Automatically Creating Realistic Targets for Digital Forensics Investigation

Frank Adelstein
ATC-NY
33 Thornwood Drive, Suite 500
Ithaca, NY 14850
Contact: frank@atc-nycorp.com

Yun Gao
Golden G. Richard III
Department of Computer Science
University of New Orleans
New Orleans, LA 70148
Contact: *golden@cs.uno.edu*

Abstract

The need for computer forensics education continues to grow, as digital evidence is present in more crimes, whether the crimes directly involve computers or not. An essential component of training in computer forensics is hands-on, realistic laboratory assignments. Creating detailed, realistic lab assignments, however, is a difficult task. The "crime" must be played out on the machine, often in real-time, since timestamps present in numerous places in the system, such as files and logs, must be discovered and examined by students. Developing, running, and evaluating the labs can be labor intensive and instructors have limited time to spend on creating and grading laboratory experiments.

We are developing FALCON (Framework for Laboratory Exercises Conducted Over Networks), an extensible framework that addresses the problem of creating, running, and evaluating detailed, realistic computer laboratory assignments in computer forensics. FALCON includes a component that enables instructors to set up scenarios on virtual target machines for the students to investigate. Existing tools for both "live" and "dead" machine investigations can be integrated into FALCON. In addition, FALCON logs all student activity for automated assessment of student performance. Currently, FALCON is a work in progress and some tasks remain manual. The goal is to automatically transform high-level descriptions of digital forensics scenarios into detailed investigative targets which contain activities derived from the scenarios, as well as historical activity (timestamps, logs, history, etc.). While the initial version of FALCON focuses on computer forensics, it will be extensible to other areas, such as incident response, as well as general computer security instruction.

Keywords: computer forensics, computer security, education

1 Introduction

The number of computer-based crimes, including fraud, identity theft, embezzlement, and child pornography, is increasing. Furthermore, many crimes not typically classified as computer-based, such as drug trafficking, now often leave a digital trail. Our society needs more highly-trained computer forensic investigators who can analyze digital evidence and identify the perpetrators of these crimes. Computer forensic training is a relatively new area with an increasing number of schools offering courses to meet these needs.

The field of computer forensics encompasses a wide range of topic areas in computer science, such as extraction and preservation of digital evidence, as well as various legal issues, such as "chain of custody" and admissibility of evidence in court [8,9]. In addition, students are required to be familiar with computer forensic tools. However, knowing the subject matter is not enough. Students must be able to put all of the pieces together to solve crimes. They need opportunities for practicing forensic investigation and their skills need to be assessed in a comprehensive way. Tools for assessing the synthesis of computer forensic concepts do not exist.

Typically, digital forensics training is accomplished using a combination of classwork (i.e., lectures and recitations) and lab work. Because of the difficulty in preparing complex lab exercises, most exercises that are currently being offered cover only a small portion of a forensic problem. Such simplified laboratory exercises have limited benefits to students, leaving them unprepared for real-world situations. For example, if the largest storage medium a student has analyzed in a computer forensic course is a 1 megabyte floppy disk, the student is likely to be ill-prepared to analyze media 4-6 orders of magnitude larger.

A typical digital investigation might target a user stealing files or resources, running an illegal server, or using steganography to hide information. Evidence may also be contained in email, web page caches, and chat logs. A running system may have many users, each running numerous processes and owning numerous files. Most of the activity on a system, as well as most of the users, will not be malicious, providing "background noise." Because the forensic investigator will only be given a small amount of information initially, he or she must sift through a large amount of data in order to determine what happened, how it was done, and who was responsible for it. A realistic lab must reflect this situation. Not only must the users on a forensic target created for a lab have many files, the timestamps on the files must be consistent. Inconsistent timestamps may damage the realism of the lab because it makes normal behavior look suspicious or hides suspicious behavior. Note that even if a perpetrator in a real case wipes the timestamps on the files, there should be a consistent pattern.

In real life, forensic investigators must look for the "needle in the haystack"—realistic lab exercises should have large haystacks and small needles. In the real world, users not directly involved in an incident will still have a history. They will have sent and received email, browsed web pages, entered commands, and written to and read from files. A realistic lab must have well-defined users—the attacker must not be the only one with an "interesting" history.

Creating a realistic laboratory exercise is a tedious, time-consuming process. Although disk images for investigation may be harvested from a variety of sources (e.g., from e-bay, a personal server, etc.), the authors have learned from personal experiences that these sources are far from ideal. First, it is generally difficult to frame the contents of the image in terms of a scenario for investigation, e.g., a computer crime. Second, drive images extracted from unknown sources may contain contraband or extremely personal material, such as medical data or financial information. Third, the instructor must perform a detailed investigation herself to determine what evidence is present before an assignment can be crafted. Drawing more on personal experience, the authors note that manually constructing a set of "target" machines for a realistic investigation scenario is a tedious, time-consuming process because the instructor must generate all of the activity that is occurring and has occurred on the machines.

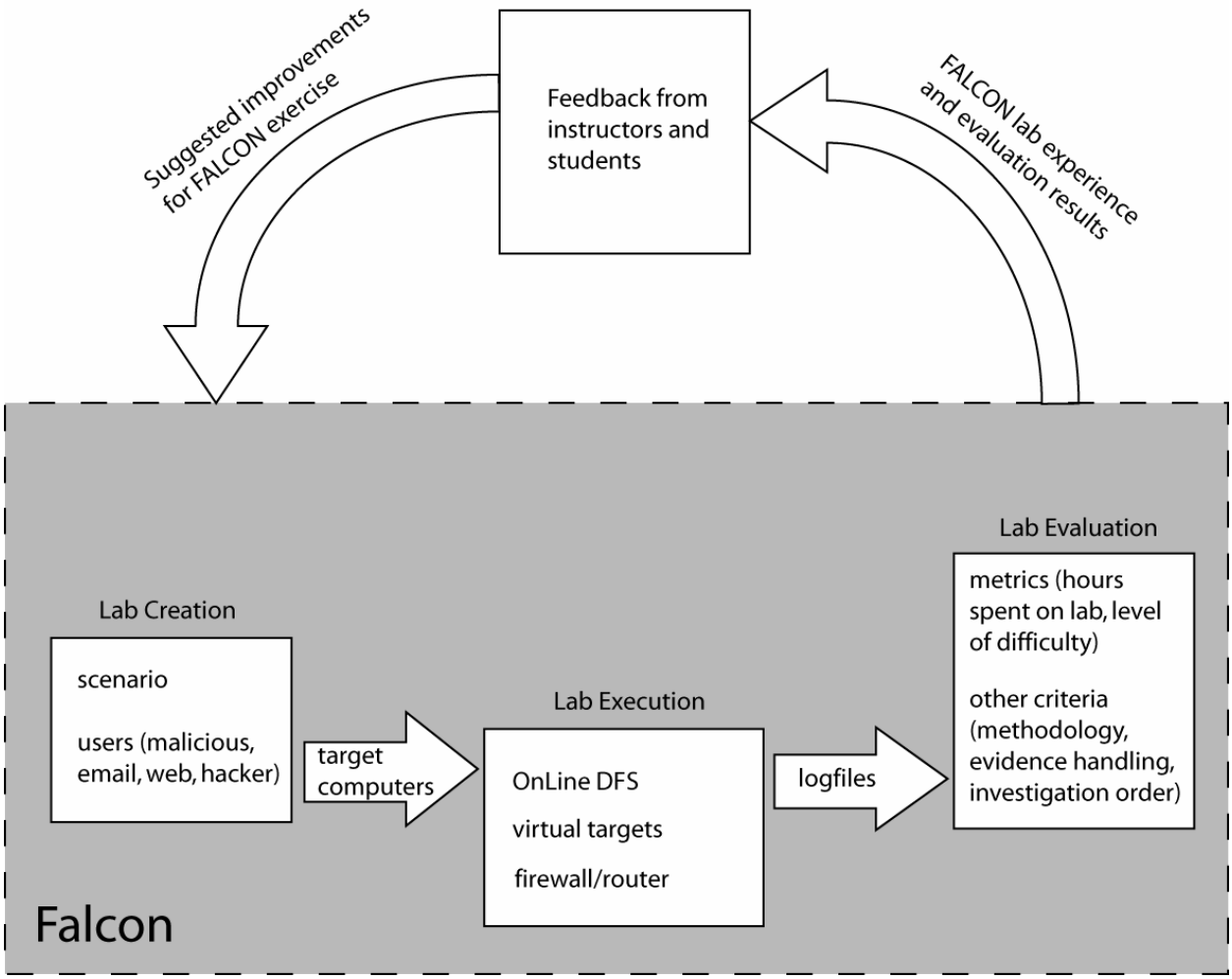

Figure 1: FALCON and the lab development process

Evaluating a complex lab completely and consistently for all students is also very challenging. Because a perpetrator's actions typically leave evidence in multiple places on the computer, there are many paths a student can take to find evidence of what happened. Currently, no means of obtaining detailed records of student investigative actions exists, and if such a record could be obtained, it would be lengthy and hard to analyze by hand. Instructors need automated assistance for student evaluation.

Another issue in digital forensics training is that computer laboratory facilities are expensive to set up and time consuming to staff and maintain. Also, they require that the students be physically present, which limits laboratory use by students who are disabled, live far away, or have full-time work schedules that restrict their available time. One approach to ease this burden is to move some of the training out of the lab, by allowing students to perform some of the laboratory exercises remotely, e.g., at home. In addition, remote access allows for the possibility of using a virtual environment in which a single machine can appear as different machines to different students (thus, reducing the hardware resources required). Many laboratories also require lab personnel to monitor the facility when labs are open. Because remote facilities do not require human security monitors, students can have

potentially around-the-clock access to these facilities.

In this paper, we discuss FALCON (Framework for Laboratory Exercises Conducted Over Networks), our architecture for the automatic creation, deployment, and evaluation of digital forensics laboratory exercises. FALCON consists of a Lab Creation Tool (LCT), a Lab Execution Environment (LEE), and a Lab Evaluation Tool (LET). The LCT allows an instructor to carefully specify the types of activity, both malicious and otherwise, that should reflected on a target computer. These activities include editing files, running network services, etc. The LEE supports the forensic investigation itself, logging the activities of students. Finally, the LET can be used to evaluate a laboratory exercise, providing details on what actions students took, which dead ends were pursued, etc. The paper also presents some of our initial results in creating, administering, and evaluating sophisticated digital forensics laboratory exercises using a prototype of FALCON.

2 FALCON Architecture

The FALCON architecture is illustrated in Figure 1 and consists of 3 components, the Lab Creation Tool (LCT), the Lab Execution Environment (LEE), and the Lab Evaluation Tool (LET). Each FALCON component is discussed below.

2.1 Lab Creation Tool

The FALCON Lab Creation Tool (LCT) will automatically configure a set of target machines for a lab exercise based on a specification that the instructor provides. Such a specification includes descriptions of both malicious and non-malicious activity on the target machines. Non-malicious activity includes users and their actions that are unrelated to the investigation. For example, a scenario specification might involve ten users, including six programmers, three web and mail users, and one "hacker." The LCT creates the users and populates their directories with representative tools and files, creating the appropriate time stamps on the files and log entries within the appropriate logs. For web and email users, their web cache and web history file is filled with appropriate web pages and a saved mail folder with various messages. Much of the email might be "spam" but with the headers reflecting an accurate time reference. The programmer class of users "use" software development tools, including compilers and editors.

An example of a user specification:

```
user {
    name = next("Namelist");
    class = web, mail;
    tools = pine, firefox |
         mozilla, spamassassin;
    timeframe = 2005-03-21 :
                2005-05-12;
    onlinehistory = daily*4;
}
```

The above specification pulls the user name from the next entry in the file "Namelist." The class "web" and "mail" defines the activity and types of files in the home directory. Note that a class specifies a narrow type of activity and a user can belong to more than one class. The class can be parameterized with options, such as what browser and mail tools are used. The timeframe defines the beginning and end of the user's activity. "onlinehistory" describes how often the user is online, in this case 4 times per day, on average. More complex specifications might be defined as needed.

The overall scenario is described as a series of events, each of which includes:

- An optional label for the event, so other events can refer to it.
- The probability that this event occurred. If the command is always to be executed, the value will be 1.0. The event may also depend on other events.
- The command to execute.

- Who executed it (note that an alias will be used, so that a set of machines can be created, each with a different "bad guy").
- The time it occurred. This can be absolute or relative, with a random component.

An example of an event specification:

```
event {
    label = "syslog-nuke";
    probability = 1.0;
    command = "cd /var/logs;
               rm syslog";
    name = next ("Namelist");
    time = 5min after
      event("install-backdoor");
}
```

The LCT uses the information in the specification to take a clean OS install and create the users, the scenario, and the history. This includes running services as well as creating files and log files with the appropriate historical entries.

2.2 Lab Execution Environment

The FALCON Lab Execution Environment (LEE) provides students with an interactive environment, allowing them to conduct a forensic investigation. Currently, we target live digital forensics investigations and use the OnLine Digital Forensic Suite™, a web-based forensic investigation tool created by ATC-NY. OnLineDFS, developed under the AFRL-funded project, "Mobile Platforms to Support Network Forensics" [6,7], is an extensible platform that contains tools for acquiring and analyzing data, focusing on the running state of a Unix or Windows computer. This data includes processes, network connections, and memory dumps, as well as file-based forensics used by traditional forensic tools. OnLineDFS is a robust tool that can be used "as is" to fulfill the requirements of the LEE, at least for live forensics investigations. We are also investigating incorporation of a "dead" forensics investigation tool, such as the Sleuthkit [1], Encase [2], FTK [3], or S.M.A.R.T [5].

Because OnLineDFS was designed to support law enforcement investigations, it creates an extensive audit log detailing all actions taken by investigators, ranging from logging in and out and acquiring data to viewing data and performing searches. These audit logs are used to assess the performance of the students.

2.3 Lab Evaluation Tool

The FALCON Lab Evaluation Tool (LET) takes the audit log file generated by the LEE and analyzes it for certain patterns, providing metrics for objective lab assessment. Output of the LET supplements the typical mechanisms for evaluation of students—students will still turn in a report, but the LET can perform additional analyses which can be used both in grading and in refining future versions of an assignment. For example, the LET might determine whether the student successfully found a hidden file or performed a detailed analysis of a specific process. This allows an instructor to determine how many assignment goals a student achieved. Similarly, there might be actions which should not occur in the log. For example, a scenario might involve a search warrant that limits the types of information the student can examine, such as email sent to or from a limited set of users. Any other user's email would be defined as "off limits" by the warrant and would therefore not be admissible in court. The LET allows an instructor to check that the student did not inadvertently examine mail from an "off limits" user.

Finally, the LET might be used to gather information to gauge the level of difficulty of the lab, such as the total amount of time the student spent on the lab, the number of sessions, and the average time spent per session. While time spent does not indicate the

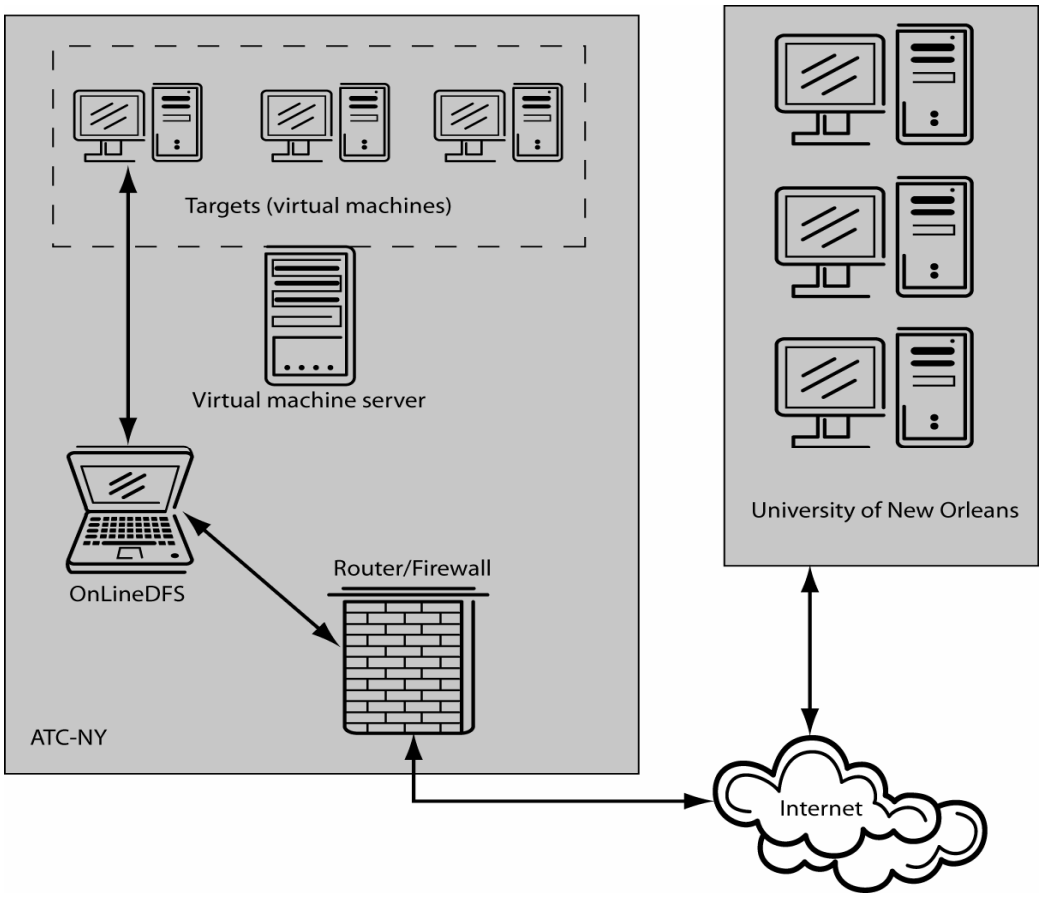

Figure 2: Network configuration in proof-of-concept experiment.

success of the lab, it does help determine whether the lab requires a reasonable amount of time to complete.

3 Preliminary Results

In order to test the feasibility of our approach to creating laboratory assignments for digital forensics instruction, we recently conducted an experiment, creating a laboratory exercise for students in a digital forensics course at the University of New Orleans. Figure 2 shows the network configuration used for the assignment.

The students (in the role of investigators) were located in New Orleans, LA. The target machines running the scenario, along with the OnLineDFS machine, were located in an isolated subnet, behind a firewall/router, in Ithaca, NY. The target machines were VMware virtual machines, run by a virtual machine server. UNO students were permitted access to the subnet only through the web interface of the OnLineDFS machine. The figure shows a simplified version of this network setup.

In the laboratory scenario, an employee of a company was running an illegal web server, disguised as a normal editor program. The images being served were located in another user's directory (as an attempt to direct suspicion to the other user), and the "bad guy" had a connection off-site to a password-protected web page at the content supplier's site. Each machine had ten user accounts; three users were actively logged on to each machine during the investigation.

The students were only told that the company, an ice cube company, was having problems with "penguin pornography," which was defined in the assignment as any depiction of a penguin without a hat. The assignment was meant to parallel a child pornography investigation, while keeping the tone light. The students were given 11 questions to answer about details of the case (who did it, what did they do, how did they do it, what is the supplier's site, what is the password to the site, etc.). The questions asked in the assignment appear below. **The complete assignment is available at http://www.cs.uno.edu/~golden/FALCON/assignment1.pdf.**

1. Who's the bad guy?
2. Is there penguin porn on the box?
3. If there's penguin porn, how is it being distributed?
4. What clever techniques, if any, were used to obscure the activities?
5. (Exhaustively) where on the box is the bad guy storing applications, data associated with their dark, evil crime, or the penguin porn itself?
6. Has the bad guy attempted to implicate anyone else who uses the target machine in the crime? If so, who?
7. What is the numeric IP address of the site of the bad guy's supplier?
8. Are there any password-protected pages on the supplier's site? If so, provide URLs, usernames and passwords.
9. What is the exact hostname of the site in question 7? Hint: virtual hosting is probably used, so a simple reverse lookup will not give you the correct answer.
10. What is the name of the supplier of penguin porn? Hint: Not the bad guy mentioned in Q1!
11. What species of penguin does the supplier not have pictures of? Does he say anywhere when he might have this type again?

The logs were monitored during the assignment to ensure that the students were on track and to provide rapid feedback that other investigative approaches might be needed to complete parts of the assignment. This included verifying that they had found the directory containing the "illegal" images and that they were looking at processes, memory dumps, and network port information. The audit logs were on the order of 1 megabyte in size per student team.

After the assignment was completed, the students and the authors had a debriefing session via a conference call to provide feedback. There were 12 students in the class. In terms of difficulty on a scale of 0-10, most students rated the lab a 5-6. In fact, all students were able to complete the assignment. Students reported spending 4-6 hours to complete "most" of the work, and then spending significant time, up to 20 hours total, to finish the assignment. They commented that there were not many dead-ends, which made the assignment easier than if they had to pursue many false leads.

For this proof of concept experiment, much of the work of the LCT was performed manually. All target machines were clones of each other, so the same user was the "bad guy" for all of the teams. It took almost a month to set up the exercise, and there were numerous items that were not included due to time limitations, such as mail messages, including junkmail and some real messages containing clues. In addition, the users not involved in the scenario had little history of activity (logins, .bash_history, file time-stamps, etc.). Since the assignment focused on live analysis, the lab was intended to discourage the students from simply sifting through all files and encourage them to examine live information such as running processes and open ports. This experiment validated our approach, but argues for a much more sophisticated LCT.

4 Conclusions and Future Work

To provide adequate training to students of digital forensics, both lectures and relevant lab exercises are needed. Unfortunately, most laboratory exercises are "toy"-

sized, because of the difficulty of creating full-blown targets for investigation. Even if an instructor is willing to invest the substantial time required to create detailed assignments, this is a tedious and error-prone job. Depositing digital evidence on a target, while providing sufficient "background noise" (in the form of actions performed by normal users) requires careful attention to timestamps, the order in which files are created, and which services are running on the target. Further, many slightly different targets may be required, in order to discourage students from sharing laboratory solutions.

In this paper we discussed an architecture for the creation, deployment, and evaluation of laboratory exercises to support digital forensics education, named FALCON. FALCON consists of components for creating forensic targets "to order", using a specification of user behaviors and events, for deploying an exercise, complete with logging of user actions, and auditing the user actions during an investigation, in order to improve future editions of the exercise. Since FALCON supports remote administration of laboratory exercises, it can decrease operational costs and make digital forensics education available to a wider audience.

Much work remains to be done on FALCON, particularly to further improve the Lab Creation Tool (LCT) component, but initial results are promising. Our first laboratory exercise was deployed at ATC-NY, in Ithaca, NY, for students at the University of New Orleans. Based on comments received from students after the assignment was completed (in early May 2005), we are refining the way assignments are specified in the LCT to automatically increase the amount of "irrelevant" evidence attributable to innocent users. While we currently rely on the OnlineDFS for implementation of the FALCON component for administering an exercise (and this largely limits us to live digital forensics investigations), we are also considering use of a traditional "dead" forensics tool, such as the Sleuthkit or a commercial product, in the LEE.

5 References

[1] Sleuthkit and Autopsy, http://www.sleuthkit.org.

[2] Encase forensics software, http://www.encase.com.

[3] Forensics Toolkit (FTK), http://www.accessdata.com.

[4] iLook Investigator forensics software, http://www.ilook-forensics.org/.

[5] SMART forensics software, http://www.asrdata.com/SMART/.

[6] F. Adelstein, "MFP: The Mobile Forensics Platform," Proceedings of the 2002 Digital Forensics Research Workshop, http://www.dfrws.org.

[7] F. Adelstein, "MFP: The Mobile Forensic Platform," International Journal of Digital Evidence, 2(1), 2003.

[8] E. Casey, Digital Evidence and Computer Crime, Academic Press, 2nd Edition, March 2004.

[9] E. Casey, Handbook of Computer Crime Investigation: Forensic Tools & Technology, Academic Press, October 2001, pp. 2-3.

[10] E. Casey, "Network traffic as a source of evidence: tool strengths, weaknesses, and future needs," Digital Investigation, 1(1), Elsevier, 2004, pp. 28-43.

[11] P. Sealey, "Remote forensics," Digital Investigation, 1(4), Elsevier, 2004, pp. 261-265.