DOI: https://doi.org/10.1145/3386363

Project-Based Learning Continues to Inspire Cybersecurity Students: The 2018–2019 SFS Research Studies at UMBC

Enis Golaszewski, Alan T. Sherman, Linda Oliva, Peter A. H. Peterson, Michael R. Bailey, Scott Bohon, Cyrus Bonyadi, Casey Borror, Ryan Coleman, Johannah Flenner, Elias Enamorado, Maksim E. Eren, Mohammad Khan, Emmanuel Larbi, Kyle Marshall, William Morgan, Lauren Mundy, Gabriel Onana, Selma Gomez Orr, Lauren Parker, Caleb Pinkney, Mykah Rather, Jimmy Rodriguez, Bryan Solis, Wubnyonga Tete, Tsigereda B. Tsega, Edwin Valdez, Charles K. Varga, Brian Weber, Ryan Wnuk-Fink, Armand Yonkeu, Lindsay Zetlmeisl, Damian Doyle, Casey O'Brien, Joseph Roundy, Jack Suess

Introduction

The world needs cybersecurity professionals who can solve new complex problems arising from an evolving landscape of networked information systems. One way to develop these workers is to give them challenging real-world tasks in a safe mentored environment, where they can apply their knowledge to gain experience while making a positive impact. To that end, for the past three years, we have organized project-based learning (PBL) workshops to engage cybersecurity students at the University of Maryland, Baltimore County (UMBC) to analyze the security of targeted portions of their campus network. Students worked collaboratively to identify potential vulnerabilities, devise proof-of-concept attacks, and recommend mitigations.

We report on our continuing experiences with these studies, highlighting the new problems addressed in 2018 and 2019, changes we made, and lessons we learned conducting these real-world hands-on educational research activities, in the hope that others may benefit from our experiences.

In 2017, students analyzed the NetAdmin custom scripts that enable faculty and staff to open the UMBC firewall to permit external access to machines they control on the research subnet [5,6]. In 2018, students analyzed the WebAdmin custom software that UMBC students, faculty, and staff use to manage credentials and accounts. In 2019, students analyzed the Virthost system UMBC uses to host student webpages. Each study enjoyed the strong support of the UMBC Department of Information Technology (DoIT), which administers the UMBC computer systems. DoIT provided all needed sourcecode and a functional virtual copy of the relevant portions of the campus network to enable participants to work safely on the virtual copy. Despite the increasing difficulty of the challenge each year, students quickly engaged and produced insightful solutions.

Although the ideas underlying the attacks found are not new, the analyses of UMBC's WebAdmin and Virthost are. Notable aspects of the studies include the following. In 2018, students found a beautifully instructive example of a "confused-deputy attack," made possible by failure to sanitize security questions and answers: An IT staff member——through carrying out their proper procedures for resetting a user password——unwittingly executes malware on their own machine by viewing the answers to security questions. In a new twist to help students learn about the importance of the insider threat, as participants were surprised to learn at the final briefing, one student in the 2019 study secretly worked throughout the week as a "mole," passively collecting passwords of the other participants.

After providing some background on our research study, we discuss the technical details of the 2018 and 2019 studies. We then discuss our experiences, evaluate each study's educational effectiveness, identify lessons learned, and present takeaways for students, educators, and system administrators.

SFS Research Studies

Inspiration for our research studies came from two sources. First, we sought a way to initiate new cybersecurity students into our existing cohort. Second, as strong believers in the value of PBL [3,4], we planned for students to learn by engaging in small groups to solve authentic tasks. Our study of student misconceptions in cybersecurity [7] revealed that many students fail to appreciate the subtlety and complexity of real-world cybersecurity. Analyzing the security of portions of the UMBC network sustains student interest and helps students appreciate such subtlety and complexity.

As a National Center of Excellence in Cybersecurity Education and Research (CAE, CAE-R), UMBC offers several cybersecurity scholarships, including the DoD Cybersecurity Scholarship Program (CySP) and NSF Cybercorps Scholarship for Service (SFS). UMBC is one of a handful of schools that extends SFS scholarships to nearby community colleges. Our research studies provide a way to introduce the community college scholars into the UMBC cybersecurity environment. Although we started our research studies for SFS scholars, the concept would work for any group of cybersecurity students.

Each day, 15–20 SFS scholars met in a spacious conference room from approximately 9am to 5pm on the UMBC campus. Students brought laptops and arranged themselves into informal groups based on interests, and worked on focused tasks that the groups self-selected. Each participant signed a non-disclosure agreement. Throughout each day, 1-3 mentors (including UMBC professors and NSA experts) visited to support the students and answer any technical questions. Particularly interesting were the daily afternoon briefings, during which the programmer from DoIT responsible for the custom software being analyzed also visited to discuss student findings. Students who could not attend in person joined a student-led evening chat session on Google Hangouts or Slack. Some of the more experienced students emerged as leaders. Following each workshop, volunteer participants finished writing a report.

Analyzing custom software offers many advantages. In contrast with a large commercial system, custom software ensures that we can easily gain access to the sourcecode and that it has a manageable size. Furthermore, because the sourcecode under consideration had never undergone a careful security review, it seemed likely that there would be vulnerabilities.

Discussion

We now discuss five selected aspects of our studies: our confused deputy attack from 2018, our experience teaching about the insider threat in 2019, evaluation of the studies, lessons learned, and how others might benefit from our experiences.

Confused-Deputy Attack

The most interesting attack we found during the 2018 study is an ironically beautiful and instructive example of a "confused-deputy attack" [1], wherein a DoIT staff member—through carrying out their proper procedures for resetting a user password—unwittingly executes malware on their own machine. First, with their own or acquired credentials, the adversary inserts a malicious script into the security question answers for a user's account (possibly their own). Second, acting or masquerading as the user, the adversary telephones DoIT requesting a password reset. Third, following procedure, the staff member views the user's security questions and answers, thereby executing the malware. Properly sanitizing user inputs would thwart this attack.

Insider Threat

During the 2019 study, to help students understand and appreciate the dangers of insider threats, study organizer Alan Sherman secretly recruited a student "mole" to passively gather passwords of the other study participants throughout the week. (Because this study was purely for educational purposes, the UMBC Institutional Review Board ruled that no special review was required.) The mole succeeded in collecting several passwords to UMBC systems, smartphones, and gmail accounts. During the briefing at the conclusion of the study, Sherman announced that there had been a security breach and wrote the first three characters of several passwords on the board. None of the students had any idea of the nature of the breach. When Sherman informed the group that there had been a mole, none of the students could identify the culprit.

This event was a powerful learning moment. Our hope is that, in the future, our students will be more mindful about the dangers of "shoulder surfing," sharing passwords, malicious insiders, and leaving unattended machines unlocked. Whether participants will actually change their future behaviors remains to be seen. Due to the success of this surprise, we plan to create similar learning moments in future studies through unannounced attacks.

Evaluation

Throughout the study activities, facilitators noticed and reflected on challenges and indicators of success. At the end of each study, participants completed a survey. Students overwhelmingly expressed the value of teamwork experienced. The majority of respondents stated they had the opportunity to try something they had not previously done before. There was consensus that access to DoIT staff and technical experts contributed to successful outcomes. 82% of the participants agreed with the statement: "The study improved my cybersecurity knowledge and skills." (53% strongly agreed and 29% agreed). 80% of the participants agreed with the statement: "I would recommend this study to other cybersecurity students." (61% strongly agreed and 19% agreed). There were no significant differences in the responses for the 2018 and 2019 studies.

Lessons Learned

As documented by student responses to feedback forms, each of our studies was highly successful in inspiring students and helping them learn through authentic challenges. The strong support of UMBC's DoIT was a crucial component. We hope that the following reflections are useful to anyone who wishes to conduct a similar study.

While no time is convenient for everyone, scheduling the event in January (during our open winter period between semesters) worked much better than did late May. Because of the intensive nature of the study, we scheduled the event while students were not taking any classes, which run September–December and February–mid May. Many of our SFS scholars started summer internships immediately after final exams, conflicting with the May study.

Offering an evening chat session helped make the study accessible to those with conflicts during one or more days. Text-based chat (versus an audio conference) was helpful because it created a transcript of the session and made it easier for students to join the session late. In future studies, for maximum flexibility, we plan to try a continuously ongoing session throughout the week.

Encouraging the students to organize into specialized groups worked well, enabling students to pursue their interests and exercise selected skills. Smaller groups also facilitated greater participation and interaction by all. Students formed and used such groups more formally and thoroughly in 2018 and 2019 than they did in 2017. For example, in 2018, students organized themselves into groups focusing on reconnaissance and identifying network topology, white-box testing (with complete knowledge of the software), static analysis of software, analyzing network traffic, and exploring known vulnerabilities of the operating system and component software in use.

Throughout each study, it was essential for students to be able to write and share notes, observations, ideas, and findings. Each year we became more organized in how to record and share such information. We found it very convenient to use Google Docs, including as a place to ask questions to DoIT and the programmer and to receive answers asynchronously.

Each year we struggled with the challenge of how and when to write a report documenting the study, a challenge we addressed better each year but have not completely resolved. It takes significant time and effort to write a report, and after the study ends many students are busy with courses and other activities. Although participants were supposed to leave important notes on the shared file system, some of them left important information only on their personal laptops. Our recommendation is to build time for documentation into the schedule and, within reason, to write as the study progresses. Everyone should be encouraged to record all important information on the shared file system. Still, it will be necessary for some team members to spend a significant amount of time after the study completing the report.

In 2017, with all students bringing laptops, we initially did not have enough electrical receptacles. Learning from this experience, in each subsequent year we came prepared with a sufficient number of power strips.

Some students felt uncomfortable with the lack of more instructor-driven guidance, but we feel that much of the learning experience came from students having to work independently, devising their own approaches and plans.

For some students, it would have been helpful to prepare them for the study by providing instruction on selected tools in advance (e.g., static code analyzer, network packet analyzer).

How Others Might Benefit from Our Experiences

While UMBC performs this activity with SFS scholars, an enthusiastic IT department, and highly skilled government security experts, a similar activity could be carried out at virtually any institution of higher learning. SFS scholars are not required for the activity, only students with sufficient security skills, interest, and integrity. While DoIT has been extremely accommodating, students could investigate any software with security ramifications, preferably where the sourcecode is available and stakeholders can provide guidance and feedback. Such software could be provided by any local business or government agency with appropriately challenging and accessible problems. Although UMBC is close to and has relationships with NSA security experts, students would benefit from any observers who have security skills and the ability to mentor students and help them reason about open-ended problems. Instead of conducting the activity as a non-credit one-week study, one could offer a more substantial variation of the activity as a credit course.

Conclusion

Our studies engaged and motivated students. Partnering qualified students with IT Departments can reap benefits for everyone: students gain exciting, concrete, hands-on collaborative experiences; educators are given rich and realistic case studies supporting Project-Based Learning (PBL); and IT Departments receive free cybersecurity consultations. Our use of PBL for all studies enhanced the productivity and value of the experiences. Students worked effectively in teams on authentic problems and developed valuable skills through sustained and focused efforts. There is strong evidence that PBL is one of the most valuable learning methodologies in a variety of fields, particularly in the field of science and engineering [3,8]. That we have been able to carry out a similar study for three years successfully demonstrates that our methods are useful and repeatable. We look forward to continuing similar studies each year and hope that other schools can benefit from similar activities.

Acknowledgments

We thank Jack Suess and Damian Doyle (UMBC Division of Information Technology) for their enthusiastic cooperation. Thanks to Richard Baldwin and Travis Scheponik for helpful comments. This project was supported in part by the National Science Foundation under SFS grants 1241576 and 1753681. Sherman was also supported by the National Science Foundation under SFS capacity grant 1819521 and by the U.S. Department of Defense under CAE-R grant H98230-17-1-0349 and CySP grants H98230-17-1-0387 and H98230-18-0321.

References

1. Hardy, N. The confused deputy: (or why capabilities might have been invented). ACM SIGOPS Operating System Review, 22 (1988), 36–38.

2. NVD CVE-2017-3167;https://nvd.nist.gov/vuln/detail/CVE-2017-3167. Accessed 2019 October 8.
3. Pucher, R. and Lehner, M. Project Based Learning in Computer Science: A review of more than 500 projects. Procedia-Social and Behavioral Sciences, 29 (2011), 1561–1566.
4. Alan T. Sherman, M. Dark, A. Chan, T. Morris, L. Oliva, J. Springer, B. Thuraisingham, C. Vatcher, R. Verma, and S. Wetzel. The INSuRE Project: CAE-Rs collaborate to engage students in cybersecurity research. IEEE Security & Privacy, 15(4):72–78, August 2017.
5. Alan T. Sherman, Enis Golaszewski, Edward LaFemina, Ethan Goldschen, Mohammed Khan, Lauren Mundy, Mykah Rather, Bryan Solis, Wubnyonga Tete, Edwin Valdez, Brian Weber, Damian Doyle, Casey O'Brien, Linda Oliva, Joseph Roundy, and Jack Suess. The SFS summer research study at UMBC: Project-Based learning inspires cybersecurity students. Cryptologia, 32(4):293–312, March 2019.
6. Alan T. Sherman, Peter A. H. Peterson, Enis Golaszewski, Edward LaFemina, Ethan Goldschen, Mohammed Khan, Lauren Mundy, Mykah Rather, Bryan Solis, Wubnyonga Tete, Edwin Valdez, Brian Weber, Damian Doyle, Casey O'Brien, Linda Oliva, Joseph Roundy, and Jack Suess. Project-Based Learning Inspires Cybersecurity Students: A Scholarship-for-Service Research Study. IEEE Security & Privacy, 17(3):82–88, May-June 2017.
7. J. Thompson, G.L. Herman, T. Scheponik, E. Golaszewski, A.T. Sherman, D. Delatte, D. Phatak, K. Patsourakos, and L. Oliva. Student misconceptions about cybersecurity concepts: Analysis of student think-aloud interviews. Journal of Cybersecurity Education, Research & Practice, 1(5), 2018.
8. Tseng, K.-H., Chang, C., Lou S., and Chen, W. Attitudes towards science, technology, engineering and mathematics (STEM) in a project-based learning (PjBL) environment. International Journal of Technology and Design Education, 23 (2013), 87–102.

The appendices present the technical cybersecurity details of the 2018 and 2019 studies, including for each the problem, adversarial model, and potential vulnerabilities, attacks, and recommendations. These details offer many authentic instructive examples, including the importance of sanitizing inputs and properly configuring systems.

2018 SFS Research Study

Following the success of the 2017 SFS research study, we scheduled a second one for January 2018. Here, we discuss its problem, background, scope, and adversarial model. We also analyze the system, identify vulnerabilities and attacks, and present recommendations.

Problem

UMBC provides authenticated web-based services to over ten thousand students, faculty, and staff per semester, requiring a method for administrators and users to manage credentials, profiles, and settings. WebAdmin is a web application developed by DoIT for this purpose. DoIT administrators use WebAdmin to reset passwords, lock accounts, and control access for user accounts, while users can access other credential, email, and directory services. WebAdmin data contains potentially sensitive information including home addresses, personal telephone numbers, email addresses, birthdates, campus affiliations, pictures, and identification numbers. Compromise of WebAdmin could result in a critical information disclosure and subject UMBC community members to malicious behavior. The sheer volume of private data and the global availability of WebAdmin make it a critical and attractive target. With DoIT's

cooperation, the 2018 UMBC SFS cohort analyzed the security of WebAdmin to determine potential vulnerabilities, risks, and attacks resulting from WebAdmin's design and implementation.

System Background

As shown in Figure 1, WebAdmin consists of an Apache web server running a Perl web application, and UMBC's Lightweight Directory Access Protocol (LDAP) service. Users of WebAdmin are in numerous campus groups (e.g., staff, faculty, student), some of which provide special administrative privileges. The web application is Internet-accessible and requires only a UMBC account, but the LDAP server resides on DoIT's private internal network.

WebAdmin relies heavily on HTML web forms, which users fill out. Submitted forms trigger Perl functions that execute queries reading and writing fields on the LDAP server. WebAdmin ensures that users are affiliated with groups appropriate for their requests, preventing low-privilege accounts (e.g., students) from submitting administrative requests. Communication between users and WebAdmin uses end-to-end encryption (HTTPS), preventing casual eavesdropping attacks.

Scope

Our analysis of WebAdmin searched for ways to escalate privileges, compromise privileged accounts, and exfiltrate sensitive personal data by exploiting flaws in WebAdmin code, website, and DoIT policies. As in 2017, we did not consider social engineering attacks on DoIT staff, nor did we consider attacks on UMBC's network or physical attacks on machines, students, faculty, or staff. The security of Apache sourcecode and the third-party LDAP implementation were also out of scope.

Adversarial Model

We assume a moderately skilled adversary with stolen student credentials. With these credentials, the adversary can access WebAdmin from anywhere in the world and view, modify, or delete the victim's personal information. The adversary also has access to numerous web forms and can target them with injection attacks and other malicious behaviors. Our adversary's goal is to gain administrator privileges and access private information of other users. We assume our adversary cannot break cryptography and does not have zero-day exploits against other components (e.g., Perl or browsers). The attacker can call DoIT and masquerade as a legitimate student, does not wish to be discovered, and does not want to crash WebAdmin.

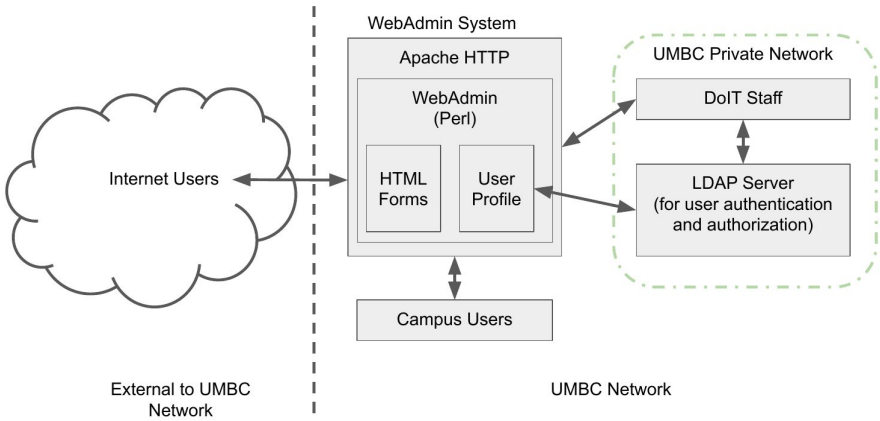

Figure 1: Architectural diagram of UMBC's WebAdmin web-based account administration application, which consists of an Apache HTTP web server serving an application scripted in Perl. User profiles, authenticated through LDAP, include HTML forms for modifying home address, telephone numbers, security settings, and other personal information. Users, whether connecting from the campus network or from the Internet, cannot access the UMBC private network. Arrows depict communication paths.

How We Analyzed the System

On the first day, DoIT briefed the students and gave them a virtualized copy of WebAdmin and the LDAP server, and a pair of "dummy" credentials. Virtual test environments enabled students to investigate aggressively, developing and testing exploits without harming production systems. Some students spent the first morning exploring WebAdmin's capabilities and its numerous HTTP forms, quickly identifying potentially vulnerable fields. Other students began testing WebAdmin with a battery of reconnaissance tools, verifying some of the first group's findings and identifying several new possibilities. Although the students previously knew little about WebAdmin's internals, by the end of the first day, the group discovered how it worked and how it might be broken.

For the rest of the week, students worked in interest-based groups producing ideas, developing exploits, and recommending policy. Groups used a variety of tools, including Kali Linux, web-vulnerability, and static code analyzers. Students examined WebAdmin sourcecode line-by-line. Each afternoon, students enthusiastically presented their findings to WebAdmin maintainers. These presentations usually resulted in a patch that evening or the very next morning. By week's end, having exhausted simpler vulnerabilities, teams analyzed DoIT policies, resulting in numerous potential attacks leveraging WebAdmin's administrative procedures.

Potential Vulnerabilities, Attacks, and Recommendations

1. Many form fields insufficiently sanitize and validate inputs. For example, answer fields for security questions accept any input, and while email fields require variations of "string@string",

this restriction is easy to circumvent. Adversaries can add malicious JavaScript and HTML into these fields to perform cross-site scripting, trick administrators by modifying the page, or to submit forms, all run under the privilege of the page viewer. As shown in Figure 2, an adversary can trick an administrator into viewing these fields by initiating a phone-based password reset. DoIT policy requires administrators changing passwords to authenticate the user by manually verifying personal information including security question answers. To mitigate these threats, WebAdmin must properly sanitize all user input so that adversaries cannot submit malicious scripts as inputs, preferably using community-maintained libraries, such as Sanitize.

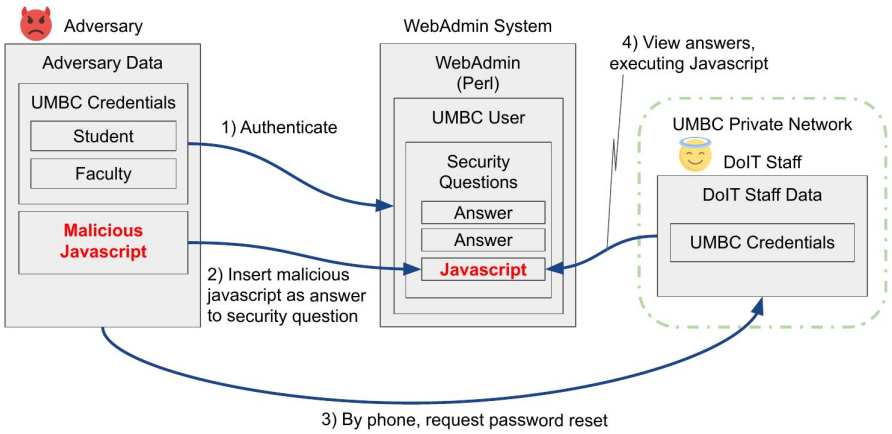

Figure 2: A confused-deputy attack that tricks a DoIT system administrator into executing malicious JavaScript placed in the user's answers to security questions. Exploiting DoIT's policies for phone-based account password recovery, an adversary with compromised UMBC credentials requests a password reset, prompting a DoIT staff member to view the security questions and answers, thereby executing the malware.

2. WebAdmin sourcecode includes valid, hard-coded administrator-level credentials for the LDAP server with privilege to read and modify sensitive data for virtually every UMBC account. Anyone with the sourcecode and network access to the LDAP database could exfiltrate or modify LDAP records. WebAdmin sourcecode should not include hard-coded credentials, but should store them separately and read them when launching.

3. Security question answers and two-factor authentication settings are modifiable without re-authentication, allowing anyone with brief access (e.g., borrowing an unlocked, logged-in device) to change them. By changing security answers to known values and disabling two-factor authentication, an adversary can reset the password and take over the account. Instead, WebAdmin should require users to re-authenticate when making any security changes to their WebAdmin accounts or viewing security question answers.

4. Apache is nine years out of date. The current version is 2.4.41, but WebAdmin version 2.2.15 from 2010 has a vulnerability [2] potentially allowing a compromised Apache module to bypass normal authentication mechanisms. Administrators must aggressively patch public-facing services including WebAdmin.
5. WebAdmin exposes details of administrator functionality to all users regardless of privilege. While non-privileged users cannot use these functions, they can learn what is available to administrators. Do not expose administrator functionality to non-administrative users, even if non-functional. While we do not advocate "security by obscurity," providing free reconnaissance to adversaries can reduce the effort required and increase the effectiveness of cyber and social engineering attacks.

2019 SFS Research Study

Our partnership with DoIT continued with the January 2019 study, where students analyzed DoIT's Virthost project. As before, we discuss the security problem, system background, scope, adversarial model, analysis process, and the vulnerabilities and potential attacks we found and our recommendations for DoIT.

Problem

UMBC has long provided web hosting for students, faculty, and staff. Unfortunately, over time, these websites often linger in unmaintained and vulnerable states. To mitigate the risk of vulnerable websites, DoIT hosts sites inside isolated virtual areas called "webspaces," served by a system called Virthost. Each webspace serves only the domain name assigned to it and isolates itself from other webspaces, limiting damage caused by compromise. Compromised webspaces pose a serious risk to Virthost and to the campus distributed file system, but the system had never undergone a security evaluation.

The 2019 SFS cohort analyzed the Virthost system to answer three primary questions:

- Can an adversary with a compromised user webspace break the containment to attack another webspace?
- Can an adversary with a compromised user webspace break the containment to attack the Virthost system?
- How secure is mod_waklog, an Apache module critical to Virthost?

System Background

To implement a university-wide networked file system, UMBC uses the Andrew File System (AFS). AFS is a well-known, distributed file system that uses secret-key cryptography for authentication. UMBC stores files on an AFS instance shared with many other universities around the globe, relying on the access control mechanisms and cryptographic properties of AFS for security.

As shown in Figure 3, Virthost comprises four load-balanced servers running CentOS 7 and several Apache web server instances, which serve content from webspace directories residing on AFS. To create a webspace, DoIT creates a new directory in a designated area of AFS and assigns read-only and read-modify privileges to Virthost and the webspace owners. Webspaces can contain web files and scripts,

use content managers (e.g., Wordpress), and make database connections, providing many potential vulnerabilities. There are more than 100 such webspaces, owned and maintained by various entities. An adversary with a compromised webspace gains read-modify access to its AFS directory.

An adversary able to escalate the privileges of a webspace could potentially attack other webspaces or unrelated areas of AFS, potentially belonging to other universities. To enforce separation between webspaces, Virthost uses mod_waklog, an Apache module last maintained in 2015, which interfaces with AFS and manages the cryptographic keys that authorize read, write, and modify operations.

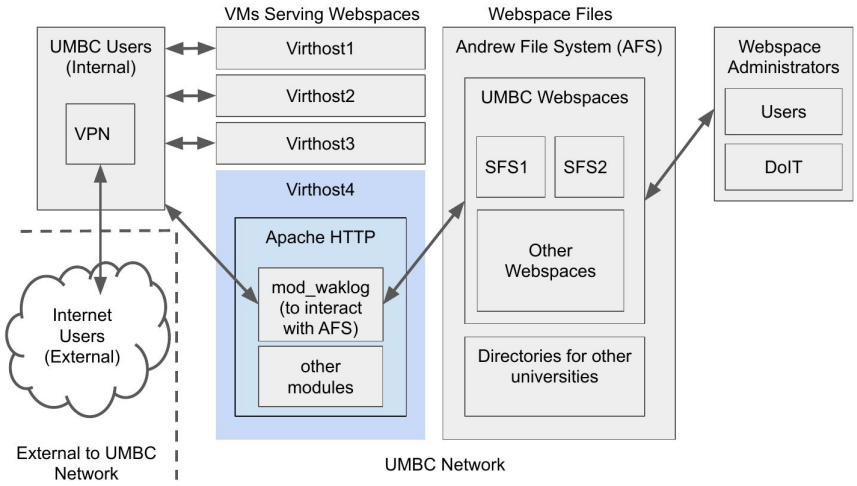

Figure 3: Architectural diagram of UMBC's Virthost system for fielding faculty and student organization webpages. It interfaces with UMBC's Andrew File System (AFS) mode, Apache HTTP running mod_waklog, and the SFS1 and SFS2 webspaces created for the study. Webspace administrators upload files, including HTML documents, scripts, and images, which Apache serves to page visitors. Arrows depict communication paths.

Scope

The 2019 SFS Study's goal was to evaluate the security of Virthost. We included two test AFS webspace directories (`sfs1` and `sfs2`) and all public directories on UMBC's AFS volume. We also evaluated mod_waklog (including its sourcecode), due to its manageable size and its importance for Virthost. We did not consider the sourcecode of AFS, Apache, or any other software. Also out of scope were attacks against AFS nodes or volumes that did not belong to UMBC, physical attacks of any kind, or social engineering or other forms of coercion.

Adversarial Model

We assume the adversary possesses credentials for at least one user with read-modify permissions for the Virthost webspace `sfs1.umbc.edu`, so they can read, modify, and create files and scripts that Virthost

will serve or execute. These credentials do not have administrator privileges on UMBC computing systems. The adversary must be on the campus network, either physically or via a VPN. The adversary understands the software and configuration of Virthost and the structure and permissions of AFS. A major goal of the adversary is to exfiltrate Virthost's keytab file containing AFS secret keys. We assume cryptographic primitives without public flaws are secure. The adversary can identify and use existing exploits against any Virthost component, but will not develop any zero-day exploits. The adversary's initial goal is to attack the `sfs2.umbc.edu` webspace, either through a direct attack originating from `sfs1` or by compromising Virthost. If the adversary is able to gain access to the underlying Virthost machine, they will try to escalate privileges and establish a long-term presence.

How We Analyzed the System

DoIT took one of the live Virthost servers offline, initialized the sfs1.umbc.edu and sfs2.umbc.edu test webspaces, and provided this instance for the security evaluation. DoIT gave each of the webspaces their own AFS directories, which included administrator privileges for the study. To coordinate and consolidate findings, the UMBC SFS cohort created a shared Google Drive directory. Within the SFS Google Drive directory, different groups of students maintained documents detailing their team's investigations and findings. Towards the end of the week, the group drafted a master document with notes on all discoveries and recommendations. To conduct the evening sessions, SFS scholars created a dedicated Slack instance. Communication with DoIT staff took place asynchronously through Slack and synchronously during daily afternoon meetings.

Potential Vulnerabilities, Attacks, and Recommendations

1. Within a compromised webspace, an adversary can upload and execute arbitrary scripts that usurp Apache's privileges to read, write, and modify Apache owned files, establish network connections, and run programs including compilers, packet analyzers, reverse shells, and kernel exploits. As shown in Figure 4, exploiting a configuration blunder unwisely granting Apache read access to mod_waklog's keytab file, students used a reverse shell to exfiltrate this critical file, thereby recovering cryptographic keys enabling them to gain full access to other webspace AFS directories. Although DoIT quickly fixed this issue, there remains an opportunity to exfiltrate the keytab file while Apache HTTP loads, during which the file is accessible to the Apache user. To mitigate remote attacks and reconnaissance, the permissions and capabilities of the Apache user should reflect a "least privilege" approach: the Apache user should not be able to read any unnecessary files, compile and execute arbitrary software, or initiate arbitrary outbound connections.
2. Apache HTTP and mod_waklog configurations are globally readable, allowing any webspace owners and the Apache user to read them. The adversary can read server configurations to search for exploitable weaknesses as part of their reconnaissance.
3. Virthost implements no countermeasures to well-known individual or distributed denial-of-service (DoS/DDoS) attacks, even though every webspace request results in network file system operations, and attacks such as Slowloris and SSL renegotiation attacks are capable of exhausting the server's resources. Instead, Apache should use the existing modules that mitigate common DoS techniques and use best practice configurations that require adversaries to expend more resources to deny Virthosts service.

4. Using legitimate certificates and domains, compromised webspaces can serve malicious content or steal credentials by tricking UMBC users into communicating with authentication services without encryption. One partial mitigation is to require end-to-end encryption (e.g., HTTPS) through the use of HTTP Strict Transport Security, which forces HTTPS connections for all users.
5. CentOS 7 runs the 3.10.0 Linux kernel, which is vulnerable to exploits, including Meltdown, which allows an adversary to exfiltrate server memory, including cryptographic keys, by exploiting a flaw in the CPU. To mitigate this vulnerability, Virthost should be updated to a modern kernel that can prevent Meltdown and similar exploits.

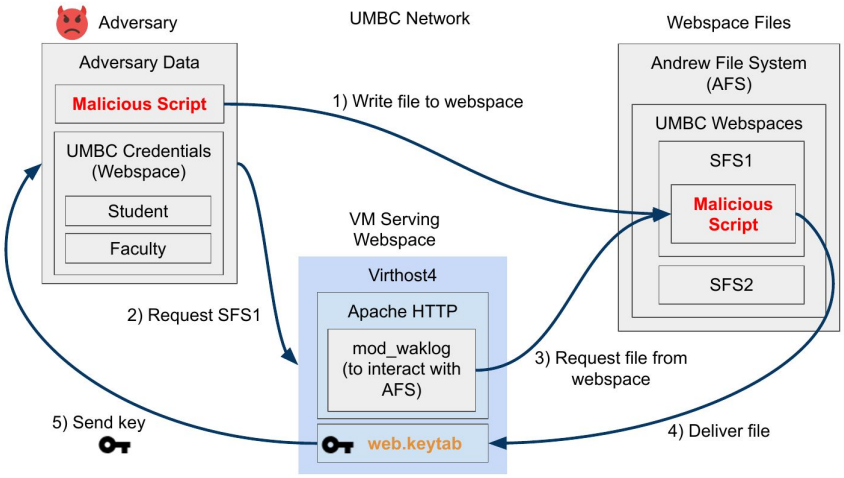

Figure 4: Attack in which an adversary uploads a malicious script to the SFS1 webspace and requests it, causing the script to execute with Apache user permissions. The script fetches an Apache-owned web.keytab file, containing AFS secret keys for SFS1, SFS2, and any other webspaces running on Virthost4, and returns this file to the adversary.

Authors

Cyber Defense Lab, University of Maryland, Baltimore County (UMBC), Baltimore, MD 21250

Enis Golaszewski <golaszewski@umbc.edu>
Alan T. Sherman <sherman@umbc.edu>
Linda Oliva <oliva@umbc.edu>
Michael R. Bailey <mbaile57@montgomerycollege.edu>
Scott Bohon <bohons1@umbc.edu>
Cyrus Bonyadi <cbonyad1@umbc.edu>
Casey Borror <cborror1@umbc.edu>

Ryan Coleman <ryancoleman@umbc.edu>
Johannah Flenner <johco1@umbc.edu>
Elias Enamorado <eenamor1@umbc.edu>
Maksim E. Eren <meren1@umbc.edu>
Mohammad Khan <khanmoh1@umbc.edu>
Emmanuel Larbi <elarbi1@umbc.edu>
Kyle Marshall <kmarsha2@umbc.edu>
William Morgan <wmorgan1@umbc.edu>
Lauren Mundy <lmundy1@umbc.edu>
Gabriel Onana <gonana1@umbc.edu>
Selma Gomez Orr <sorr1@umbc.edu>
Lauren Parker <park31@umbc.edu>
Caleb Pinkney <calebpin@umbc.edu>
Mykah Rather <mrather1@umbc.edu>
Jimmy Rodriguez <jrodr233@umbc.edu>
Bryan Solis <bsolis1@umbc.edu>
Wubnyonga Tete <wtete1@umbc.edu>
Tsigereda B. Tsega <ttsega1@umbc.edu>
Edwin Valdez <evaldez2@umbc.edu>
Charles K. Varga <cvarga1@umbc.edu>
Brian Weber <brianw5@umbc.edu>
Ryan Wnuk-Fink <wnukry1@umbc.edu>
Armand Yonkeu <ayonkeu1@umbc.edu>
Lindsay Zetlmeisl <lzetl1@umbc.edu>
Damian Doyle <damian@umbc.edu>
Jack Suess <jack@umbc.edu>

University of Minnesota Duluth, 1049 University Dr, Duluth, MN 55812

Peter A. H. Peterson <pahp@d.umn.edu>

Prince George's Community College, 301 Largo Rd, Largo, MD 20774

Casey O'Brien <obrien.casey@gmail.com>

Montgomery College Germantown Campus, 20200 Observation Drive, Germantown, MD 20876

Joseph Roundy <joseph.roundy@montgomerycollege.edu>