Viruses 101

J. Aycock and K. Barker
Department of Computer Science
University of Calgary
2500 University Drive N.W.
Calgary, Alberta, Canada T2N 1N4
{aycock,barker}@cpsc.ucalgary.ca

ABSTRACT

The University of Calgary introduced a controversial course in the fall of 2003 on computer viruses and malware. The primary objection about this course from the anti-virus community was that students were being taught how to create viruses in addition to defending against them. Unfortunately, the reaction to our course was based on a dearth of information, which we remedy in this paper by describing key pedagogical elements of the course.

Specifically, we present four aspects of our course: how students are vetted for entry, operation of the course, course content, and the instructional materials used. In addition, we pay particular attention to the controversial course assignments, discussing the assignments and the need for balance, objectivity, security, and learning in a university environment. Our experiences with the course and future plans may be helpful for other institutions considering such course offerings. It should also provide opponents of the course with valuable information about the true nature of the course, the pedagogy used, and the value provided to the computer community as computer science graduates with this kind of expertise take their place as the next generation computer security experts.

Categories and Subject Descriptors

D.4.6 [**Operating Systems**]: Security and Protection—*Invasive Software*; C.2.0 [**Computer-Communication Networks**]: General—*Security and Protection*; K.6.5 [**Management of Computing and Information Systems**]: Security and Protection—*Invasive Software*

General Terms

Security

Keywords

Computer viruses, Anti-virus software, Malware, University course

1. INTRODUCTION

It is hard to think of any truly controversial topics in computer science. It is even harder to think of any that are far-ranging beyond computer science: not "religious wars" about a favorite editor or programming language, but topics which incite passion and opinion in the general public.

We stumbled across just such a topic when the University of Calgary introduced a course in computer viruses and malware in the fall of 2003. One distinguishing feature of our course is that students learn about malware by creating their own under strictly controlled conditions. Prior to our announcement of the course, we were aware of only one other institution that did this [6]; since that time, we have found out about a few others. To the best of our knowledge, we are currently the only institution in Canada that offers such a course, and only one of a handful worldwide. Typically, any discussion of computer viruses seems to occur in the abstract, as a small part of some other "legitimate" course offering like operating systems or networking.

The reaction to our virus course announcement astounded us. We were inundated with email, swamped with media attention from all over the world (e.g., [5, 15, 19, 20]), and castigated by the anti-virus community (e.g., [4, 13]).

Much of this reaction took place in the absence of any concrete details about what we would be teaching. In the remainder of this paper, we present these details. Starting with high-level course design issues in Section 2, we move on to course admission and operation in Section 3. Section 4 talks about the course syllabus and material, followed by the assignments in Section 5 and our secure laboratory environment in Section 6. We conclude with our future plans for the course.

2. THREE BIG QUESTIONS

Generally speaking, there are three questions that need to be asked when teaching a course:

1. How do we teach this subject?
2. Can we teach this subject safely?
3. Would teaching the subject make the world worse or better?

Normally, course content is so clearly benign that we do not even consider Questions 2 and 3, but the questions are implicitly present and must be considered as part of an ethical instructional design.

We addressed Question 2, safety, by designing a secure environment for students to work in (see Section 5). Question 3 is somewhat trickier; the full argument is outside the scope of this paper, but it can be summarized as follows. First, malware is a valid area of study, and such computer security research is becoming vital to our increasingly computer-dependent society. Second, it is very easy to learn how to create malware, even for people with no programming expertise. It is not easy, however, to learn this in a safe environment, nor to get an objective view of the entire field, both malware and anti-malware.

Surprisingly, our answer to Question 1 was the thing that caused us the most grief. We took a pedagogical approach to the virus course not unlike that we would use for other courses, trying to ensure that students would learn in the most effective way possible. To this end, many educators would recognize the quote:

> I hear, and I forget.
> I see, and I remember.
> I do, and I understand.
>
> – Anonymous

We wanted to produce high-quality students who possess a deep understanding of this aspect of computer security. We therefore wanted students to "do," which in the context of malware, means that students must write both virus and anti-virus software. However, writing viruses, even if they are not new types of viruses, earned us instant, stiff opposition from the anti-virus community. If we had omitted the virus-writing aspect of the course, we were told that the anti-virus community would have supported our initiative. But we did not back away from this pedagogical principle, because doing so would have compromised our ability to teach students objectively and effectively. Our students, informally polled at the end of the course, confirmed that they thought they learned the material better through doing.

3. COURSE ADMISSION AND OPERATION

Enrollment in the virus course is limited to 16 students – both graduate and undergraduate – due to lab space limitations. The 16 spots are assigned competitively; students must meet the following criteria:

- A grade point average of 3.0 or better.
- Computer Science students only. This restriction allows us to properly enforce penalties for lab protocol violations, should it become necessary to do so.
- Fourth-year (senior) standing or better, which includes graduate students.
- A passing grade in operating systems and computability theory courses. Operating systems background is obviously essential, and this prerequisite also transitively ensures that students have taken assembly language courses. The key elements from the computability theory course are finite automata and undecidability results. Depending on the content covered in the virus class, a computer networking course would also be useful.
- A one-page essay, on why the students wants to take the course, and what learning expectations they have from it.[1] The essays are ranked by committee, and students who meet the other criteria are offered positions in the class using this ordering.

Noteworthy in its absence is the lack of any criminal background checks or other security checks. It was decided at higher levels of administration that such a requirement would run contrary to the mission of a university.

Students are notified in advance that they will have to sign a legal agreement to take the course and abide by strict laboratory protocols. In the interest of full disclosure, applicants are also notified that some companies have made public statements that they will not hire people who have taken the virus course [18]. A discussion of this ban is beyond the scope of this paper, but we have noticed that it has unfortunately dissuaded some students from taking the course.

No auditing or "sitting in" the lectures is allowed by students not admitted to the course, and this policy is enforced by the instructor checking students' identification.

4. SYLLABUS AND COURSE MATERIAL

As a "capstone" course, studying malware combines ideas from across computer science as well as other disciplines: low-level concepts, operating systems, programming language implementation, networking, security, automata theory, law, ethics, psychology, and human-computer interfaces.

Our current syllabus is below. Law and ethics are covered first, deliberately, before any programming assignments are done; this is part of establishing a secure environment. The material on "weaknesses exploited" is actually more closely related to worms, but was moved up to accommodate a guest lecture on the topic.

- Introduction
  - Lab protocol
  - Legal agreement
  - Basic definitions
- Law and malware
  - United States law
  - Canadian law
  - Council of Europe Convention on Cybercrime
  - Extradition
- Ethics and malware
  - General ethical theories
  - Moral development
  - Ethical decision-making processes
  - Codes of ethics
    - Computing profession
    - Anti-virus profession
- Weaknesses exploited

[1]The admission process for graduate students involves a personal interview instead of the essay.

  - Software
  - Humans
    - Social engineering
    - Hoaxes
- Viruses
- Anti-virus techniques
- Anti-anti-virus techniques
- Worms and deworming
- People and communities
  - Malware creators
  - AV community
- "Applications"
  - Benevolent Malware
  - Information Warfare
  - Cyberterrorism

We are careful to maintain objectivity when presenting material about virus writers. It is not unusual, at anti-virus conferences, to hear virus writers referred to in less-than-flattering terms, but such rhetoric has no place in an objective university environment. We also rely on established research when studying virus writers [8, 9], and not stereotypes [16].

Objectivity plays a strong role when teaching ethics, too. Clearly, we do not want students who think it ethically sound to deliberately unleash malware into the world, yet it is not the job of a university instructor to indoctrinate students with a certain set of ethics. That would neither be objective, nor would it allow students to be inquisitive. We must instead give students an appropriate framework with which to make choices: teaching ethical theories, working through case studies, and emphasizing the effect of actions on others.

Where possible, the material in the syllabus is complemented by guest lectures from domain experts. In our first offering of the virus course, we had three guests, who spoke about criminal law, ethics, and buffer overflow defenses in the OpenBSD operating system.

Finding course material is an ongoing problem. We do not use a textbook at present, because no text exists that covers the correct topics at the appropriate level of detail for a senior/graduate course. Some material is quite easy to find, like computer-related ethics [2]; computer law is similarly easy, although many available resources are U.S.-centric. There are an increasing number of recent books on exploiting software weaknesses [7, 12] and worms [14], too.

Virus material is prolific, but surprisingly, the single hardest area to find detailed information about is anti-virus software. Current books [10, 17] only gloss over some anti-virus techniques, and anti-virus companies treat this information as proprietary. Obviously, anti-virus companies do not want to give anyone a competitive advantage, but neither do they want to leak information to virus writers.[2] For anti-virus material, traditional sources of information are insufficient.

[2]Deciding whether or not security through obscurity is effective is left as an exercise for the interested reader.

As researchers, we have been taught to regard non-peer-reviewed, non-academic publications with extreme suspicion. Virus writers do not tend to publish in *Nature*, however. To dismiss non-academic publications is to disregard a wealth of potential information about both virus and anti-virus software. Preparing course material for our virus course thus took months of painstaking research work, using methods not unlike historians might use: taking bits and pieces from a wide selection of non-academic sources and piecing them together to arrive at some semblance of the truth. Some material unearthed this way, particularly technical detail, is directly verifiable; other material must be qualified appropriately when presented in lectures:

> "Several writers have said that..."
>
> "One author has said that..."

In lectures, this is a good opportunity to teach analyzing the veracity of source material, and to encourage a healthy level of skepticism.

Sometimes non-academic sources are the simplest ways to get certain information. In one case, the instructor was looking for a specific macro virus' source code to use in a lecture. "Legitimate" sources presented the source code in pseudocode form, if at all. In less than five minutes with a web search engine, the source code was acquired from a non-academic source.

5. ASSIGNMENTS

The virus course has five assignments: one written, four involving programming. The first, written, assignment is a malware-related ethical case study in which students must consider their response to a presented scenario. The programming assignments are grouped, reflecting the balanced nature of the course, with one "offensive" and one "defensive" assignment per group:

- Software weaknesses and defense

  Assignment 2. Basic stack-smashing attack.

  Assignment 3. Defense against stack-smashing attacks.

- Virus and anti-virus

  Assignment 4. Virus creation.

  Assignment 5. Anti-virus software to detect and disinfect.

After each offensive assignment of a group (i.e., Assignments 2 and 4), students share their code with one another. This gives them a larger set of samples to work with for their defensive assignment. For example, students had to write one virus for Assignment 4, then for Assignment 5 had to detect *all* viruses from Assignment 4 and disinfect at least one Assignment 4 virus.

The degree of difficulty for programming assignments is somewhat problematic. Obviously, we do not want to set trivial assignments. On the other hand, if assignments are too hard, an argument could be made – rightly or wrongly – that we had forced students to work outside the secure environment to complete the assignments. We have thus scaled back the assignments somewhat from what we would normally use in a course. For example, allowing students to exploit code of their own for Assignment 2, instead of finding holes in existing code, simplifies their task.

Another constraint with assignment difficulty is bureaucratic. At the University of Calgary, new courses such as the virus course aren't permitted any lab time, just lecture time. Any time spent by the instructor in the lab for instruction or assignment demonstrations detracts from lecture time. This is also a consideration when choosing the operating system used for assignments. Our students use mostly non-Windows operating systems, and so to have students doing assignments under Windows (for instance) would first require labs to instruct them on the finer points of Windows programming – not an effective use of lecture time. This is a temporary problem, however, and the virus course should have real lab time within the year.

6. THE VIRUS LABORATORY

Programming assignments are performed in a secure virus laboratory. This laboratory is one aspect of the secure environment which we have established [1]. We recognize that, by themselves, technical safeguards are insufficient; it only takes one student to work on malware outside a "secure" laboratory to render technical precautions moot.

Instead, we create a secure *environment* by employing five broad categories of safeguards:

Legal. Our legal safeguards are twofold. First, we teach the legal repercussions of writing and releasing malware through course material and a guest lecture.

Second, we impose contractual legal obligations on the students by having them sign a legal agreement prior to taking the course. The legal agreement includes adherence to the laboratory protocol, usage and handling of the course material, and liability and indemnity.

Ethical. As we mentioned in Section 4, teaching ethics with respect to malware is tricky. Again, we use a combination of course material and a guest lecture. We "test" ethics by working through case studies in lectures, and using a written ethics assignment.

Social. Peer pressure is exploited for safety conformance. Programming assignments *must* be done by students in pairs – in fact, students cannot log in to the virus lab machines unless the passwords of all group members are entered. We stress to the students that they are jointly responsible for everything that happens during their login session, and this is reinforced through the laboratory protocol.

Advocates of pair programming [3] might argue that this technique improves safety by improving software quality. It certainly adds an extra layer of security to the virus laboratory, because any illegal or unethical activity would require collusion between students.

Behavioral. Behavioral safeguards refer to conduct in the virus laboratory, which we regulate with a formal laboratory protocol. Recognizing that other sciences have a lot of experience handling dangerous substances, we initially based our protocol on biohazard protocols [11], adapting and extending the protocol to cover computer-specific risks.

Our protocol covers issues like laboratory entry, authorized personnel in the laboratory, prohibited devices and media, and safety mechanisms that must be included in software. The complete protocol may be found in [1, Appendix]. Violating the lab protocol results in an "F" grade in the course.

It is critical to note that the laboratory protocol applies to *everyone*, not just the students: the course instructor and technical staff are not exempt, for instance. This helps to underscore the importance of proper laboratory behavior.

Technical. Our technical safeguards can be divided into two parts: physical and electronic security.

Physical security. Physically, the laboratory is located within two card-key access areas. It has one entrance, with a door closer and an alarm which rings if the door is held open for too long. Two unmonitored cameras are present in the room, sending images to an external machine for recording. The machines in the laboratory are physically locked down, and the laboratory server and network switch are located in a locked cabinet. Network connectors in the room have been both physically and electronically disabled.

Electronic security. The laboratory computers have all unnecessary I/O ports disabled and the BIOS settings locked down. Ports on the network switch are locked to the MAC addresses of these computers.

Once logged in, students only have the option of running VMware, a virtual x86 machine – all work is done inside this virtual machine. The hardware architecture used for the laboratory server is different than that of the students' computers, and different operating systems are used for the server, to run VMware on, and to run inside VMware. The lack of hardware and software monocultures is deliberate, and is intended to limit the spread of any malware that should somehow escape its virtual x86 machine.

It is the combination of these five categories of safeguards, taking both human and computer elements into account, that makes the environment secure.

The safeguards are mutually reinforcing. Consider malware leaving the laboratory on a USB flash RAM drive, for example. This scenario is guarded against by technical, behavioral, legal, and social means. First, the USB ports have been disabled as part of electronic security. Second, the USB port settings cannot be altered without breaching further electronic security – a BIOS password – or the physical security of the locked-down computer case. Third, the laboratory protocol precludes such a USB device from being brought into the lab in the first place. Fourth, the legal agreement enforces adherence to the laboratory protocol. Fifth, even if the USB port were enabled somehow, collusion would be required to log in to the computer to transfer files. Sixth, any unauthorized personnel would have to compromise physical security (and be captured on camera) to get in to the laboratory. It is far easier to find malicious code using a web search engine than to take such elaborate measures!

7. FUTURE PLANS

We will be enhancing the course material in upcoming offerings of the virus course. In particular, we will be expanding coverage of worms, adding a section on virus cryptanalysis, and incorporating new information from both academic and non-academic sources.

There are a number of laboratory enhancements to consider too. The computing environment needs tuning to permit environment customization and multiple operating systems. There is also the question of whether or not to actively jam wireless electronics, but as the current safeguards prevent the use of wireless devices, this may be overkill. As for the human element, we are interested in finding ways to evaluate the efficacy of our non-technical safeguards.

Needless to say, we are committed to continuing to offer the virus course, despite the controversy. The students taking our course receive an in-depth understanding of malware and anti-malware, taught in a balanced, objective, and secure fashion.

8. ACKNOWLEDGMENTS

Many thanks to our technical staff, especially Tim Bliek, Darcy Grant, Shaun Laing, Brian Scowcroft, Jennifer Walker, and Erik Williamson. A number of people reviewed our lab specification and lab protocol to help ensure safety; Randy Abrams, Brad Arlt, Fred Cohen, Sarah Gordon, and Darcy Grant all provided valuable feedback. Thanks also to Gerry Bliss and Sid Tolchinsky from SPIE for their independent review and risk assessment. Stephen Jenuth, Ken Chapman, and Theo de Raadt gave the guest lectures on law, ethics, and OpenBSD, respectively.

9. REFERENCES

[1] J. Aycock and K. Barker. Creating a secure virus laboratory. In *13th Annual EICAR Conference*, 2004. 13pp.

[2] S. Baase. *A Gift of Fire*. Prentice Hall, 2nd edition, 2003.

[3] K. Beck. *Extreme Programming Explained*. Addison-Wesley, 2000.

[4] V. Bontchev. Should we teach virus writing? In *6th Association of anti Virus Asia Researchers Conference (AVAR 2003)*, 2003. 15pp.

[5] CNN Headline News: Hot Wired. News broadcast, 28 May 2004.

[6] F. Cohen. CJ 528/628 – Computer viruses and malicious code. http://unhca.com/CJ628/CJ628.html. Course syllabus, University of New Haven.

[7] J. Erickson. *Hacking: The Art of Exploitation*. No Starch Press, 2003.

[8] S. Gordon. The generic virus writer. In *4th International Virus Bulletin Conference*, 1994.

[9] S. Gordon. The generic virus writer II. In *6th International Virus Bulletin Conference*, 1996.

[10] D. Harley, R. Slade, and U. E. Gattiker. *Viruses Revealed*. Osbourne, 2001.

[11] Health Canada. *The Laboratory Biosafety Guidelines*. M. Best and M. Heisz, eds., 3rd (draft) edition, 2001. http://www.hc-sc.gc.ca/pphb-dgspsp/ols-bsl/lbg-ldmbl/index.html.

[12] J. Koziol, D. Litchfield, D. Aitel, C. Anley, S. Eren, N. Mehta, and R. Hassell. *The Shellcoder's Handbook*. Wiley, 2004.

[13] J. Kuo. Alberta strikes again. *Virus Bulletin*, page 2, July 2003.

[14] J. Nazario. *Defense and Detection Strategies against Internet Worms*. Artech House, 2004.

[15] B. Read. How to write a computer virus, for college credit (cover story). *Chronicle of Higher Education*, 50(19):A33, 2pp., Jan. 2004.

[16] Reuters. Looking into the mind of a virus writer. CNN.com, 19 March 2003.

[17] E. Skoudis. *Malware: Fighting Malicious Code*. Prentice Hall, 2004.

[18] Sophos. Sophos CEO says: "I won't hire virus writing students". http://www.sophos.com/virusinfo/articles/calgary2.html, 28 May 2003.

[19] P. Svensson. Antivirus industry steamed over virus article, college class. Associated Press (New York), 11 June 2003.

[20] E. Wilson. Anger at 'virus' lessons. Sydney Morning Herald (Sydney, Australia), 6 October 2003.