Using Hacking to Teach Computer Science Fundamentals

TJ OConnor
Information Technology and Operations Center
US Military Academy
West Point, NY 10996

Ben Sangster
Department of Computer Science
US Military Academy
West Point, NY 10996

Erik Dean
Information Technology and Operations Center
US Military Academy
West Point, NY 10996

ABSTRACT

Hacking is a controversial topic among computer scientists. Hacking implies the illicit circumvention of methods, processes, or systems. However, the methodology employed by hackers inspires creativity and innovation. Masters of our discipline often refer to themselves as hackers, but computer science educators rarely embrace the idea of hacking in the classroom. Instead, we focus on marketing staid curricula when the same subjects can be taught with at least the same rigor and in a far more compelling manner. This paper explores countering enrollment decline via a curriculum that carefully exposes students to hacking as a means of teaching core computer science concepts, including algorithms, networks, compilers, programming languages, and operating systems. We examine how teaching computer science fundamentals under the guise of hacking can attract and motivate students. Further, we examine the results and lessons learned from building a hacking-based curriculum at West Point, including infrastructure and curricular design requirements as well as specific activities, speakers, lessons, and techniques we have employed. Certainly, implementing hacking in the classroom has challenges, yet our results indicated hacking is a valuable tool that can increase enrollments, foster lifetime learning and generate creative thinkers for our discipline.

Categories and Subject Descriptors

D.4.6 [**Operating Systems**]: [Security and Protection]; D.2.8 [**Software Engineering**]: Metrics—*complexity measures, performance measures*

General Terms

Security

Keywords

Education, Security, Information Technology, Curriculum

1. INTRODUCTION

Ten years ago, the Information Technology and Operations Center at West Point began integration of Information Assurance (IA) education into the computer science curriculum. [6] What began as a single isolated lab environment and a small computer security club has matured into the largest club activity at the US Military Academy, a myriad of course offerings that stress the importance of computer security, and recognition as a National Security Agency Information Assurance Center of Excellence.

Despite those accomplishments, there are still several challenges to teaching students computer science at West Point. One major hurdle is that students must complete a rigorous liberal arts education in addition to their computer science curriculum. This education model leaves students with only four semesters of dedicated study of computer science. This accelerated pace of learning can occasionally leave students disenfranchised. To combat this, we began implementing hacking courses as a tool to introduce the concepts in a subtle way.

The remainder of this paper will examine the background behind using hacking to teach computer science, the design and implementation of the hacking curriculum, and the future direction of the curriculum.

2. BACKGROUND

A student pursuing a degree in Computer Science must be able to apply and understand many fundamental concepts. First and foremost, a student must be able to think critically and solve problems using a well-structured programming language. Further, the student must understand fundamental concepts such as abstract data structures, complexity theory, formal notation, operating systems, networking, and a variety of programming language characteristics. [10] It is our hypothesis that an instructor can teach several of these fundamental concepts under the guise of hacking and information assurance, which has a slightly more attractive appeal for many students.

Bratus introduced using a hacker curriculum to teach computer networking at Dartmouth. [3] In his research, he examined the diverse sources of information hackers use for learning materials. This includes classic textbooks, electronic magazines such as Phrack, online forums, security talks and private communications at hacker conventions, source code from released tools, and various internet relay chat communities. [4] Such a broad spectrum of materials

forces hackers to examine concepts from non traditional approaches.

Hackers think and pursue knowledge quite differently than traditional academia. They routinely approach problems with a different lens that does not respect the approved solution or boundaries of a given protocol. In their academic pursuit they look for ways to exploit or bend an accepted standard into a new truth or understanding of a problem. This alternate angle and lens of view can prove exciting for students in the domain of computer science. Additionally, academia routinely adopts hacker-inspired inventions or academic creations. Hackers simply have different psychological traits. Conti noted why computer scientists should embrace this psychology. [5]

An example of such a unique psychology involves the method of fuzzing software. For several years in the late 1990s, two hackers named Mudge and Hobbit routinely found security compromises in Microsoft products. When asked by Microsoft engineers how the pair was able to so easily compromise the Microsoft product, Mudge explained that he fuzzed or inputted random data into the products trying to crash them. The engineers explained they were astonished; the idea of trying to crash a product had never entered their process or thoughts. [11]

Locasto at George Mason is one instructor who has embraced this psychology in the classroom. [9] He routinely forces his students to write exploit code. In crafting exploits, his students learn to appreciate the alternate approach taken by hackers. We embrace the ideas of Bratus, Mudge, and Locasto at the United States Military Academy. In our curriculum, we very carefully expose students to hacking in an effort to teach core concepts.

3. DESIGN

The goal for our students is to teach them the knowledge, skills, and abilities to think critically in the domain of computer science. To excel at computer science requires an ability to understand and apply specific concepts such as compiler theory, regular expressions, networking fundamentals, and object oriented design. Thus, our department has courses in all of the preceding topics. However, we argue that we can teach the basics behind many of these topics by disguising them as hacking instead. The following sections outline a couple of methods for specific examples to understand this design.

3.1 Languages for Intrusion Detection

Traditionally in computer science, we teach students how to parse statements such as A+B * C using different language parsing techniques. An LL1 parser will parse the preceding statement quite differently than an LR1 parser. To augment the traditional approach, we teach languages in our Cyber Warfare class by introducing the concept of intrusion detection systems. An intrusion detection system (IDS) monitors computer networks and reports incidents based on a set of user-defined rules and configurations. A properly configured IDS can assist a network administrator to identify attacks and respond to threats posed by network adversaries. The science and expertise in IDS deployment involves the proper configuration of the rules, where each rule must obey a defined grammar and language. However, even grammatically correct rules can present a problem if they work inefficiently, exhausting our resources to detect attacks. Examining the data content of a network packet proves costly expensive since it requires the use of a matching algorithm such as Boyer-Moore. However, reading header detail costs significantly less since the header has fixed parameters.

```
alert tcp any any -> 192.168.1.0/24 143 (content:
"|90C8 COFF FFFF|/bin/sh"; msg: "IMAP buffer overflow!";)
```

Figure 1: An intrusion detection system rule.

To be able to write efficient rules, students must understand how the specific IDS configuration language implements recursion and grammars and match header details prior to any packet inspection. Figure 3 depicts a rule from the SNORT intrusion detection system that we use in class. In this example, the students write a rule that matches specific data content directed at the IMAP port of a mail server. We find that teaching students how to parse languages in terms of intrusion detection can assist such traditional methods as building or evaluating a compiler.

3.2 File System Forensics

Most operating systems classes in computer science examine how computers store and organize data in file systems. For this traditional approach, it is important students understand how a file allocation table allocates blocks or clusters on disk to the operating system to write data to disk. Additionally, it is important for students to understand what happens to these blocks and the file allocation table when a user deletes a file.

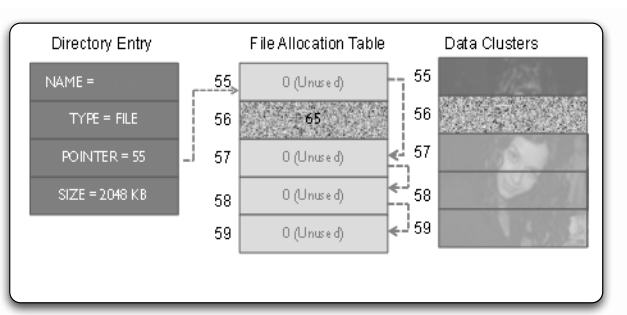

Figure 2: File reconstruction using elements of a file system unallocated, allocation table, and directory entry.

Arguably, file systems are a mundane topic but important to the overall understanding how an operating system works. In our Digital Forensics class, we present the idea of understanding file systems in order to recover digital artifacts from deleted data. To increase student involvement and interest in the material, we provide the students with several hard drives with deleted data. Then, we show the students how to recover the data by examining the unallocated space of the hard drive and using the residual metadata in the file allocation table. Figure 2 depicts an image recovered using this methodology. Creating challenges and exercises in data

recovery further helps reinforce the material and student interest in an otherwise mundane topic. Similarly, a number of courses use competitive techniques to motivate students.

3.3 Software Exploitation

Arguably programming and understanding assembly language can be one of the most difficult topics in the field of computer science for an undergraduate student. Understanding how and which variables and functions are placed onto the stack and heap can prove challenging to comprehend. Further, understanding memory allocation can frustrate students. Typically, this level of instruction occurs in a compiler theory course or even a separate level course on assembly language.

```
$ (perl -e 'print "\x90"x147 .
"\xeb\x1f\x5e\x89\x76\x08\x31\xc0\x88\x46\x07\x89\x46
 \x0c\xb0\x0b\x89\xf3\x8d\x4e\x08\x8d\x56\x0c\xcd\x80
\x31\xdb\x89\xd8\x40\xcd\x80\xe8 \xdc\xff\xff\xff/bin/sh"
."\x99\xfc\xff\xbf"x8âĂŹ; cat -) | ./gameover
```

Figure 3: A software exploit created to take advantage of poorly validated input.

To augment the student's understanding of these concepts, we introduce the idea of writing software exploits that take advantage poorly defined variables in simple programs. Not only does this solidify the student's understanding of stack memory allocation and assembly language, but it reinforces critical elements of software design such as input validation and code review.

4. IMPLEMENTATION

In implementing a curriculum that carefully exposes students to hacking concepts, there are several constraints. Hacking is not a core competency we teach students. Rather it is simply a means to make computer science topics attractive to students. We understand this method may not work for all students. Thus, these courses must be optional or elective for the students. We also understand that students may wish to pursue hacking as an independent research topic after developing an interest in one of the elective courses. Finally, we hope to inspire not only computer scientists but several students from outside academic domains. To make all this possible, we use a three pronged approach including elective courses, independent research topics, and extracurricular clubs.

4.1 Courses

Some instructors from our core competency program have adopted similar approaches to teaching challenging or mundane computer science concepts. However, there are three main courses where we use this methodology exclusively to teach computer concepts under the guise of hacking. All courses involved in our approach are elective courses that are not required to complete a degree within the department. However, a majority of our computer science majors elect to take these courses. Additionally, majors outside the domain of computer science and electrical engineering routinely enroll in these courses as well.

- IT460 Politics, Strategy and Tactics of Info Warfare
- CS485 Digital Forensics Investigations
- CS482 Information Assurance and Cyber Warfare.

IT460 provides students with a basic and conceptual understanding of Information Warfare. This course takes an approach to understanding the political, economical, and military applications of information warfare. In the capstone project, students must demonstrate a broad understanding of information warfare in a practical exercise as it applies to the revolution in security and military affairs confronting our nation.

CS485F Digital Forensics focuses on acquisition and analysis of evidence from computer systems. The course teaches a scientific methodology to examine evidence from volatile and persistent data sources. Three course projects revolve around students selecting and creating tools to solve a problem presented in their forensic investigation. In the culminating event, the students perform a full forensic investigation into systems compromised by the National Security Agency.

CS482 Information Assurance focuses on security of operating systems, information assurance , system and network security, offensive and defensive information operations. A course project and term paper bring together the diverse concepts learned in CS482. In the culminating exercise, the students design and implement defensive measures to protect a production network from attack by a red team from National Security Agency.

4.2 Independent Research

Independent research projects have also provided an excellent venue to expose the students to hacking concepts.

Figure 5: Streaming video feed from hacking unmanned aerial vehicle built by Michael Weigand.

The last few years our students have spoken at hacker conferences such as ShmooCon and DEFCON. This has proven an incredibly valuable resource in motivating students to excel on independent and capstone level research. The audiences at hacker conferences can often be less forgiving than traditional academic conferences. For instance, at ShmooCon, the conference presenters give each audience member a rubber ball to throw at speakers when they wish to challenge

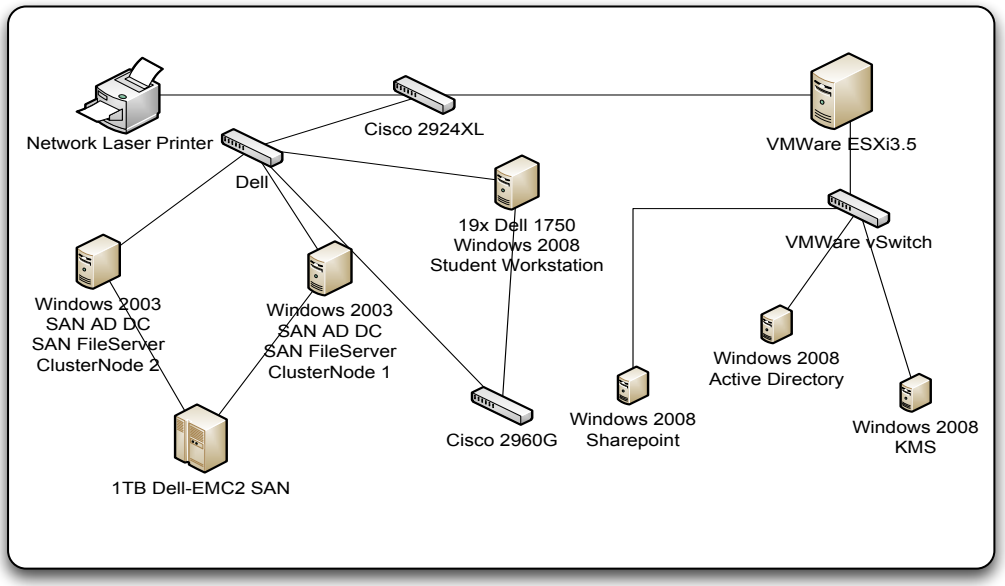

Figure 4: The Information Warfare and Analysis Lab (IWAR) provides students with an isolated, air-gapped network to research hacking in a secure, safe, modular environment.

a speaker's hypothesis or evidence. [2] Such pressure forces the students to diligently prepare their research and supporting evidence for their hypotheses. Our students have spoken on topics such as mapping binary files, creating mobile device honeypots, third-party privacy tracking on web browser sessions, and creating inexpensive autonomous vehicles capable of hacking. Figure 5 depicts the results of some of this outstanding work, showing a realtime video stream from student's unmanned aerial vehicle.

During the summer months, our students participate in academic individual development at several corporate and government agencies. Most notably, we send as many as twenty students to research independent projects at the National Security Agency (NSA). The NSA serves an incredibly dynamic environment to enforce the concepts already learned by the students as well as introducing new areas of research for the students. The assignments occur before the student's junior or senior year at college and often result in students bringing back topics to research in a culminating capstone project during their senior year.

4.3 Extracurricular Clubs

Our security club, The Special Interest Group on Security Audit Control (SIGSAC) provides one vehicle to introduce potential recruits to the methodology behind our hacker curriculum. SIGSAC affiliates with an Association for Computer Machinery (ACM) chapter that addresses all activities involved with maintaining and protecting computers and their programs, focusing on the architectural foundation of secure systems. [1] [7] Club activities revolve around hacker related activities such as cyber warfare competitions, expert presentations of emerging threats, and travel to hacker conferences. We have received an overwhelming positive response from students for the aforementioned activities. Speakers such as Bruce Potter (Shmoo Group), Mike Kershaw (Kismet Wireless), Joe Grand (Hardware Hacking), and Dan Kaminsky (Technical Advisor on DNS Security) have spoken to the club about cutting edge security concepts. Other student activities include:

- Google 101 - A lecture on using google to perform passive reconnassaince against network targets proves invaluable in introducing students to our methodology.
- Hacking 101 - An introductory lecture on security mechanisms and the methods used by hackers to break into systems. Students typically enjoy learning about the vulnerabilities of most computer systems.
- Wireless Penetration Test - An exercise that passively maps the security of wireless and mobile devices on the campus.
- Cyber Defense Exercise - The culminating event for our CS482 course, where students challenge other universities in competition to defend a network from attack by the National Security Agency. However, students outside of the course participate in a myriad of roles.
- National Security Agency Trip - A trip that exposes students to emerging technologies used by the National Security Agency.
- ShmooCon, DEFCON, Blackhat Conference - Trips to hacker conferences to both attend and speak at have proven invaluable to motivating students.

These exercises, lectures, and trips prove valuable for not only the students but for building the security of our campus as well. For example, the google reconnaissance exercise

discovered a significant vulnerability on the university network that leaked partial social security numbers of faculty. When operating on the university's official network, we coordinate all tests and exercises with the appropriate network authorities well in advance. For those activities that we cannot conduct legally on our networks, we have a separate isolated network and research facility.

4.4 Research Facilities

The Information Warfare and Analysis Lab (IWAR) provides us with an excellent venue for a hacking curriculum. [8] Figure 4 depicts one of the multiple labs in the IWAR. The completely isolated, air-gapped lab hosts five separate domains, ranging from completely open source operating systems such as Linux and FreeBSD to Microsoft Windows 2003 and 2008 Active Directory domains. Each domain provides a myriad of services including email, web, voice over ip, and instant messaging. The isolated environment allows our students to test information assurance skills defending and penetrating machines. Each student workstation runs on a virtual platform. Using VMWare Workstation images, our students are able to quickly return a compromised or disabled machine to an original functioning state. Additionally, the storage area networking (SAN) devices allow us to quickly deploy new workstation images for the students. In courses such as Digital Forensics, we can rapidly deploy a workstation image of a machine for students to investigate. For example, an instructor can deploy a 4GB VMware workstation image with ease in the twenty minutes prior to class. The lab hosts virtual workstation images of several popular open source penetration and forensics tools such as BackTrack Penetration Testing and the DEFT Forensic Toolkit. However, the lab also has educational licenses for commercial products such as Core Impact for penetration testing and EnCase for forensic investigations.

5. ANALYSIS

We have worked tremendously hard to implement hacking into the curriculum. But does it work? Countering enrollment decline while providing a level of excitement for students in the domain of computer science are our main goals for carefully implementing hacking in our curriculum. At this point we have preliminary evidence to support our hypothesis. In the previous year, over double the amount of students than previous years chose to pursue a degree in the domain of computer science. As faculty members, we carefully exposed a large majority of these majors to different events such as the cyber defense exercise and SIGSAC club, which reflect the hacking methodology used in our courses. Our informal interviews with the new computer science majors reflected the same findings. Overwhelmingly, they responded well to the introduction of hacking into the curriculum. While this appears only a single data point, it does reflect the large amount of effort to accomplish by junior faculty.

It is important to note that we have encountered resistance in our efforts. The idea of hacking revolves around innovation and creativity, breaking the protocols and assumptions. Thus, the curriculum must be dynamic. Recreating completely different lessons for different semesters can prove challenging and cumbersome on the instructor. Further, some faculty disagree with our hypotheses that hacking in the classroom can work. Typically, they argue one of two major points. First, they complain that we have not thought enough about the ethical considerations of teaching hacking. Second, they argue that teaching hacking is paramount to giving the students a list of tools. We address both these issues. First, we provide constant reinforcement of the ethical dilemma presented in hacking classes. Events or classes typically begin with some identification of the moral events surrounding the concept as well as the applicable laws that enforce those concepts. We follow this with several personal conversations with cadets as they continue research into grey areas of computer science. Second, we do not teach tools. Rather, we expose the students to the concepts involved with breaking a protocol well in advance of any particular tool. Implementing tools serves as only a method to reinforce the concept the same way another instructor might use Microsoft Access as a tool to reinforce the concepts in a database class.

6. CONCLUSIONS AND FUTURE WORK

This paper presents the ideas of carefully exposing students to hacking as a means to teach computer science fundamentals. We argue that hacking in the classroom can assist to counter declining enrollment in a computer science degree by making the material more attractive to students. However, this is simply a work in progress. Initial metrics and informal interviews with students support our hypothesis. However, we must continue to evaluate these metrics over a much longer period of time before we can make any definitive assertions about our work.

7. REFERENCES

[1] Acm sigsac. http://www.sigsac.org/, Feb 2010.

[2] Shmoocon. www.shmoocon.org, 2010 February.

[3] S. Bratus. Hacker curriculum : How hackers learn networking. *IEEE Distributed Systems Online*, 8, 2007.

[4] S. Bratus. What hackers learn that the rest of us don't: Notes on hacker curriculum. *IEEE Security and Privacy*, 5:72–75, 2007.

[5] G. Conti. Why computer scientists should attend hacker conferences. *Commun. ACM*, 48(3):23–24, 2005.

[6] G. Conti. Hacking and innovation. *Communications of the ACM*, 49(5), June 2006.

[7] G. Conti, J. Hill, S. Lathrop, K. Alford, and D. Ragsdale. A comprehensive undergraduate information assurance program. pages 243–260, 2003.

[8] S. D. Lathrop, G. J. Conti, and D. J. Ragsdale. Information warfare in the trenches. pages 19–39, 2003.

[9] M. E. Locasto. Helping students 0wn their own code. *IEEE Security and Privacy*, 7:53–56, 2009.

[10] B. Mace. Software engineering with the context of a computer science programme. *Software Engineering Journal*, 4:200–202, 1989.

[11] A. Oram and J. Vega, editors. *Beautiful Security*. OReillly, Sebastopol, CA, first edition, April 2009.