A Design for a Collaborative Make-the-Flag Exercise

Matt Bishop(✉)

University of California at Davis, Davis, CA, USA
mabishop@ucdavis.edu

Abstract. Many people know how to compromise existing systems, and capture-the-flag contests are increasing this number. There is a dearth of people who know how to design and build secure systems. A collaborative contest to build secure systems to meet specific goals—a "make-the-flag" exercise—could encourage more people to participate in cybersecurity exercises, and learn how to design and build secure systems. This paper presents a generic design for such an exercise. It explores the goals, organization, constraints, and rules. It also discusses preparations and how to run the exercise and evaluate the results. Several variations are also presented.

1 Introduction

Cybersecurity has become a major concern, and its lack a serious problem in society. Exacerbating this problem is the poor quality of software and systems, enabling attackers to exploit vulnerabilities that compromise security. This is a product of many things, including the economics of the marketplace [3,4] and a lack of programmers and system developers who understand how to craft programs and systems that meet a specific set of security requirements, as well as more generic robustness requirements.

In computer security curricula and competitions, a common exercise is to have students find flaws in existing systems. In some cases, the organizers of competitions make their own systems (such as DefCon's Clemency system [1]). The goal of these exercises and competitions (called "Capture-the-Flag" or "CTF" contests) is to teach students how to find and exploit vulnerabilities, thereby teaching them what to avoid doing.

A variant of these CTF competitions is to provide the contestants with an existing system that is known to have vulnerabilities. They are given some period of time, such as a month, to harden the system so that any vulnerabilities cannot be exploited, and all attempts to do so are recorded. The systems are then attacked by other teams or a "red team" and the contestants are given points for the attacks they have blocked. These "Protect-the-Flag" ("PTF") competitions are more constructive than the CTF ones because the emphasis is on securing a system, not breaching it.

L. Drevin and M. Theocharidou (Eds.): WISE 2018, IFIP AICT 531, pp. 3–14, 2018.
https://doi.org/10.1007/978-3-319-99734-6_1

Consider the ultimate goal of security. It is to create systems that satisfy a specific set of requirements. The CTF competition focuses on showing an existing system fails to do this. A PTF competition focuses on protecting an existing but fundamentally non-secure system to prevent it from violating a set of security requirements. Neither of these do what a "secure system" is to do: demonstrate to some desired level of assurance that a system meets a set of specific requirements, including security requirements.

This suggests an alternate exercise in which the contestants design and implement a system to meet specific requirements, including security requirements. This exercise, a "Make-the-Flag" (MTF) exercise, has the teams work from the ground up to design and build a secure system, rather than work from the top down to take a system apart. Such a exercise would of necessity involve a special-purpose system because designing and implementing a general-purpose system from scratch would take too long. This shifts the focus to creating secure systems, thereby decreasing the problem of a lack of practitioners who can do that. It also forces students to pull together everything they have learned in computer science classes—software engineering, robust programming, networking, security, and so forth—to build a system that will be tested thoroughly for vulnerabilities. It will also encourage academic programs to put more emphasis on teaching this art of construction.

A second aspect of an MTF exercise is that it can be run collaboratively rather than competitively. This makes it attractive to people who either find competition distasteful or do not have confidence that they will score well on a competition. In the collaborative form, team members can support members of other teams as well as members of their own team. The teams compete against a set of requirements, and the evaluation of a team's effort results in a non-numeric report of the quality of their work. Thus, there is no high or low score. Of course, an MTF exercise can be run as a competition by providing numeric scores for the components of the evaluation; we shall return to this later.

In this paper, we explore how a collaborative MTF exercise might be organized and run.

2 Background

Traditional CTF competitions are exercises in which contestants set up systems containing a "flag," or indicator. The object of the competition is to capture as many flags from other teams' systems while preventing the capture of your flag. Scoring takes into account both the number of flags captured and the number of times the contestants' own flag has been captured.

Several versions of this basic format exist. MIT Lincoln Labs held a CTF contest for Boston-area universities with the goal of providing practical cybersecurity education [15]. They distributed the system as a virtual machine, encouraged the students to study it, and before the contest provided five lectures on various aspects of cybersecurity and vulnerabilities. The final lecture was a lab exercise in which students worked through various challenges using Google's Gruyere service [8]. The score for the exercise depended on defense, calculated as a weighted

sum of confidentiality, integrity, and availability measures, and then combined with a weighted measure of offense, or the capture of other teams' flags. The organizers released two plug-ins, one near the beginning of the contest and the other near the end, that had to be added to the systems. Failure to do so diminished the availability score.

Every DefCon has a CTF contest. In one DefCon CTF contest, the "flag" was a data file, and a "capture" was defined as corrupting that file. So the goal was to corrupt as many opponent's data files as possible without allowing yours to be corrupted. Cowan's team used this to test their Immunix server [6].

The U.S. military schools run a PTF, the Cyber Defense Exercise (CDX), annually. In some competitions, the schools could choose their own network architectures and associated security architecture [10]. In another [2], each military school was given control of enterprise systems that were poorly managed. In all exercises, the systems were on an isolated, closed network. The teams had to identify vulnerabilities and ameliorate them as well as secure the networks, and do so within a given time and budget. The U.S. National Security Agency then provided a red team to attack, and that team used only publicly available exploits. The students were scored on their ability to keep services running in the face of attacks as well as their success in detecting the attacks and defending the systems. They also had to submit reports and respond to requests.

The Collegiate Cyber Defense Competition (CCDC) is a civilian CTF run the same way as the CDX [5,12]. The students are presented with a business environment, including a web server, email, and other services. It emphasizes the operational aspects of securing the network infrastructure, as well as solving business problems. A red team acts as adversary, with limits similar to those of the CDX red team. The CCDC has grown from a small competition among Texas schools to a U.S. national competition. The traditional edition of the International CTF Competition [14] also uses this scheme.

Vigna describes three versions of these exercises [13]. The Red Team/Blue Team exercise is essentially a CTF exercise, with one set of participants playing the role of attackers (Red Team) and the other playing the role of defenders (Blue Team). The CTF exercise again split the participants into two teams, with each team attacking the other team's system and detecting (not preventing) attacks on its system by the other team. Another exercise, called the Treasure Hunt, had two teams compete to complete given tasks in a specific period of time, and the competition was to do so first within that time.

The Cyber Security Exercise Workshop [7], sponsored by the U.S. National Science Foundation, considered four types of exercises: defensive exercises; small, internal CTF competitions; national CTF competitions; and semester-long CTF competitions. It described organizational and logistical issues in establishing a CTF cybersecurity exercise.

A perceptive paper [11] discussed ways to involve members of groups that are traditionally underrepresented in the cybersecurity field, such as females. The paper presented several ways to make cybersecurity competitions more attractive, and how to support the participation of these members. Interestingly, its scope was restricted to *competitions*; it did not consider collaboration at all.

Some of the CTF exercises require that the system be protected, which typically requires the design of configurations to harden the systems; in some cases, the contestants may have to write programs. The difference is that the focus of the MTF contest is to design and implement a system from the ground up as opposed to hardening something that exists. Further, the focus of this work is on collaboration.

3 The Make-the-Flag Exercise

An MTF exercise has several steps: organizing the contest, preparing the teams, running the exercise, and then evaluating the results. Doing so involves several groups.

- The *teams* are groups that are participating in the exercise. Their goal is to build a robust, secure system.
- The *managers* of the exercise set the requirements to be met, the components of the system to be used, and any additional constraints (such as when and for how long the exercise is to run, and who may participate).
- The *testers* test the systems at the end of the competition. They do not score the results numerically; instead, they provide written reports that can be given to the teams.
- The *judges* evaluate the results of the testing and of the exercise in general. They determine the effectiveness of each system based upon the this evaluation.

The managers and judges are together called the *organizers*, and the testers and judges are together called the *evaluators*.

Teams are not ranked against one another; instead, the systems are evaluated and the evaluation serves as the results. This emphasizes that the goal of this exercise is *cooperation*, not competition.

3.1 Organizing the Contest

In this step, the organizers meet to determine the goals and rules of the contest, and to organize themselves into the managers and judges.

Goals. The generic goals of the teams in an MTF exercise are twofold. First, develop a system that meets the requirements stated by the organizers. Second, ensure the system is robust, in the sense that generic attacks such as buffer overflows do not result in the system entering a compromised state.

Each contest also has more specific goals for the teams to meet. The organizers must decide what those goals are, and to what level they are to be specified. One approach is to present an objective, leaving teams to determine how best to meet it.

Example. The objective of this MTF exercise is to develop a small computer system that will manage a set of street lights on corners. Lights opposite one another are to be paired so they are always the same color. When one set is red (stop), the other set is green (go). The computer is to be managed through an Internet connection. ■

Rules. Given this objective, the teams must develop a set of requirements, show that a system meeting the requirements will meet the objective, and then develop the system. In doing so, they will also have to develop the necessary network protocols, command interface and language, and output protocols. Further, they must document these thoroughly enough so that people who were not the developers can configure and use the system. This approach thus offers the teams the maximum degree of freedom, while teaching them to document their interfaces and other external features of their system thoroughly enough for the evaluators to be able to use and to test their system.

The disadvantage is that each system developed in the contest is likely to have completely different interfaces. This makes the utility of the system more difficult to evaluate, especially if it is to be used in a particular environment. It also increases the time needed for thorough testing. In this case, the objective should include some details of inputs and outputs:

Example. The objective of this MTF exercise is to develop a small computer system that will manage a set of street lights on corners. Lights opposite one another are to be paired so they are always the same color. When one set is red (stop), the other set is green (go). The lights will be connected using the connector described in the addendum, and controlled using the protocol described there. The computer will accept inputs as described in the addendum, both over the network and from a command-line interface. ■

The addendum specifies the interfaces with the external environment, limiting what the computer can do but providing a standard interface for all teams to implement. Thus, they need not document the protocols or the command-line interface unless they add extensions, in which case those must be documented.

An exercise to construct a simple firewall gives another example of a very detailed set of requirements,

Example. The objective of this MTF exercise is to develop a simple firewall system that will accept or reject network packets based on rulesets. The managers have devised a little language for the ruleset. The specific requirements are:

1. The system must receive packets on one network interface.
2. The system must either forward the packets over another interface, or discard them, as dictated by the ruleset.
3. The system must accept rulesets written in the RULESET language; see the addendum.
4. The system must provide a command-line interface; see the addendum.

5. Once started, the system will run until a SHUTDOWN command is entered at the user interface or until powered down. ■

In addition, the managers provide the addendum describing the RULESET language and the command-line interface.

In addition to meeting the requirements and objective, the teams must develop systems that are robust in the sense that they will handle error conditions in a reasonable manner—providing informative error messages, rejecting the bad input, and taking other actions as appropriate. For example, if the traffic light system receives inputs telling it to turn all lights green, the system should reject that input; if something fails, then the system should have the lights fail safe, that is, all either turn red or enter some other specified state. If the firewall system has too many rules, it should inform the user of the overflow, and reject the excess rules; it should not simply ignore all rules.[1]

Constraints. The organizers must also decide on other aspects of the exercise. The first is the time for the contest: when it starts and how long the teams have. The second is what equipment, and other financial limits, are necessary for each team, and how it will be procured.

Time is a complex constraint. Some team members will be students and their schedules will require attention to schoolwork, especially when examinations are being given. Similarly, non-students will have job-related constraints. So the organizers should aim for a time that minimizes the disruption of the schedules of the expected team members. It will not be possible to accommodate everyone's schedule, but the organizers should position the contest so the team members can devote maximum effort to the contest.

The financial constraints are also critical, and simulate real-world constraints. The simplest method for handling them is for the organizers to procure a set of hardware and software, and loan each team what is needed. If the objective allows the use of commodity hardware (such as PCs), then the organizers can expect the teams to have their own available, although the organizers should have some financial aid available for teams who have neither the equipment nor the support of their institution to purchase the equipment. The organizers should make any additional constraints, financial and otherwise, explicit *before* the exercise starts.

Organization. At this point, the organizers need to assemble the teams. This is a recruiting and marketing issue, and techniques similar to those used in CTF contests should work. Word-of-mouth, reaching out to faculty in cybersecurity programs, to cybersecurity clubs, and other groups will be helpful, as will the organizers determine an enticing set of prizes. Also, the organizers should seek industry and government sponsorship and support, because those groups are attempting to improve the state of commercial and non-commercial software and

[1] This is from an incident where the author and his students were testing a firewall. The bug was quickly fixed.

systems, and so should be happy to support such a contest. Their support will encourage teams to form, because the contest will be a showcase for their talents. Indeed, as with many CTF exercises, once the MTF exercise runs successfully a few times, little recruiting will be necessary.

3.2 Preparing the Teams

Once the teams are organized, they must learn the rules and objectives of the MTF contest. The organizers must be explicit is what is, and is not, allowed. For example, must the teams use the equipment that the organizers supply, or may the teams use their own equipment. If the latter, the organizers need to specify any constraints—for example, that a USB-3 port will be required to connect some specialized peripherals, or that the system is to use particular drivers that the organizers will supply. Given that many MTF contests will require the teams to develop special-purpose systems, whether the teams must develop their own system supervisor or whether they can use a commodity system like picoLinux or Windows CE must also be specified.

An interesting question is whether cross-team collaboration should be encouraged. As this is not a competition, cross-team collaboration may provide the teams with fresh ideas. It also improves the assurance aspect of the systems, as teams can co-operate to test each others' systems and point out problems that would otherwise not be discovered until the testing phase of the exercise. The organizers should make any limits clear to all.

Another question is whether there are constraints on software development. The organizers can specify that a particular language, set of libraries, or development environment is to be used, as well as a particular software development methodology. If they do this the organizers must also determine how the teams are to demonstrate that they have used the specified methodology and environment. In some sense, such a detailed specification would violate the purpose of the contest. If the goal is to encourage teams to meet the requirements, then *how* they meet those requirements should be left up to the team. As the cliché says, "You can tell me what to do or how to do it. If you want to tell me both, *you* do it!"

Once the organizers have presented the objectives, requirements, and rules, undoubtedly teams will have questions. The more clarity the organizers can provide at the beginning of the contest, the better prepared the teams will be to meet the objectives. The organizers should emphasize the non-competitive nature of the exercise by making the questions and answers available to all teams.

It will be critical to emphasize the collaborative nature of the exercise, and ensure the teams realize the only competition is against themselves—they build the requisite system, provide evidence it does what it is supposed to, and have the system do so in the face of both regular and malicious testing. Every team that does this wins. Conceivable, *all* teams could win; similarly, all teams can lose. But team A winning does not interfere in any way with team B winning.

3.3 Running the Exercise

This part is divided into two phases. The first is the development phase; the second, the testing phase.

During the development phase, the teams design and implement their systems in accordance with the rules. Their ultimate goal is to convince the evaluators that their system meets the stated requirements and objectives. Accumulating assurance evidence—evidence that the system meets the requirements provided by the application of specific techniques such as requirements tracing—is part of the way to show this.

The organizers may define both the form and the content of the documentation of the development process. The manner in which that content is gathered is up to the teams. What does matter is how they write the documentation. Documentation that organizes the methods used to gather assurance evidence, that describes how those methods were applied and the results of their application, will provide information that customers—here, the evaluators – can use to determine how well the system meets its requirements. The documentation also provides information that the evaluators can use to assess the interfaces through which people communicate with the system.

Part of gathering assurance evidence is testing. The documentation needs to describe the testing in enough detail so others can reproduce the tests and verify the results. In some cases, repeating a test may produce different results; for example, a race condition may occur infrequently, so one test will trigger it but others will not. These test results should be identified as occurring intermittently, so the evaluators understand that the results of repeated tests will vary. The developers should explain why this occurs.

Undoubtedly, in some cases the developers will not be able to conduct all the tests they think of. The tests that are not run should still be described, so the evaluators know what the developers would do with more time.

The description of each test should follow the concepts used in the flaw hypothesis methodology's flaw hypothesis generation step [9]. Each test attempts to validate a hypothesis or claim. That claim is to be stated, and the testing methodology shown to demonstrate or refute the claim. The developers then state the results, noting anything unexpected, and state whether the results support the claim.

When the time period for the exercise ends, the developers submit their system and documentation to the evaluators.

3.4 Evaluating the Results

The job of the evaluators is to determine whether the system meets the requirements and objectives of the exercise, as well as other factors such as usability and robustness.

The evaluator can use techniques such as source code analysis to look at the quality of the system. In some cases, the exercise will specify use procedures; in others, the developers will need to design these procedures to enable the system

to meet its requirements. In either case, as security is a product of the system and how and under what conditions it is to be used, the evaluators need to consider these procedures as part of the security of the system.

The teams submit documentation that assembles assurance evidence, and the methods used to collect it. The reason is to provide a road map for the evaluators. The evaluators look at what was tested, and possibly repeat the validation steps in that document. The evaluators also look at what was *not* tested to find areas that the teams either did not think of or did not test.

The evaluation itself is qualitative, not quantitative. Knowing one system scored 5 out of a possible 10 points, and another 7 out of 10, says nothing about the significance of the difference. The causes of the ratings may speak to the same aspect of the system, in which case the difference in scores may be significant, or they may speak to different aspects of the system, in which case the difference in scores is not significant. Hence a qualitative description will provide information that the team members can learn from.

4 Variations on This Theme

Variations of the exercise proposed here provide other benefits. The exercise can be framed as a competition, in which case a quantitative evaluation is necessary. The teams can also act as testers to provide preliminary feedback before the evaluation phase of the exercise.

Cooperative Competition. Many institutions and groups desire a ranking of results, as well as (or rather than) a detailed evaluation. This exercise can be changed into a competition by the judges assigning team rankings based upon their evaluation. If the organizers wish to base the rankings on specific numbers of points, they should assign points to each requirement, and a general point count to "robustness." The testers would evaluate each aspect of the requirements and robustness to which points were assigned, and the judges would determine the number of points to be assigned.

The critical aspect here is that the teams receive more than just a numeric ranking or point score. They need to learn what problems the testers found, because problems teach more than success.

Teams as Testers. To give experience in testing systems, the teams themselves can act as testers.[2]

One way to do this is to give the teams access to all systems during the testing step. Then each team would test the systems, and provide reports to both the team that developed the system and the judges. The test reports would describe the tests conducted in sufficient detail that others can reproduce them. The judges would then evaluate the systems based on the test reports. If the teams test the system, the judges should also evaluate the test reports themselves.

[2] Thanks to Dan Ragsdale for this suggestion.

A second way eliminates the students as evaluators. The MTF exercise is structured into three rounds, in the following order:

1. *Development round.* In this round, the teams build their systems. This round proceeds as the MTF exercise described above.
2. *Test round.* In this round, the teams are given access to one another's systems. They then carry out the testing. As in the other version, the test reports are made available to the other teams and the evaluators.
3. *Repair round.* The teams fix the problems found in the previous round.

After the third round, the testers, and the judges then evaluate both the results of the testing and the reports from the test round. The results of the exercise depend on both the system's quality and the quality of each team's testing.

5 Conclusion

This paper explored an alternative to a traditional CTF contest. This alternative is constructive, in that the goal is to create a secure system rather than find holes in an existing system. While the latter is instructive and necessary to teaching students how to "think like an attacker," the former gives the students experience in creating hardened systems. It allows them to practice the principles of secure development, implementation, and evaluation.

The other aspect is an emphasis on collaboration rather than competition. CTF exercises can be intimidating, particularly to those who have never participated in one, or who feel themselves overmatched by more experienced players. The Make-the-Flag exercise encourages teamwork that helps team members learn from one another and from a qualitative evaluation of their work. In collaborative exercises, there is no penalty for aiding another team, so the learning can cross team boundaries. This type of exercise will be appealing to those who are not by nature competitive.

One could also picture a CTF contest run collaboratively. Such an exercise would follow the pattern of the CCDC or CDX, except that the teams would work together to help each other secure their systems. They would compete as one group against the red team testers, and their goal would be to minimize the success of that team. The scoring or evaluation would require the red team to keep careful track of what they tried, and what worked and what did not, so the other teams could receive a detailed evaluation of each of their systems. Such a framing would allow teams to try different approaches and see which ones worked best, without the fear of "losing" or "winning" due to these different implementations.

Competitive exercises, and exercises emphasizing attacks, have their place. But many potential cybersecurity students are dissuaded from following that field because of this emphasis. Emphasizing collaboration and construction may draw them in, to the benefit of the field, the profession, and the community. It is an idea worth trying.

Acknowledgements. Thanks to Dan Ragsdale of Texas A&M University and Kara Nance of the Virginia Polytechnic Institute and State University for helpful discussions. The author gratefully acknowledges support of the National Science Foundation under Grant Numbers DGE-1303211 and OAC-1739025, and a gift from Intel Corporation. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation, Intel Corporation or the University of California at Davis.

References

1. The cLEMCy architecture, July 2017. https://blog.legitbs.net/2017/07/the-clemency-architecture.html
2. Adams, W.J., Gavas, E., Lacey, T., Leblanc, S.: Collective views of the NSA/CSS cyber defense exercise on curricula and learning objectives. In: Proceedings of the Second Workshop on Cyber Security Experimentation and Test. USENIX Association, Berkeley, August 2009. https://www.usenix.org/legacy/event/cset09/tech/full_papers/adams.pdf
3. Anderson, R.: Why information security is hard–an economic perspective. In: Proceedings of the 17th Annual Computer Security Applications Conference. IEEE Computer Society, Los Alamitos, December 2001. https://doi.org/10.1109/ACSAC.2001.991552
4. Anderson, R., Moore, T.: Information security economics – and beyond. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 68–91. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74143-5_5
5. Conklin, A.: The use of a collegiate cyber defense competition in information security education. In: Proceedings of the Second Annual Conference on Information Security Curriculum Development, pp. 16–18. ACM, New York, September 2005. https://doi.org/10.1145/1107622.1107627
6. Cowan, C., Arnold, S., Beattie, S., Wright, C., Viega, J.: DefCon capture the flag: defending vulnerable code from intense attack. In: Proceedings of the 2003 DARPA Information Survivability Conference and Exposition. IEEE Computer Society, Los Alamitos, April 2003. https://doi.org/10.1109/DISCEX.2003.1194878
7. Hoffman, L.J., Rosenberg, T., Dodge, R., Ragsdale, D.: Exploring a national cybersecurity exercise for universities. IEEE Secur. Priv. **3**(5), 27–33 (2005). https://doi.org/10.1109/MSP.2005.120
8. Leban, B., Bendre, M., Tabriz, P.: Web application exploits and defenses (2017). https://google-gruyere.appspot.com/
9. Linde, R.R.: Operating system penetration. In: Proceedings of the AFIPS 1975 National Computer Conference, pp. 361–268. ACM, New York, May 1975. https://doi.org/10.1145/1499949.1500018
10. Mullins, B.E., Lacey, T.H., Mills, R.F., Trechter, J.M., Bass, S.D.: How the cyber defense exercise shaped an information-assurance curriculum. IEEE Secur. Priv. **5**(5), 40–49 (2007). https://doi.org/10.1109/MSP.2007.111
11. Pusey, P., Gondree, M., Peterson, Z.: The outcomes of cybersecurity competitions and implications for underrepresented populations. IEEE Secur. Priv. **14**(6), 90–95 (2016). https://doi.org/10.1109/MSP.2016.119

12. Pusey, P., OBrien, C.W., Lightner, L.: Preparing for the collegiate cyber defense competition (CCDC): a guide for new teams and recommendations for experienced players. National Cyberwatch Center, Largo, January 2015. https://www.nationalcyberwatch.org/resource/resource-guide-preparing-for-the-collegiate-cyber-defense-competition-ccdc-a-guide-for-new-teams-and-recommendations-for-experienced-players-2/
13. Vigna, G.: Teaching network security through live exercises. In: Irvine, C., Armstrong, H. (eds.) Security Education and Critical Infrastructures. IFIPAICT, vol. 125, pp. 3–18. Springer, Boston (2003). https://doi.org/10.1007/978-0-387-35694-5_2
14. Vigna, G., Borgolte, K., Corbetta, J., Doupe, A., Fratantonio, Y., Invernizzi, L., Kirat, D., Shoshitaishvili, Y.: Ten years of iCTF: the good, the bad, and the ugly. In: Proceedings of the 2014 USENIX Summit on Gaming, Games, and Gamification in Security Education. USENIX Association, Berkeley, August 2014. https://www.usenix.org/conference/3gse14/summit-program/presentation/vigna
15. Werther, J., Zhivich, M., Leek, T., Zeldovich, N.: Experiences in cyber security education: the MIT Lincoln laboratory capture-the-flag exercise. In: Proceedings of the Fourth Workshop on Cyber Security Experimentation and Test. USENIX Association, Berkeley, August 2011. http://static.usenix.org/legacy/events/cset11/tech/final_files/Werther.pdf