A Flexible Framework for Cyber Security Education

Richard Weiss, The Evergreen State College, Olympia, WA
Michael E. Locasto, The University of Calgary, Calgary, AB
Jens Mache, Lewis and Clark College, Portland, OR

Abstract

One of the primary goals of the EDURange project is to introduce students to security principles in the context of hands-on exercises and the security mindset, which allows students to see how software can fail and be made to fail. We have developed a framework and a set of exercises that we have used in the classroom and we have been refining them over the last few years. These exercises were designed with four goals in mind:
1. Flexibility to use simple scripts to specify exercises at a high level and create variations. Many of the exercises written by others were good but not exactly what we wanted, and they were not easily modified. Flexibility supports re-use.
2. Ease-of-use for faculty, which includes providing easy access to exercises, making them easy to create (not requiring configuration of VMs manually), easy to modify, and easy to share.
3. Emphasis on analysis skills, the security mindset, and the CS2013 guidelines.
4. Engagement of students, based on our experience with cyber security competitions.

The three main features of our approach are:
1. We take advantage of the elasticity of cloud computing. It can accommodate bursty student activity around high-demand times such as the end of term.
2. We developed a domain-specific language for scenario description, that automates the creation of exercises from a high-level description of the scenario.
3. We have built in assessment features, including a rudimentary scoring engine and the ability to capture traces of student activity.

We have conducted surveys with students and faculty, and these have been used to improve the framework and the content. The surveys also show that our exercises are as engaging as other cyber security exercises. We have begun a new phase where we focus on student assessment. We have collected data with the goal of recognizing patterns that indicate misconceptions that students have with the goal of providing instructors with information so they can provide relevant feedback to students. This is a feature we have not seen in other cyber security exercises and frameworks, and we expect that this will be an incentive for faculty to put in the small amount effort to try the framework.

The exercises have been used in over six classes at four schools with over 150 students. We have also presented the framework to faculty at 10 workshops. We believe that the ability to provide a visualization of how students are navigating the exercises will have a big impact on how faculty assess student learning.

Introduction:

The US faces a major shortage of cybersecurity workers to defend our information infrastructure from attack. In recognition of this need, security has been included as a core topic in the new ACM/IEEE Computer Science 2013 Curricula. Cybersecurity is also mentioned in more than half of the other knowledge areas in this report. At educational conferences such as SIGCSE and regional CCSC conferences, we are also seeing a growing interest in cybersecurity among faculty who do not have expertise in this area. We are seeing a growing concensus articulated in the CS 2013 Curricula and in the ACM report "Toward Curricular Guidelines for Cybersecurity" to integrate cybersecurity into the Computer Science curriculum at multiple levels in multiple courses. It benefits faculty who do not have time or expertise to create new exercises.

One of our main design objectives for EDURange is to create exercises that nurture analysis skills. When speaking of analysis skills, we mean the ability to reason about large, complex, and opaque data and systems. Strong analytical skills enable people to impose structure and meaning on such artifacts, reason about these relationships, and draw meaningful conclusions or inferences. These are precisely the kinds of skills that we believe are useful in many cybersecurity scenarios from security policy design to reverse engineering to vulnerability analysis. Analysis skills work in conjunction with the security mindset, which is the ability to think about how systems can fail, and be made to fail in different ways, even as one is designing a system. Questioning assumptions plays an important role in both defense and attack. In developing our exercises, we focus on the following analysis skills:

1. Verifying assumptions by checking network messages, protocols, file formats and other input data constraints to see if layers of abstraction are coherent and correct. Enumerating and checking if failure modes, exceptions, and errors are controlled, caught or anticipated.
2. Gaining understanding of program, network, or system behavior and semantics, network topology or organization, or a defense posture. Observing and enumerating how software components or network elements are actually composed.
3. Extracting Information from opaque artifacts. For example, analyzing a raw dump of network traffic or intrusion alerts or firewall logs and recognizing true anomalies.
4. Creating Emergent Resilience by understanding a system well enough to design and propose enhancements to reliability, fault tolerance, or availability.

This paper is about the design of visualization tools that allow us to track a student's progress in carrying out one of the exercises in our framework.

Related work

There is a growing pool of curriculum material, instructor/faculty training, and VM-based labs. Yet, most of the deployed exercises and hosted environments did not meet our goals[16]. For example, Towson's "Security Injections"[9] mainly focus on several important secure programming patterns, but do not emphasize analysis. The SEED[6] project presents a mature, well-documented set of exercises, which are not typically interactive or dynamic and require significant work to set up and run. Some of them were designed for the graduate level. SecKnitKit presents a number of exercises that can be integrated into upper division computer science classes such as Database

Management, Operating Systems and Networking[15]. They give good explanations of the important concepts and students get hands-on experience with important tools. However, it would not be easy for a faculty member to modify these exercises, nor do they focus on analysis skills.

Our philosphy on information security education stems from our understanding and teaching of the hacker curriculum as described by Bratus[1]. This approach is predicated on the utility of understanding failure modes. Rather than teaching students the "success" cases, we attempt to deliver a culture shock that makes them disrespect API boundaries and adopt a cross-layer view of the CS discipline as described by Bratus et al.[3]. We also routinely encourage our students to adopt a dual frame of mind (attacker and defender) when solving problems to prevent artificial abstraction layers from becoming boundaries of competence[11]. The importance of analysis skills as explained by S. Bratus et al.[2] is based on linking expected behavior to actual behavior as seen in network traces, log files, program binaries, rules/policies, system call traces, network topologies, network interactions, unknown protocols, injected backdoor code, etc. All of our exercises are based on these skills.

Our work follows the tradition of creating cybersecurity games or exercises, which are known to engage students[10,7]. This includes competitions such as CCDC, Plaid, notsosecure, iCTF[5], CSAW[7], TRACER FIRE, Packetwars, and many others. From our perspective, one problem with these competitions is that they require a significant amount of infrastructure and preparation by the organizers. For example, it took several grad students six months to create the exercises for iCTF[5]. Some competitions such as CCDC and Packetwars require the installation of physical hardware, and they require that students and their faculty travel to participate. There are also a number of non-technical games with the goal of interesting students with no technical background in cybersecurity (Control-Alt-Hack[4], [d0x3d!][8]).

A description of recon

The Recon 1 exercise was inspired by a largely similar exercise from PacketWars. We have now run the Recon 1 exercise with over 120 students in 10 different settings spanning five different institutions. It has evolved from our initial version which was a very straight forward exercise in scanning a very large IP range for hosts that respond to ICMP ping and have standard TCP ports open. Students are expected to use the tool nmap and figure out how to do this efficiently. Over time, we have created multiple levels of this exercise that challenge students to think more about the assumptions that they are making. For example, we have introduced a firewall that filters ICMP traffic and potentially other packets. Nevertheless, in the first level, we are expecting that students are not familiar with nmap and its options and how to partition networks. Thus, this is their initial focus.

As we developed the recon exercise and others, we were able to evaluate our students' assessment of the exercise, but we were faced with the challenge of assessing what they were learning. The scoring system gives students a point for every correct piece of information that they can recover about a host in the space to be scanned; however, that is very limited in terms of measuring their understanding. For example, students

could ask their neighbors for some of the answers. Actually, we want them to collaborate by dividing up the search space and each person taking a part. The scores don't indicate this. By hosting the exercise on the EC2 cloud, we are able to collect data about exactly what students did. This is a flexible framework that allows us to investigate how to assess students at a level we haven't done before[14]. It creates the possibility of observing the process of solving a complex cyber security problem. We chose to capture all of the commands that students entered and to construct a graph illustrating that data.

Discussion

It is challenging to assess student competence in cybersecurity, because we are not just interested in whether they got the answers but how they solved the problem and their ability to analyze the problem, to analyze complex systems. There has been analogous work to model students' understanding of programming[12,13]. Fact-based multiple choice tests are generally not sufficient because there is a level of skill required in using the tools well that is difficult to capture without having the student use the tool.

We chose a graphical representation because the raw data (a transcript of the commands that a student typed) is difficult to understand and distill. We believe that by extracting information and displaying it graphically we can see phases of activity in the student's work. The data are comprised of long sequences of commands that the students have typed with no comments about what they were thinking. Interestingly, some of the graphs display phases of activity. The graphical structure exposes some of the semantics of what students did and the process that they went through. It captures some of the information that could alternatively be obtained with more effort from in-depth interviews and having students thinking aloud as they solve the problem. As domain experts, we can fill in the gaps that are not recorded in the graphs. We know the skills and abilities that we want to foster, so we can apply our own filters, as well. Figure 1 shows examples of graphs that were contructed from bash histories for the first level of Recon.

Team 4 is an example of a satisfactory attempt to solve the problem. There are three distinct subgraphs which roughly represent phases of the student's activity:
1. exploration of the tools (nmap). At the top of the graph we see nmap with no arguments. The student is exploring what nmap can do before looking a solutions in more detail.
2. discovering the landscape (applying nmap to discover possible targets)
3. exploration of each target. At the top we see nmap with no arguments. The student is exploring what nmap can do before looking at solutions in more detail.

This is the process we went through in our analysis of the graph:
1 finding structure in the graph
2 look at labels, mentally collapse semantically similar labels, some of which was already done during graph generation.
3 filtering artifacts: mentally removing careless errors
4 looking for surprises – either links or labels that we didn't expect, we were looking for nmap /17, but we saw nmap /32

5 re-evaluation of initial assessment/classification into phases, what supporting evidence is there?
6 where did the student do well, what did they miss? What questions would we ask the student, what suggestions would we make?

For Team 6, the sequence of searches indicates that maybe they don't understand how to subdivide a network. They are rescanning IP ranges that they already scanned. We could use the graph as a basis for asking them. The graph for Team 6 also divides into three subgraphs.

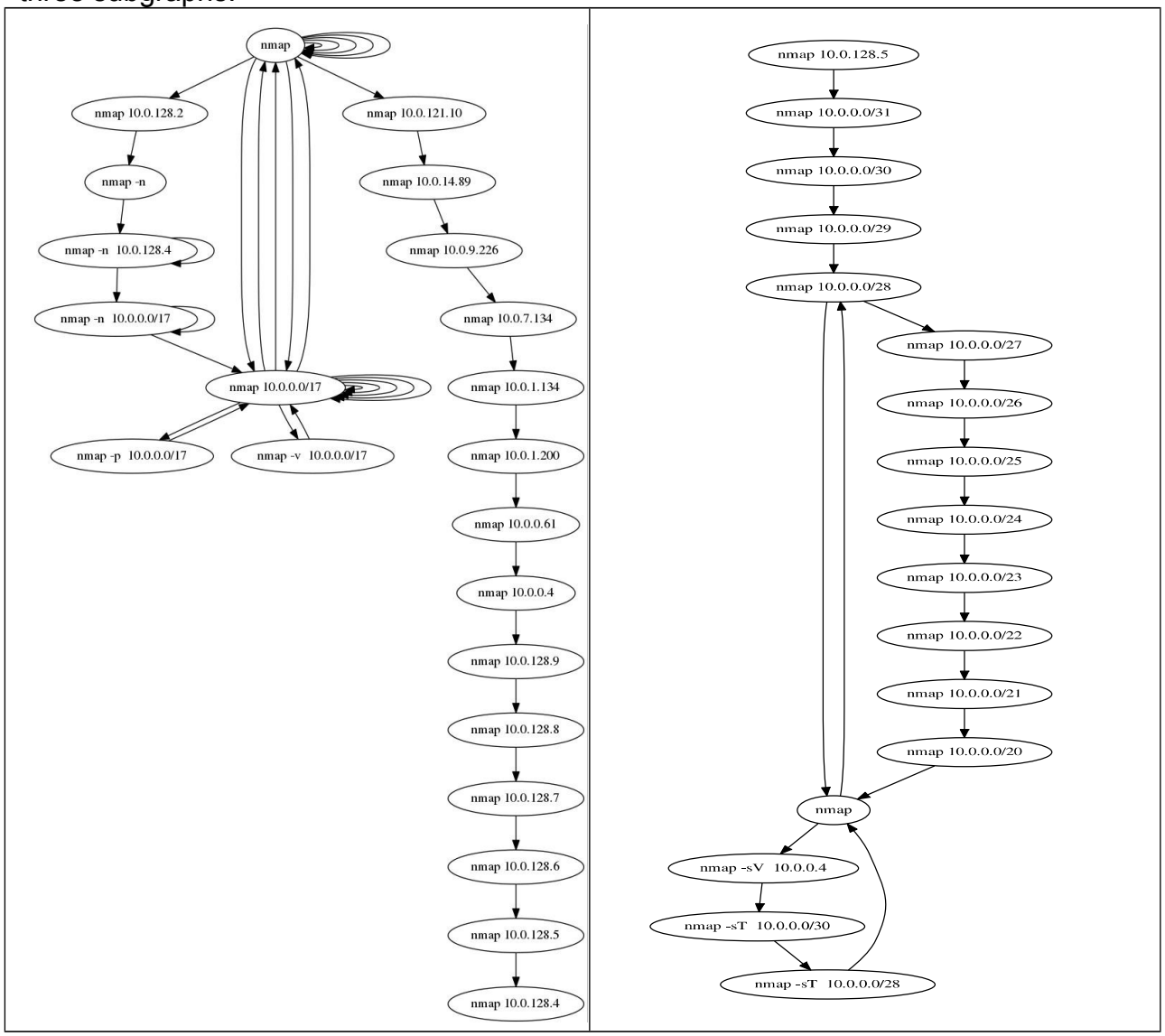

Figure1: A graphical representation of the activities for teams 4 and 6 for Recon.

Conclusion

Assessing cyber security skills is hard. We have developed an approach that provides a graphical representation of the activities of the students during an interactive exercise. This can be used by instructors to give feedback or by students to help them reflect on what the did and how they could do it better. In some cases the graphs can be easily

decomposed into pedagogical phases by the instructor, e.g. exploration of the tools, broad exploration of the target space, and focused exploration of the target space. The challenge for the future will be to automate this process so that the system can flag anomalous behavior.

References

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

[2] Bratus , S., D'Cunha, N., Sparks, E., and Smith, S. W. Toctou, traps, and trusted computing. In Trusted Computing-Challenges and Applications. Springer, 2008, pp. 14–32.

[3] Bratus , S., Shubina , A., and Locasto, M. E. Teaching the principles of the hacker curriculum to undergraduates. In Proceedings of the 41st ACM technical symposium on Computer science education (New York, NY, USA, 2010), SIGCSE '10, ACM, pp. 122–126.

[4] Denning, T., Lerner, A., Shostack, A., and Kohno, T. Control-alt-hack: the design and evaluation of a card game for computer security awareness and education. In Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security (2013), ACM, pp. 915–928.

[5] Doupe, A., Egele, M., Caillat, B., Stringhini, G., Yakin, G., Zand, A., Cavedon, L., and Vigna , G. Hit 'em where it hurts: A live security exercise on cyber situational awareness. In Proceedings of the 27th Annual Computer Security Applications Conference (New York, NY, USA, 2011), ACSAC '11, ACM, pp. 51–61.

[6] Du, W., and Wang, R. Seed: A suite of instructional laboratories for computer security education. J. Educ. Resour. Comput. 8 (March 2008), 3:1–3:24.

[7] Gavas, E., Memon, N., and Britton, D. Winning cybersecurity one challenge at a time. Security & Privacy, IEEE 10, 4 (2012), 75–79.

[8] Gondree, M., and Peterson, Z. N. Valuing security by getting [d0x3d!]: Experiences with a network security board game. Presented as part of the 6th Workshop on Cyber Security Experimentation and Test (Berkeley, CA, 2013), USENIX.

[9] Turner, C. F., Taylor, B., and Kaza, S. Security in computer literacy: a model for design, dissemination, and assessment. In Proceedings of the 42nd ACM technical symposium on Computer science education (New York, NY, USA, 2011), SIGCSE '11, ACM, pp. 15–20.

[10] Vigna, G. Teaching Network Security through Live Exercises. In Proc. 3rd Ann. World Conf. Information Security Education (WISE 03) (2003), Kluwer Academic, pp. 3–18.

[11] White, G., and Nordstrom, G. Security across the Curriculum: Using Computer Security to Teach Computer Science Principles. In Proceedings of the 19th National Information Systems Security Conference (1996), NIST, pp. 483–488.

[12] Soloway, E., and Ehrlich, K. Empirical studies of programming knowledge. Software Engineering, IEEE Transactions on SE-10, 5 (Sept 1984), 595–609.

[13] Blikstein, P., Worsley, M., Piech, C., Sahami, M., Cooper, S., and Koller, D. Programming pluralism: Using learning analytics to detect patterns in the learning of computer programming. Journal of the Learning Sciences 23, 4 (2014), 561–599.

[14] Weiss, R., Locasto, M., and Mache, J. A Reflective Approach to Assessing Student Performance in Cybersecurity Exercises. Proceedings of the 47th ACM Technical Symposium on Computer Science Education (2016), SIGCSE'16, ACM

[15] Siraj, A., Ghafoor, S., Tower, J., and Haynes, A. Empowering Faculty to Embed Security Topics into Computer Science Courses. In Proceedings of the Conference on Innovation & Technology in Computer Science Education (ITiCSE '14), (2014), ACM, pp. 99-104.

[16] Weiss, R., Mache, J., and Nilsen, E. Top 10 Hands-on Cybersecurity Exercises. Journal of Computing Sciences in Colleges 29, 1 (2013), 140–147.