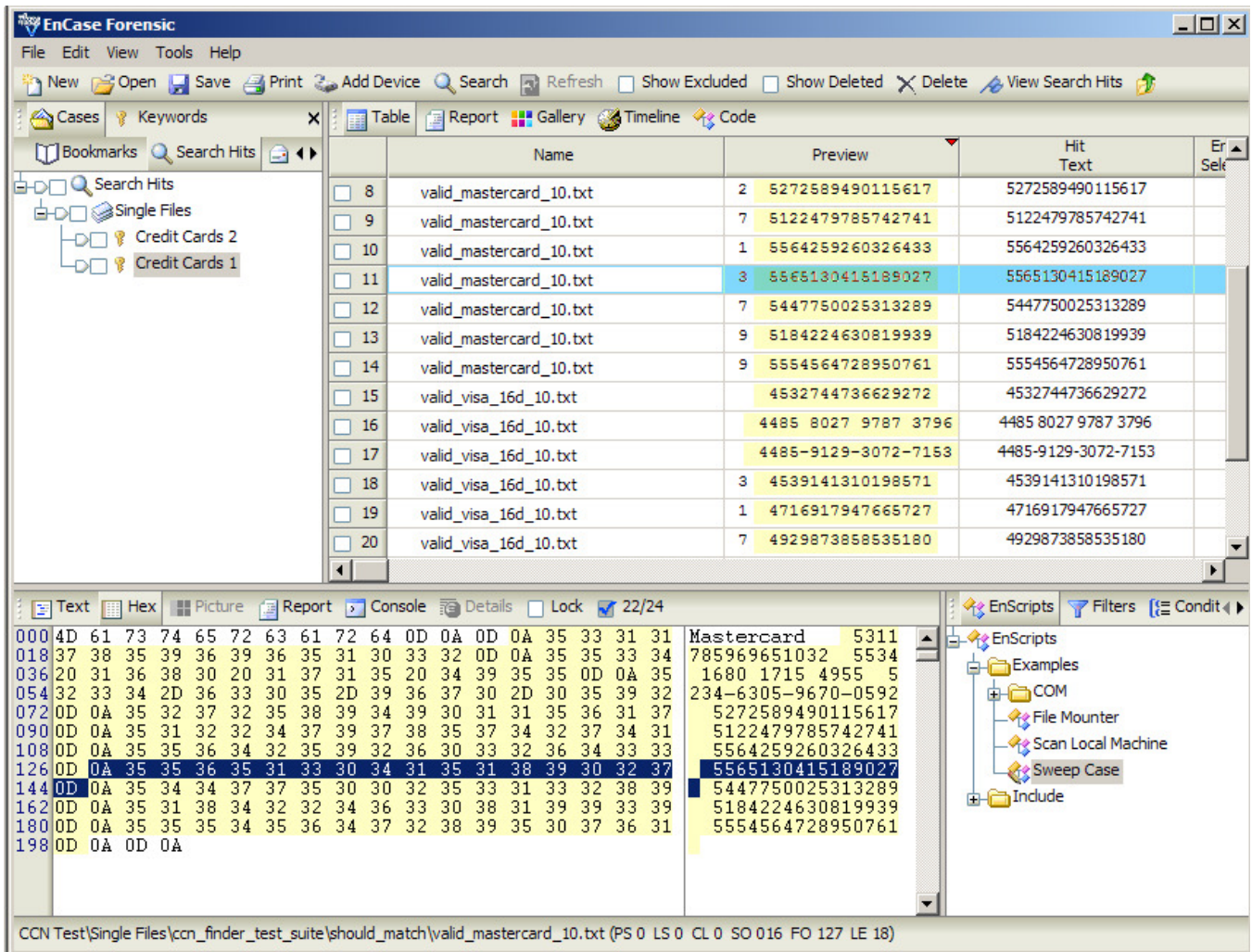


Figure 12: Viewing credit card number matches in EnCase.

13. Select a search hit in the upper right pane.

14. In the lower left pane click the Text or Hex tab. The file context for that match will be shown.

The credit card script that comes with EnCase only finds MasterCard and Visa numbers.

**FTimes:** While FTimes runs under Windows, it is distributed only in source code form. Describing how to build the source is beyond the scope of this document, but if you have built or otherwise obtained executables then usage is almost exactly the same as the UNIX version described below. Please see FTimes in the UNIX section below.

## UNIX

This section discusses the use of tools compatible with UNIX-like operating systems.

**Cornell Spider:** In order to install the Linux version of Spider, you first need to install a number of dependencies if you do not already have them. For more information on the Linux version:

<http://www.cit.cornell.edu/computer/security/tools/spider-linux.html> . Currently spider requires Perl 5.6.1 or higher with the following modules:

- MD5
- SHA1
- Crypt::CBC and Crypt::Blowfish
- Config::IniFiles
- File::LibMagic (requires 'libfile' from ftp://ftp.astron.com/pub/file/)
- File::Path
- Archive::Zip

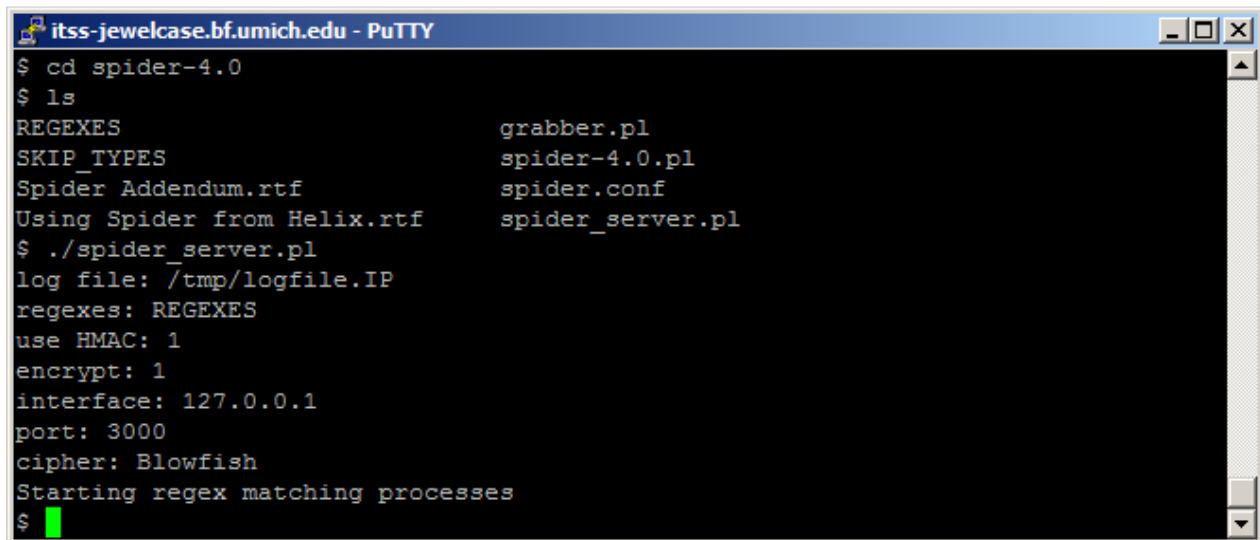
If you are missing any modules, use 'perl -MCPAN -e shell' to install the missing ones.

You may also want to install one or more of the following optional programs to benefit from their features:

- file
- wvText (for converting Word docs to text)
- unzip
- unrar
- lha
- unzoo
- arj
- readpst

Download the Spider gzipped tar file from <http://www.cit.cornell.edu/computer/security/tools/spider.html> and extract it inside an appropriate directory such as your home directory.

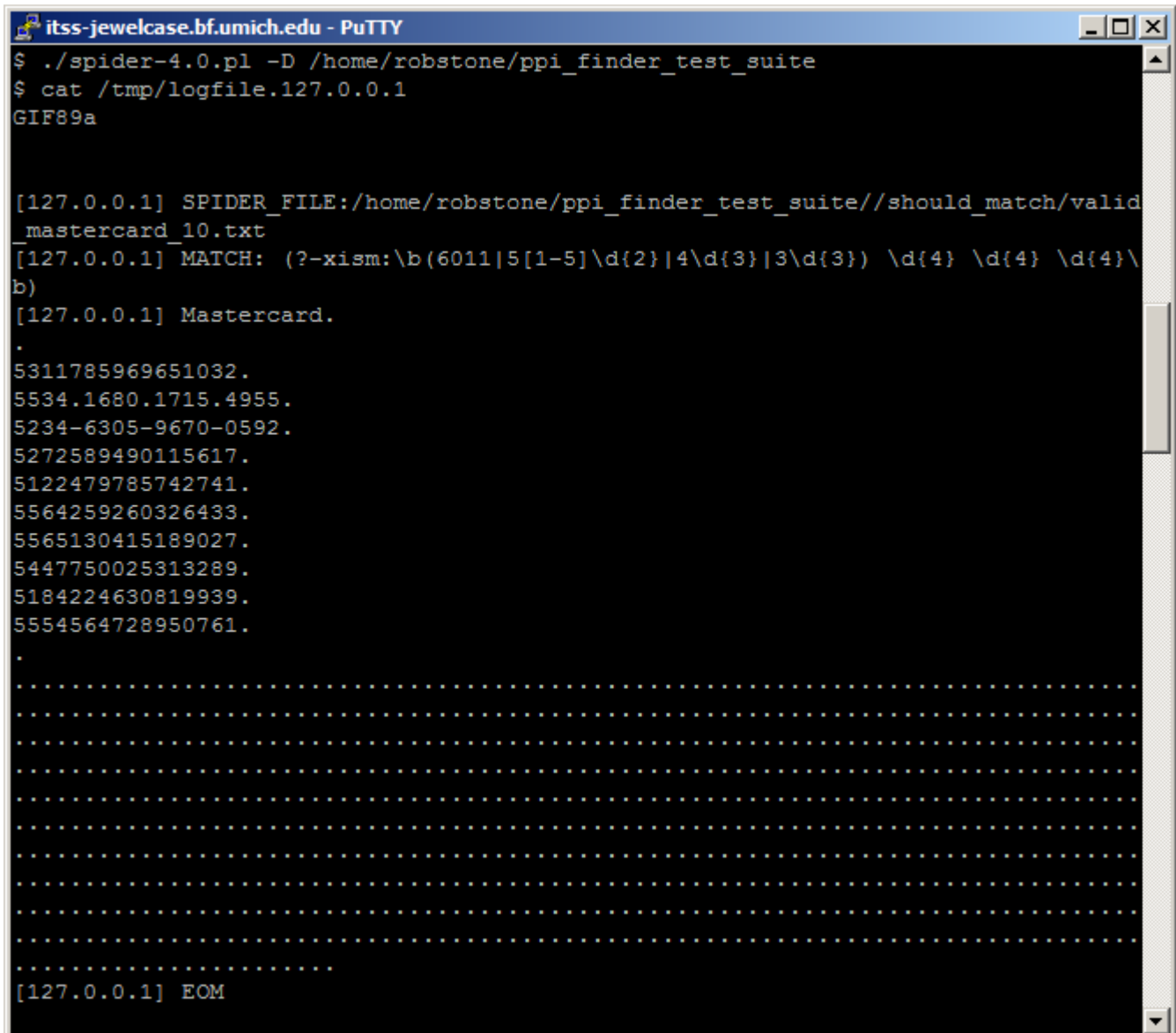
First, launch the Spider server as shown in Figure 13.



```
itss-jewelcase.bf.umich.edu - PuTTY
$ cd spider-4.0
$ ls
REGEXES          grabber.pl
SKIP_TYPES      spider-4.0.pl
Spider Addendum.rtf  spider.conf
Using Spider from Helix.rtf  spider_server.pl
$ ./spider_server.pl
log file: /tmp/logfile.IP
regexes: REGEXES
use HMAC: 1
encrypt: 1
interface: 127.0.0.1
port: 3000
cipher: Blowfish
Starting regex matching processes
$
```

Figure 13: Executing the Spider server for local use.

Next, execute the Spider client script on a directory containing files to search. This will not appear to do anything based on the output from the client, but the server's log file should contain a listing of the results as shown in Figure 14.



```
itss-jewelcase.bf.umich.edu - PuTTY
$ ./spider-4.0.pl -D /home/robstone/ppi_finder_test_suite
$ cat /tmp/logfile.127.0.0.1
GIF89a

[127.0.0.1] SPIDER_FILE:/home/robstone/ppi_finder_test_suite//should_match/valid
_mastercard_10.txt
[127.0.0.1] MATCH: (?-xism:\b(6011|5[1-5]\d{2}|4\d{3}|3\d{3}) \d{4} \d{4} \d{4}\
b)
[127.0.0.1] Mastercard.
.
5311785969651032.
5534.1680.1715.4955.
5234-6305-9670-0592.
5272589490115617.
5122479785742741.
5564259260326433.
5565130415189027.
5447750025313289.
5184224630819939.
5554564728950761.
.
.....
[127.0.0.1] EOM
```

Figure 14: Example output from Spider's log file. The contents of the file surrounding the match are shown to make it easier to manually verify if a match is valid or not.

**SENF:** SENF usage under UNIX-like operating systems and Mac OS X is basically the same as under Windows, as described above. Instead of executing 'senf' though you should execute 'senf.sh' or rename the shell script to 'senf', place it somewhere in the path, and execute it as you would in Windows.

See the SENF section under the Windows section above for more information on using SENF.

**FTimes:** Download FTimes from <http://ftimes.sourceforge.net/FTimes/> and install it.

First, create a configuration file called "ppi.cfg" with the following contents.

```

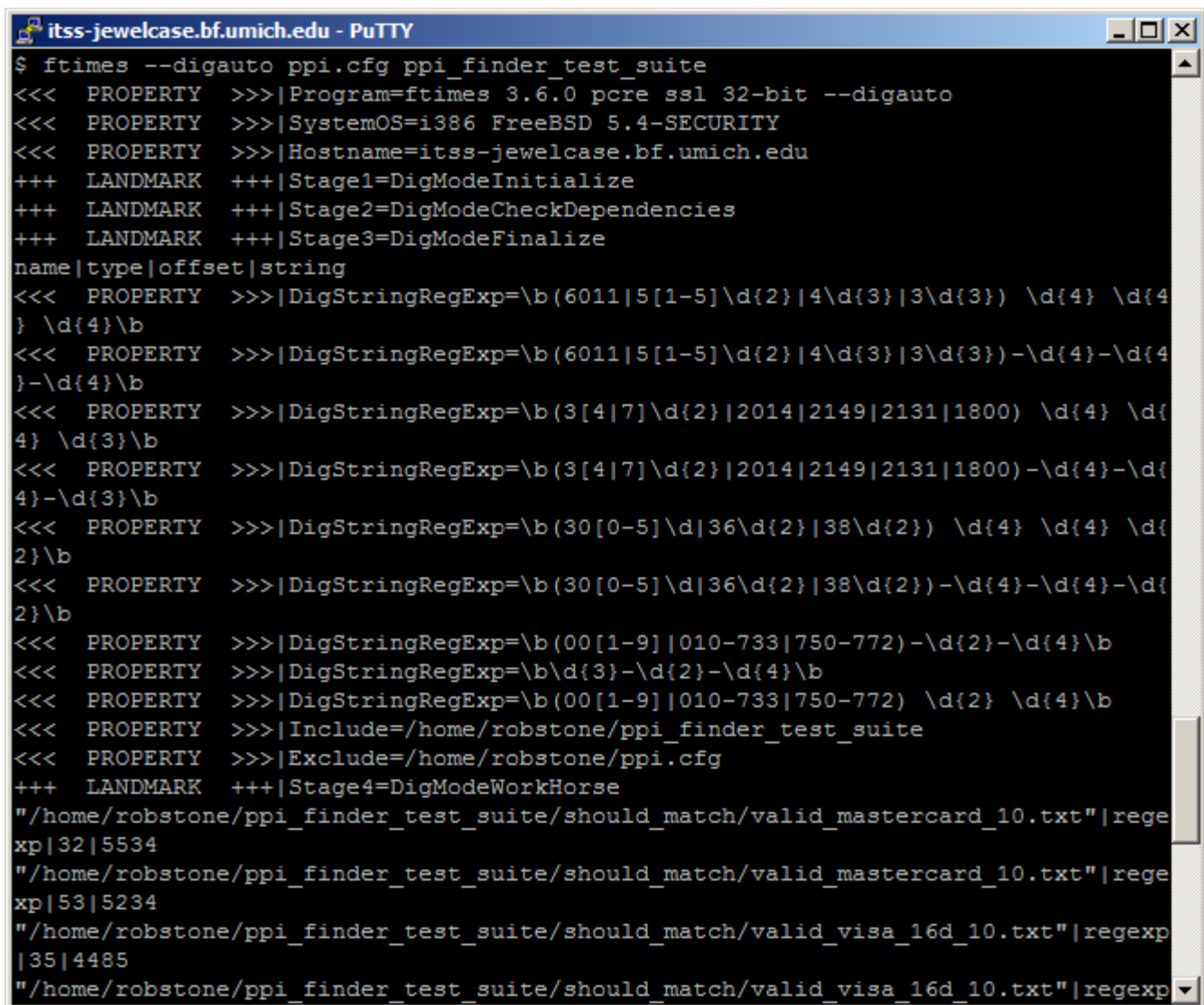
DigStringRegExp=\b(6011|5[1-5]\d{2}|4\d{3}|3\d{3}) \d{4} \d{4} \d{4}\b
DigStringRegExp=\b(6011|5[1-5]\d{2}|4\d{3}|3\d{3})-\d{4}-\d{4}-\d{4}\b
DigStringRegExp=\b(3[4|7]\d{2}|2014|2149|2131|1800) \d{4} \d{4} \d{3}\b
DigStringRegExp=\b(3[4|7]\d{2}|2014|2149|2131|1800)-\d{4}-\d{4}-\d{3}\b
DigStringRegExp=\b(30[0-5]\d|36\d{2}|38\d{2}) \d{4} \d{4} \d{2}\b
DigStringRegExp=\b(30[0-5]\d|36\d{2}|38\d{2})-\d{4}-\d{4}-\d{2}\b
DigStringRegExp=\b(00[1-9]|010-733|750-772)-\d{2}-\d{4}\b
DigStringRegExp=\b\d{3}-\d{2}-\d{4}\b
DigStringRegExp=\b(00[1-9]|010-733|750-772) \d{2} \d{4}\b

```

(These regular expressions for credit card numbers and US Social Security numbers are taken from Cornell Spider's default 'REGEXES' file.)

Execute FTimes as follows:

```
$ ftimes --digauto ppi.cfg <directory to search>
```



```

itss-jewelcase.bf.umich.edu - PuTTY
$ ftimes --digauto ppi.cfg ppi_finder_test_suite
<<< PROPERTY >>>|Program=ftimes 3.6.0 pcre ssl 32-bit --digauto
<<< PROPERTY >>>|SystemOS=i386 FreeBSD 5.4-SECURITY
<<< PROPERTY >>>|Hostname=itss-jewelcase.bf.umich.edu
+++ LANDMARK +++|Stage1=DigModeInitialize
+++ LANDMARK +++|Stage2=DigModeCheckDependencies
+++ LANDMARK +++|Stage3=DigModeFinalize
name|type|offset|string
<<< PROPERTY >>>|DigStringRegExp=\b(6011|5[1-5]\d{2}|4\d{3}|3\d{3}) \d{4} \d{4}
} \d{4}\b
<<< PROPERTY >>>|DigStringRegExp=\b(6011|5[1-5]\d{2}|4\d{3}|3\d{3})-\d{4}-\d{4}
}-\d{4}\b
<<< PROPERTY >>>|DigStringRegExp=\b(3[4|7]\d{2}|2014|2149|2131|1800) \d{4} \d{
4} \d{3}\b
<<< PROPERTY >>>|DigStringRegExp=\b(3[4|7]\d{2}|2014|2149|2131|1800)-\d{4}-\d{
4}-\d{3}\b
<<< PROPERTY >>>|DigStringRegExp=\b(30[0-5]\d|36\d{2}|38\d{2}) \d{4} \d{4} \d{
2}\b
<<< PROPERTY >>>|DigStringRegExp=\b(30[0-5]\d|36\d{2}|38\d{2})-\d{4}-\d{4}-\d{
2}\b
<<< PROPERTY >>>|DigStringRegExp=\b(00[1-9]|010-733|750-772)-\d{2}-\d{4}\b
<<< PROPERTY >>>|DigStringRegExp=\b\d{3}-\d{2}-\d{4}\b
<<< PROPERTY >>>|DigStringRegExp=\b(00[1-9]|010-733|750-772) \d{2} \d{4}\b
<<< PROPERTY >>>|Include=/home/robstone/ppi_finder_test_suite
<<< PROPERTY >>>|Exclude=/home/robstone/ppi.cfg
+++ LANDMARK +++|Stage4=DigModeWorkHorse
"/home/robstone/ppi_finder_test_suite/should_match/valid_mastercard_10.txt"|rege
xp|32|5534
"/home/robstone/ppi_finder_test_suite/should_match/valid_mastercard_10.txt"|rege
xp|53|5234
"/home/robstone/ppi_finder_test_suite/should_match/valid_visa_16d_10.txt"|regexp
|35|4485
"/home/robstone/ppi_finder_test_suite/should_match/valid_visa_16d_10.txt"|regexp

```

Figure 15: Partial output from FTimes under FreeBSD.

## Conclusions and Recommendations

Currently, Cornell Spider is the best tool overall because of its detection, client/server, file identification, and compressed file inspection features. While it does not always detect numbers properly, the tool is actively supported and most problems are fixed relatively quickly when reported to the author. There is also a support mailing list for Spider users (see Resources below.)

While no tools for finding credit card and Social Security numbers are 100% reliable, these tools can be extremely useful. Users are advised to take caution and understand that no tool is perfect. Currently, none of the tools are very mature, and users are still likely to run into many false negatives and false positives, but these tools do fill a niche that is lacking, and users can expect to see them become more refined as time goes on.

## Resources

Cornell Spider

<http://www.cit.cornell.edu/computer/security/tools/>

Cornell Spider Support e-mail list:

Send e-mail to [lyris@cornell.edu](mailto:lyris@cornell.edu) with `subscribe cuspider-l "Your Name"` in the message.

Eraser (secure data removal tool)

<http://www.tolvanen.com/eraser/>

FTimes

<http://ftimes.sourceforge.net/FTimes/>

Guidance Software (EnCase)

<http://www.guidancesoftware.com/>

Running SENF Remotely Using Windows Mapped Drives

<http://www.ce.utexas.edu/senf/senf.htm>

SENF

<https://webpace.utexas.edu/xythoswfs/webui/weiland/www/senf/1.rocklee.17?action=frameset&subaction=print&uniq=5yfxox>

Structure of Social Security Numbers

<http://www.cpsr.org/prevsite/cpsr/privacy/ssn/ssn.structure.html>

## References

1. [Social Security number](http://en.wikipedia.org/wiki/Social_security_number) (United States), Wikipedia ([http://en.wikipedia.org/wiki/Social\\_security\\_number](http://en.wikipedia.org/wiki/Social_security_number))
2. [Luhn Algorithm](http://en.wikipedia.org/wiki/Luhn_algorithm), Wikipedia ([http://en.wikipedia.org/wiki/Luhn\\_algorithm](http://en.wikipedia.org/wiki/Luhn_algorithm))
3. [Anatomy of Credit Card Numbers](http://www.merriampark.com/anatomycc.htm) (<http://www.merriampark.com/anatomycc.htm>)