Experiences with Honey-Patching in Active Cyber Security Education

Frederico Araujo Mohammad Shapouri Sonakshi Pandey Kevin W. Hamlen
The University of Texas at Dallas
{*frederico.araujo, mxs139130, sbp140130, hamlen*}*@utdallas.edu*

Abstract

Modern cyber security educational programs that emphasize technical skills often omit or struggle to effectively teach the increasingly important science of cyber deception*. A strategy for effectively communicating deceptive technical skills by leveraging the new paradigm of* honey-patching *is discussed and evaluated. Honey-patches mislead attackers into believing that failed attacks against software systems were successful. This facilitates a new form of penetration testing and capture-the-flag style exercise in which students must uncover and outwit the deception in order to successfully bypass the defense. Experiences creating and running the first educational lab to employ this new technique are discussed, and educational outcomes are examined.*

1 Introduction

Industrial and governmental demand for employees with superlative, comprehensive cyber security expertise has risen meteorically over the past several years. Cyber security job postings have increased 74% from 2007 to 2013—over double the rate of increase of other IT jobs—and take 24% longer to fill on average than other IT job postings [8]. One reason demand for high expertise is eclipsing supply is the increasing sophistication of threats faced by cyber professionals. In 2014, cyber attacks against large companies rose 40%, yet malicious campaigns are becoming smaller and more efficient, using 14% less email to successfully infiltrate victim networks [15].

This underscores a need for effective yet broadly deployable educational strategies for all aspects of cyber security training. Yet some cyber skills are exceptionally difficult to convey effectively in a classroom setting. A prime example is *cyber deception*, which is becoming an increasingly central ingredient of many offensive and defensive scenarios (cf. [1, 16, 21]). Deceptive social engineering attacks in which attackers impersonate government officials account for over $23,200 in losses *per day* in 2014, according to the FBI Internet Crime Complaint Center [13]. Advanced malware attacks often undertake elaborate user deceptions, such as Stuxnet's replaying of pre-recorded, normal equipment readings to operators at the Natanz nuclear facility during its attack [14]. In light of such practices, the U.S. Air Force has announced cyber deception as a specific focus area for 2015–2016 [22].

To raise defender vigilance against deceptive threats, a different way of thinking is required—one that adopts the thinking process of the adversary [17, 19, 23]. Modern defenders must understand the psychology of attackers, and be aware of their strategies and techniques in order to anticipate their actions. In active defense contexts, they require skills for both creating and mitigating deceptive software. Awareness of such issues facilitates development of safer programs, and limits the attack surface exposed to cyber criminals.

However, effectively teaching such awareness in a traditional classroom setting can be challenging. Typically, the scholastic experience is contrived, with lectures and assignments following a structured sequence of topics through which students expect to be guided by instructors, and where reading materials provide the theoretical backbone of a rehearsed, time-honored mode of thinking. From a security standpoint, it can be seen as antithetical to most real-world cyber security threat encounters involving advanced adversaries: Modern, targeted cyber threats are often surreptitious, diverse, and unpredictable. Advanced threat-actors are aware of standard educational practices, and therefore adopt strategies that run counter to them. To defend against such threats, future cyber security professionals must be empowered with techniques that can delay reconnaissance efforts, degrade exploitation methods, and confound attackers into moving and acting in a more observable manner.

Among the most promising approaches towards alleviating this problem are Capture the Flag exercises (CTFs), which are commonly organized as competitions where teams score points by exploiting opponents and defending from attacks in real time. However, although such

exercises are of great educational value in that they offer lessons not easily taught in a classroom and provide a realistic, safe environment for practicing offensive techniques, they often lack emphasis on *active* cyber defense topics (cf. [12]). We believe that ways must be sought to ethically teach students deception and anti-deception techniques in order to make networks more resilient against the emerging wave of advanced threats.

Toward this end, we have been examining *honey-patching* [2,3] as a new tool for effectively teaching active defense and attacker-deception to students in the Computer Science Department at The University of Texas at Dallas (UTD). Honey-patching is a recent technique developed to deceive, misdirect, and disinform attackers by deceptively portraying the outcome of blocked attacks to attackers. Specifically, honey-patches are software security patches that fix newly discovered software vulnerabilities in such a way that future attempted exploits of the patched vulnerabilities appear successful to attackers. This masks patching lapses, impeding attackers from easily discerning which systems are genuinely vulnerable and which are actually patched systems masquerading as unpatched systems. It also affords defenders tremendous opportunities to gather information about the threat (e.g., collecting and analyzing previously unseen malware, for possible attack attribution), feed disinformation to the attacker in the form of falsified honey-data (cf., [6,20,24]), and even deploy counterattack measures to strike back at attackers.

In April of 2015 we organized a small-scale computer lab at UTD to raise awareness about this technique and the broader concepts surrounding cyber deception and anti-deception. The lab was organized with the help of UTD's Computer Security Group (CSG) student association, constituting the first effort to teach honey-patching techniques and strategies outside a research setting. Although small in its size (seven students completed the lab session), the response we obtained from the participants of this early test were extremely positive. Our goal is to relate our experiences to other educators, and to recommend methods and software tools that we have found pedagogically effective for teaching students these important skills. We also plan to leverage this initial experience to contribute larger-scale CTF exercises to major competitions in the future, such as TexSAW (Texas Security Awareness Week), which is held annually at UTD every October.

The research reported herein is covered by UTD IRB approval MR15-185. The educational lab and subsequent data analysis were conducted by personnel who are NIH-certified in protection of human research subjects. The lab was organized and overseen by student officers trained in risk management, including ethical and nondiscriminatory treatment of individuals.

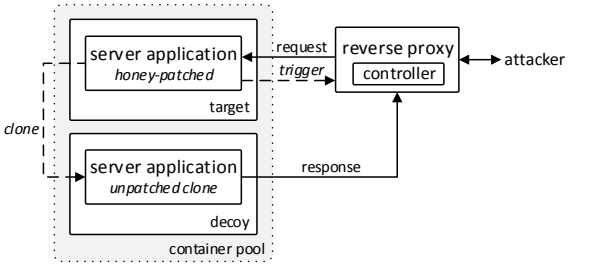


Figure 1: Architectural overview of honey-patching.

2 Honey-Patching

Patching continues to be the most ubiquitous and widely accepted means for addressing newly discovered security vulnerabilities in commodity software products. In 2014 alone, major software vendors reported over 7000 patched (or soon-to-be-patched) separate security vulnerabilities to the National Vulnerability Database, almost 25% of which were ranked highest severity [9]. However, despite the increasingly prompt availability of security patches, a majority of attacks in the wild continue to exploit vulnerabilities that are known and for which a patch exists [4,5,11]. This is in part because patch adoption is not immediate, and may be slowed by various considerations, such as patch compatibility testing, in some sectors.

As a result, determined, resourceful attackers often probe and exploit unpatched, patchable vulnerabilities in their victims. For example, a 2013 security audit of the U.S. Department of Energy revealed that 60% of DoE desktops lacked critical patch updates, leading to a compromise and exfiltration of private information on over 100,000 individuals [10]. The prevalence of unpatched systems has driven the proliferation of tools and technologies via which attackers quickly derive unique, previously unseen exploits from patches [7], allowing them to infiltrate vulnerable systems.

Attackers are too often successful at finding and exploiting patching lapses because conventional software security patches advertise rather than conceal such lapses. For example, a request that yields garbage output from an unpatched server, but that yields an error message from a patched server, readily divulges whether each server is vulnerable. Cyber criminals therefore quickly and efficiently probe large networks for vulnerable software, focusing their attacks on susceptible targets.

To counter this weakness, honey-patching, depicted in Figure 1, has been recently proposed as an effective means of adding deceptiveness to software patches. In response to malicious inputs, honey-patched applications clone the attacker session onto a confined, ephemeral, decoy environment, which behaves henceforth as an unpatched, vulnerable version of the software. This potentially augments the server with an embedded honeypot

Listing 1: Abbreviate patch for CVE-2014-6271

```
1 + if ((flags & SEVAL_FUNCDEF) && command->type != cm_function_def)
2 + {
3 +     internal_warning ("%s: ignoring function definition attempt", ...);
4 +     should_jump_to_top_level = 0;
5 +     last_result = last_command_exit_value = EX_BADUSAGE;
6 +     break;
7 + }
```

Listing 2: Honey-patch for CVE-2014-6271

```
1   if ((flags & SEVAL_FUNCDEF) && command->type != cm_function_def)
2   {
3 +   hp_fork();
4 +   hp_skip(
5       internal_warning ("%s: ignoring function definition attempt", ...);
6       should_jump_to_top_level = 0;
7       last_result = last_command_exit_value = EX_BADUSAGE;
8       break;
9 +   );
10  }
```

that waylays, monitors, and disinforms criminals. Deceptive honey-patching capabilities thereby constitute an advanced, active defense technique that can impede, confound, and misdirect such attacks, and significantly raise attacker risk and uncertainty.

Honey-patching Shellshock. In September 2014 we honey-patched the Shellshock GNU Bash remote command execution vulnerability (CVE-2014-6271) [18] within hours of its public disclosure as part of our AFOSR/NSF active defense and attack-attribution research program. Shellshock was one of the most severe vulnerabilities in recent history, affecting millions of then-deployed web servers and other Internet-connected devices. This high impact combined with its ease of exploitation makes it a prime candidate for penetration testing exercises.

Listing 1 shows an abbreviated, vendor-released patch in diff style for Shellshock. The patch introduces a conditional that validates environment variables passed to Bash, declining function definition attempts. Prior to this patch, attackers could take advantage of HTTP headers as well as other mechanisms to enable unauthorized access to the underlying system shell of remote targets. This patch exemplifies a common vulnerability mitigation: dangerous inputs or program states are detected via a boolean test, with positive detection eliciting a corrective action. The corrective action is typically readily distinguishable by attackers—in this case, a warning message is generated and the function definition is ignored.

Listing 2 presents an alternative, honey-patched implementation of the same patch. In response to a malicious input, the honey-patched application forks itself onto a confined, ephemeral, decoy environment, and behaves henceforth as an unpatched, vulnerable version of the software. Specifically, line 3 forks the user session to a decoy container, and macro `hp_skip` in line 4 elides the rejection in the decoy container so that the attack appears to have succeeded. Meanwhile, the attacker session in the original container is safely terminated (having been forked to the decoy), and legitimate, concurrent connections continue unaffected.

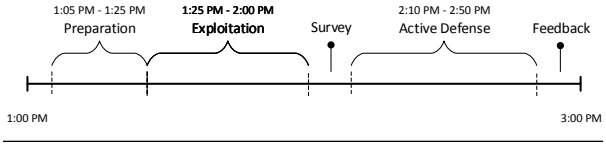


Figure 2: Lab timeline and overview.

As a result, adversaries attempting to exploit Shellshock in a victim server that has been honey-patched receive server responses that seem to indicate that the exploit has succeeded. However, the shell commands they inject are actually executing in a decoy environment stocked with disinformation for attackers to explore. This provides an ideal environment for students to penetrate as part of exercises focused on cyber deception. Some of their attacks may genuinely hijack the victim server (e.g., those that exploit unpatched vulnerabilities), others observably fail (e.g., those that exploit patched vulnerabilities), while yet others only appear to succeed (e.g., those that exploit honey-patched vulnerabilities). The challenge is to discover that a deceptive outcome exists and counter it.

3 Lab Overview

We organized an open lab with the main goal of educating students on offensive security and active cyber defense concepts using honey-patching as the underlying framework. To boost students expectation and interest, we selected the Shellshock [18] vulnerability as our unit of study. This choice was motivated by the scale and impact of Shellshock (severity 10 out of 10), and the low access complexity of the attack, suitable for the two hours allotted for the lab. Figure 2 shows the lab timeline with the approximate durations of each of its three parts.

Preparation. The first part provided an overview of the lab session, followed by a brief introduction to Shellshock. As part of this description, we covered the historical background and relevancy of Bash, and detailed the various attack vectors that can be used to exploit server applications running the vulnerable shell. In addition, we introduced basic background on our particular target server deployment (e.g., Apache, CGI). At the end of this exposition, we ran an interactive demonstration of Shellshock to test students' understanding of the vulnerability and ensure that they were familiarized with the lab environment.

Exploitation challenge. The hands-on part of the lab consisted of a challenge. Students were asked to attack our server and attempt to escalate their privileges after gaining access to the server. To complete this exercise,

students needed to build their own exploits and apply the knowledge acquired in the preparation session of the lab. At the end of this exercise, we asked students to fill out an online survey. In order to obtain an unbiased feedback, students were unaware that they were attacking a honey-patched system. We only revealed this information in the third and last part of the lab.

Deception-based active defense. In the last part of the lab, we first provided a brief overview of deception-based techniques for active defense and offensive countermeasure concepts (e.g., honeypots, decoys, beacons). Then we introduced students to honey-patching and disclosed the fact that our target server was honey-patched, explaining its underlying mechanisms, including misdirection and monitoring capabilities. We also demonstrated the process involved in honey-patching Shellshock. To conclude the exercise, we gave students the opportunity to attack the system once again, for another 30 minutes, and then presented and discussed the monitoring logs generated by the honey-patched system. Before we adjourned, we asked students to fill out a second survey providing feedback about their learning experience.

4 Lab Design

From the provisioning of the required physical resources and setup of the lab environment to the preparation of tutorials and challenges, there is a considerable amount of effort involved in organizing a hands-on cyber security lab. Even though the number of students was small, we designed this exercise to scale to a much larger number of participants. In what follows, we highlight some of the preparation steps for this lab.

4.1 Infrastructure & Preparation

Figure 3a illustrates the infrastructure created for the lab. We built this infrastructure atop VMWare's ESXi, allowing us to quickly and efficiently deploy many linked VMs as needed to create individual guest environments for each participant. The target server and attacker VMs were deployed within the same subnet, and access control rules isolated the lab from the rest of the university network. This created a safe environment in which exploits could be attempted without risk to the surrounding network.

Target server. The target server was honey-patched against Shellshock and hosted a CGI shell script deployed atop Apache for processing user authentication in a web application specifically created for this lab. To entice students to further exploit the system, decoys were generated with fake user accounts and honey-files containing "interesting" information, such as fake credentials and weakly encrypted user account passwords. To gain escalated access to the decoy, students could discover vulnerable paths

Listing 3: Decoy's file-system monitoring

```
1 25/04/2015−13:24:25 /usr/local/apache/cgi−bin/ I_Shocked_You CREATE
2 25/04/2015−13:24:25 /usr/local/apache/cgi−bin/ I_Shocked_You OPEN
3 25/04/2015−13:24:25 /usr/local/apache/cgi−bin/ I_Shocked_You ATTRIB
4 25/04/2015−13:24:25 /usr/local/apache/cgi−bin/ I_Shocked_You CLOSE...
```

Listing 4: Decoy's deep inspection of network packets

```
1 0x0020: 8018 00e5 1aed 0000 0101 080a 0032 9a09  .............2..
2 0x0030: 0032 9a09 3261 0d0a 495f 5368 6f63 6b65  .2..2a..I_Shocke
3 0x0040: 645f 596f 750a 6c6f 6769 6e2e 6367 690a  d_You.login.cgi.
4 0x0050: 6d69 6e65 0a6e 6f5f 796f 755f 6469 646e  mine.no_you_didn
5 0x0060: 740a 0d0a                                t...
```

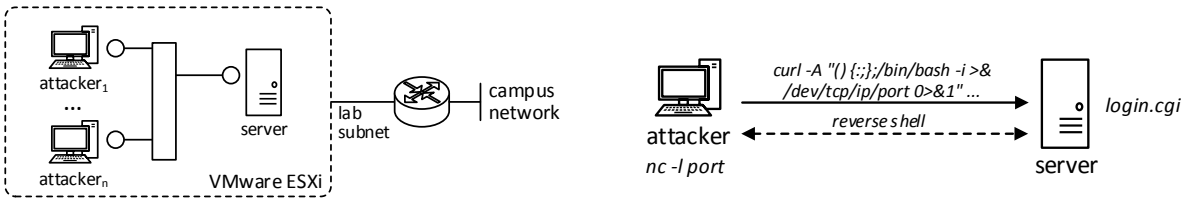
concealed within the system. For example, participants might transfer the encrypted password file to their own machines, and crack it with a password cracker (e.g., using a dictionary attack).

Monitoring. Decoys also hosted software monitors that collected fine-grained attack information. To minimize the performance impact on decoys, we used two powerful and highly efficient tools: *inotifywait* (to track modifications made to the file system), and *tcpdump* (to monitor ingress and egress of network packets). To avoid possible tampering with the collected data, all logs were stored outside the decoy environments. In addition, we tuned both monitoring tools to avoid generating spurious outputs (e.g., by excluding certain directories and limiting the monitored network traffic). Listings 3 and 4 show sample monitoring logs produced after an attack executed by a student. The logged file system events reveal that the student created a file named `I_Shocked_You` in the server's (actually, a decoy's) CGI directory and changed the created file's permissions. In the network logs, we see the response payload returned to the student for an attack that ran the `ls` command on the server.

Attacker environment. Each student was assigned a guest VM prepared specifically for the hands-on exercises. Each VM ran Ubuntu 14.04, and came with the minimal tools required to complete the demonstration session (i.e., `curl`, `nc`). Student accounts were configured with administrative privileges, and internet access was not prohibited, allowing easy installation of additional tools as needed. For example, several students downloaded and installed password crackers to use during the hands-on session.

4.2 Interactive Demonstration

The demonstration delivered at the end of the preparation session consisted of a *no-one-left-behind* exercise, in which the instructor explains each step of the demo and waits until all students have successfully completed it. This strategy worked well given our small group, but would probably need to be adjusted for a larger number of students (e.g., by having more tutors walking around and assisting whoever gets stuck). We used this demo to



(a) Lab subnet and virtualization infrastructure

(b) Obtaining a reverse shell with Shellshock

Figure 3: Lab preparation illustrating (a) lab subnet and virtualization infrastructure and (b) attack demonstration leveraging Shellshock to obtain a reverse shell.

further clarify concepts introduced in the initial lab presentation and to ensure that all students started the free hands-on session with a basic working knowledge of the techniques used to exploit Shellshock. For instance, Figure 3b illustrates one of the attacks we demonstrated, in which the attacker leverages Shellshock to obtain a reverse shell on the vulnerable server.

4.3 Participants

The lab was open to any student willing to participate, and we did not impose any restrictions on required background. To reach interested students, we announced the lab through the homepages and mailing lists of the security and computer student organizations at UTD. To catch students' attention, we promoted the lab as a hands-on challenge on Shellshock exploitation and defense.

The participants were all CS majors, with limited experience in cyber security (ranging from none to some), with a few who had performed penetration tests before. The lab was staffed by one PhD student and two Masters students who acted as tutors for the lab and offered participants individual assistance as needed. This organization dynamics worked well to solve issues quickly and facilitate the fluidity of the lab.

5 Survey Results

To informally assess the effectiveness of the lab, we asked students to complete two online surveys (see Appendix A) made available through Google Forms. These surveys were anonymous. To minimize the influence of the survey questions on student behavior during the hands-on sessions, the survey questions were not disclosed until after the students completed each portion of the exercise that was surveyed—prior knowledge of the questions would reveal too much about the exercise. Survey questions were phrased as boolean inquiries (1=yes, 0=no) followed by open-ended clarifying questions in which students were given the opportunity to comment.

Deceptiveness of Honey-patching. The first survey examined the deceptiveness of honey-patching by asking students whether they had realized that they were interacting with a decoy. All students answered *"no"* to this question. From their responses, it is clear that the honey-patched server successfully deceived students for the entire duration of the first hands-on session, which lasted about 30 minutes.

In the last part of the lab, after revealing the deception to students, we asked, "If you were given enough time, what would you attempt to do?" Responses included, *"[I would] look at the services that are running (in the decoy) and try to exploit the honey-patched system."* Another student said, *"[I would] note files of interest and various properties of them (who created them, permissions),"* and another mentioned, *"I would try to find red flags that could be used to probe a honey-patched system."* A particularly noteworthy response was that of a student who said s/he would attempt to relay back to the honey-patch components, in particular the front-end proxy, in order to look for security flaws and exploit them.

These results are a preliminary indication of the efficacy of honey-patching for raising student awareness of cyber deception and counterintelligence gathering, and its educational value for encouraging students to seek deception-exposing strategies and examine exploit outcomes critically rather than accepting them at face value.

Learning Experience. Students also answered general questions about their educational experience. For example, in response to the question, "Did you find this exercise useful for expanding your cyber security education?" students unanimously answered *"yes."* In the open-ended comments, students also said that it was exciting to see how the exploit worked first-hand. Indeed, learning the concepts involved in attacking and defending computer systems in a safe and coherent context seems to entice students' curiosity and develop their interest in applied cyber security.

We also received copious constructive feedback from students on possible ways in which we could improve the lab. These include proposals for new challenges, different methods of attack, and alternative ways to defend against them. Overall, this was a very successful learning experience with a very positive response from students. When

asked, "Did this exercise increase your interest in the research side of cyber security?" one student commented, *"I also enjoyed seeing the research being done to take advantage of these kinds of exploits in terms of defense."*

6 Discussion & Lessons Learned

Lab Organization. We organized this lab combining short, alternating structured (lecturing, demo) and unstructured (free hands-on) sessions. This choice was made to keep students focused and motivated, while giving them freedom to experiment on their own. We believe that this approach helped us to strike a good balance between guided and exploratory learning.

In addition, we believe that concealing the honey-patching deception from students during the first hands-on session raised their interest relative to disclosing it immediately, and was well received by students. While we were initially concerned that students might feel betrayed by instructors once the deception was revealed, our experiences indicate that allowing students to experience a real (but benign and educational) deception during the exercise evokes an element of surprise that students find intriguing and memorable. In particular, we observed a notable increase in interest after we introduced the research on honey-patching and revealed that they had been interacting with decoys since the beginning of the lab. This was evidenced by a surge in questions and discussions.

Second, the delayed reveal opened the way for students to imagine new application scenarios that we did not even cover in our short presentation. For example, a very interesting suggestion was to use honey-patching as a strategy to enhance incidence response and help defenders gather additional attack evidence shortly after a target is compromised.

Research & knowledge transfer. Transferring research findings and abstract knowledge into practical use is critical for improving the security posture of cyber space. This includes creating a body of security guidelines, information materials, and more comprehensive education programs focusing on fostering such transition. In our opinion, information assurance and security programs should complement the traditional classroom experience with hands-on exercises in which students are invited to try new research and become armed with state-of-the-art tools and techniques to protect our privacy and the world we live in from emerging cyber threats.

Cyber deception CTF. To cultivate additional student involvement, we also intend to develop a CTF competition at TexSAW with a focus on cyber deception and honey-patching. This will be an offense-defense team challenge, in which participants will learn and practice a variety of skills spanning deception and anti-deception techniques. We envision at least two different ways in which we can organize this competition.

In the first mode, all participants will be taught about honey-patching to use it to misdirect and deceive attacks. In this style of competition, each team will not only try to capture the flag, but also avoid submitting captured decoy flags impersonating genuine flags. To make it more challenging, flag validation and score computation will only occur at the end of each predetermined phase.

A second approach is to enter teams trained in cyber-deceptive active defense techniques into pre-existing CTF competitions, concealing that intended strategy from rival teams. If successful, this could provide empirical evidence of the efficacy of honey-patching and other deceptive defenses for waging cyber warfare. A challenge for such evaluation is finding competitions with rules sufficiently open-ended that they admit these deceptive techniques. Many CTFs are structured such that flag validation is immediate and automatic, making deception less valuable in that context than it is in practice.

7 Conclusion

Cyber deception is an increasingly important component of effective, real-world cyber defenses. It can be leveraged to level a battlefield that otherwise inherently favors attackers, who succeed if they find just one vulnerability, over defenders, who only succeed if they close all vulnerabilities. By concealing which attacks succeed and which fail, honey-patches give defenders valuable advance intelligence about attacker gambits, and offer opportunities to misdirect attackers away from critical targets toward non-critical targets.

However, like many cyber security paradigms, deception is an arms race. Effective deception depends upon effective skills imparted by effective educational methods. Our initial experiences creating and running active cyber defense lab exercises for computer science students have indicated that honey-patching can be deployed in an educational setting to teach cyber deception in ways that overcome the otherwise predictable (and therefore non-deceptive) classroom environment. We therefore advocate incorporating such exercises into future CTF competitions and into cyber educational curricula to bring these skills to a broader array of upcoming cyber security professionals.

Acknowledgments

The research reported herein was supported in part by AFOSR award FA9550-14-1-0173, NSF CAREER award #1054629, NSF Scholarship For Service (SFS) award #1027520, and ONR award N00014-14-1-0030. Any opinions, conclusions, or recommendations expressed are those of the authors and not necessarily of the AFOSR, NSF, or ONR.

A Survey Questions

A.1 First Survey

Q1. Did you succeed in attacking the server? (yes/no) If yes, what actions did you take after you were able to exploit the vulnerability?
Yes: 7/7, No: 0/7

Q2. Did the vulnerable server raise any red flags? (yes/no)
Yes: 0/7, No: 7/7

Q3. If Yes to Q2: Did you think you were interacting with a real server (i.e., not a trap)? (yes/no) If not, please explain.

Q4. If Yes to Q2: Did you observe anything anomalous in any of the following: file-system, server responses? (yes/no) If yes, how long until you observed them?

A.2 Second Survey

Q1. After your were told that the system was honey-patched, what actions did you take? Did you try to hack the system? (yes/no)
Yes: 1/7, No: 6/7

Q2. If you were given enough time, what would you attempt to do?

Q3. Did you find this exercise useful for expanding your cyber security education? (yes/no)
Yes: 7/7, No: 0/7

Q4. Were the tutorial instructions clear? (yes/no) If not, please suggest improvements.
Yes: 7/7, No: 0/7

Q5. Were the student instructors helpful and responsive? (yes/no)
Yes: 7/7, No: 0/7

Q6. Did this exercise increase your interest in the research side of cyber security? (yes/no) Please elaborate.
Yes: 7/7, No: 0/7

References

[1] ALMESHEKAH, M. H., AND SPAFFORD, E. H. Planning and integrating deception into computer security defenses. In *Proc. New Security Paradigms Workshop (NSPW)* (2014), pp. 127–138.

[2] ARAUJO, F., AND HAMLEN, K. W. Compiler-instrumented, dynamic secret-redaction of legacy processes for attacker deception. In *Proc. 24th USENIX Security Sym.* (2015). forthcoming.

[3] ARAUJO, F., HAMLEN, K. W., BIEDERMANN, S., AND KATZENBEISSER, S. From patches to honey-patches: Lightweight attacker misdirection, deception, and disinformation. In *Proc. 21st ACM Conf. Computer and Communications Security (CCS)* (2014), pp. 942–953.

[4] ARBAUGH, W. A., FITHEN, W. L., AND MCHUGH, J. Windows of vulnerability: A case study analysis. *IEEE Computer 33*, 12 (2000).

[5] BILGE, L., AND DUMITRAS, T. Before we knew it: An empirical study of zero-day attacks in the real world. In *Proc. 19th ACM Conf. Computer and Communications Security (CCS)* (2012), pp. 833–844.

[6] BOWEN, B. M., HERSHKOP, S., KEROMYTIS, A. D., AND STOLFO, S. J. Baiting inside attackers using decoy documents. In *Proc. 5th Int. ICST Conf. Security and Privacy in Communication Networks (SecureComm)* (2009), pp. 51–70.

[7] BRUMLEY, D., POOSANKAM, P., SONG, D., AND ZHENG, J. Automatic patch-based exploit generation is possible: Techniques and implications. In *Proc. 29th IEEE Sym. Security & Privacy (S&P)* (2008), pp. 143–157.

[8] BURNING GLASS TECHNOLOGIES. Job market intelligence: Report on the growth of cybersecurity jobs, March 2014.

[9] FLORIAN, C. Most vulnerable operating systems and applications in 2014. GFI Software, February 2015.

[10] FRIEDMAN, G. H. Evaluation report: The Department of Energy's unclassified cyber security program. Tech. Rep. DOE/IG-0897, U.S. Dept. of Energy, Oct. 2013.

[11] FRITZ, J., LEITA, C., AND POLYCHRONAKIS, M. Server-side code injection attacks: A historical perspective. In *Proc. 16th Int. Sym. Research in Attacks, Intrusions and Defenses (RAID)* (2013), pp. 41–61.

[12] HECKMAN, K. E., WALSH, M. J., STECH, F. J., O'BOYLE, T. A., DICATO, S. R., AND HERBER, A. F. Active cyber defense with denial and deception: A cyber-wargame experiment. *Computers & Security 37* (2013), 72–77.

[13] INTERNET CRIME COMPLAINT CENTER (IC3). 2014 internet crime report. Federal Bureau of Investigation, May 2015.

[14] LANGNER, R. Stuxnet: Dissecting a cyberwarfare weapon. *IEEE Security & Privacy 9*, 3 (2011), 49–51.

[15] LINGENHELD, M. The unfortunate growth sector: Cybersecurity. *Forbes* (April 2015).

[16] LUO, X., BRODY, R., SEAZZU, A., AND BURD, S. Social engineering: The neglected human factor for information security management. *Information Resources Management J. (IRMJ) 24*, 3 (2011), 1–8.

[17] MINK, M., AND FREILING, F. C. Is attack better than defense? teaching information security the right way. In *Proc. 3rd Annual Conf. Information Security Curriculum Development (InfoSecCD)* (2006), pp. 44–48.

[18] NIST. The Shellshock Bash Vulnerability. https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2014-6271, Sep. 2014.

[19] PATRICIU, V.-V., AND FURTUNA, A. C. Guide for designing cyber security exercises. In *Proc. 8th WSEAS Int. Conf. Recent Advances in E-Activities, Information Security and Privacy* (2009), pp. 172–177.

[20] SALEM, M. B., AND STOLFO, S. J. Decoy document deployment for effective masquerade attack detection. In *Proc. 8th Int. Conf. Detection of Intrusions and Malware, and Vulnerability Assessment* (2011), pp. 35–54.

[21] TWITCHELL, D. P. Social engineering in information assurance curricula. In *Proc. 3rd Annual Conf. Information Security Curriculum Development (InfoSecCD)* (2006), pp. 191–193.

[22] U.S. AIR FORCE MATERIEL COMMAND. Capabilities for cyber resiliency. Broad Agency Announcement, Solicitation BAA-RIK-14-07, August 2014.

[23] VIGNA, G. Teaching network security through live exercises. In *Security Education and Critical Infrastructures*, C. Irvine and H. Armstrong, Eds. Kluwer Academic Publishers, 2003, pp. 3–18.

[24] YUILL, J., DENNING, D., AND FEER, F. Using deception to hide things from hackers: Processes, principles, and techniques. *J. Information Warfare 5*, 3 (2006), 26–40.