Proceedings of the 14th Colloquium for Information Systems Security Education
Baltimore Marriott Inner Harbor
Baltimore, Maryland June 7 - 9, 2010

Promoting Skepticism in the Security Classroom

Martin C. Carlisle and Dino Schweitzer, *United States Air Force Academy*

Abstract – Generally discussions of digital signatures, cryptography and computer security focus on the complicated technical details behind the systems. Students are often led to the false conclusion that such systems are truly secure. We describe a very simple Trojan horse attack on a Department of Defense digital signature system, and how its demonstration in the classroom led to an improved understanding of weaker links in the security trust chain, and a healthy skepticism of security claims.

Index terms – Digital signatures, usability, security, trusted computing

I. Introduction

A story is told of a computer security educator who was training his son. One day, he put his son on a ledge and told him to jump into his arms. When the son said "but, Papa, it is too high, if I fall to the ground I will be hurt," he replied "Son, I am your father and love you very much - I will catch you." The boy jumped, the father made no attempt to catch him, and he crashed to the ground. The man turned to his crying son and said "The lesson my son is never trust anyone."

As educators, we do not want to be hurting our students by convincing them to jump off security ledges. On the other hand, we want to instill a healthy sense of skepticism about trusting systems that make claims of security, especially when the evidence is something as simple as an icon. Does *https://* really mean you can enter your credit card information and trust it is secure? This paper describes a simple lesson taught to students based on a vulnerability discovered in the Department of Defense (DoD) implementation of digital signatures. The ledge was not very high.

II. BACKGROUND

Information security is an important topic in educating all students, not just computer scientists or IT professionals. In today's world, everyone is a user and vulnerable to attacks, whether it is for monetary gain, access to information, or simply as an unwitting participant in an army of remotely controlled bots. It is especially important for future developers and keepers of the technology to understand vulnerabilities, assess risk, and be able to thwart attacks.

As an indication of this importance in education, the Association of Computing Machinery (ACM) listed security issues and principles as a key knowledge area for all disciplines of computing education in their 2005 Curricula Report [1]. Within the federal government, Presidential directives, policy reviews, federal reports, white papers, and a frenzy of media hyperbole have provided a constant barrage of reminders as to the critical importance of understanding and managing security to the safety of the nation.

To provide this knowledge of security principles, many educators have proposed curricula and teaching approaches for reaching students across different disciplines. Many security educators have promoted an active learning hands-on approach through laboratories, competitions, and test beds [2,3,4,5,6]. The need for ethical training, isolated labs, and virtualization often accompany these recommendations.

In addition to the necessary knowledge about security practice and principles, another important aspect of student education is their attitude toward security. Bruce Schneier, well-known author and pundit in computer security, identifies the need for skepticism and sensibility in security [7]. Other educators have suggested instilling this in students during their education is an important lesson [8].

At our institution, we teach three courses in computer security: Computer Security, Network Defense, and Crtyptography. The Computer Security class is required of all computer science majors and covers fundamental concepts, both theoretical and applied, in computer security and information warfare. The course consists of a strong hands-on component including regular laboratories, a competition with other schools, and web-based active learning tools. In addition, we strive to influence attitudes toward security, both at school and in their future careers.

We chose to conduct a lesson based on a discovered flaw in digital signatures. The remainder of this paper will

Academy Center for Cyberspace Research
2354 Fairchild Dr, Suite 6G101
United States Air Force Academy, CO 80840
(719) 333-3590
{martin.carlisle,dennis.schweitzer}@usafa.edu

ISBN 1-933510-99-4

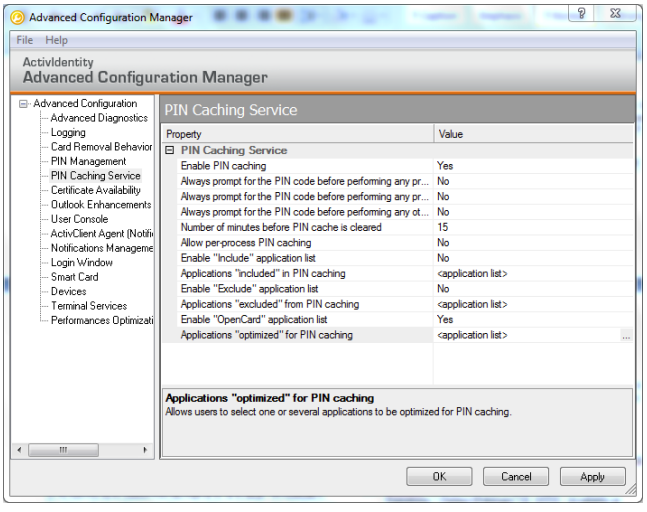
describe the flaw, how it was exploited in the classroom, and the discussions that followed.

III. The Vulnerability

Attacks on digital signatures are not new. As early as 2000, Bruce Schneier [9] noted that computers perform electronic signatures, not people, and without a trusted computing platform, "Alice's digital signature doesn't really mean anything about Alice's intentions."

Spalka et al [10] described attacks based on capturing the PIN, intercepting the data between the computer and the smart card, and tampering with the display or the data along with possible defenses. Jøsang and AlFayyadh [11] examined attacks and defenses based on formatting of the document to be digitally signed. Balacheff et al [12] described a scheme including a trusted display controller to ensure that the signer has knowledge of what they are actually signing. We have developed a much simpler attack vector based on PIN caching and automating Outlook.

The DoD uses ActivIdentity ActivClient [13] as a middleware to smart-card readers. We started with the default configuration provided by a service's Public Key Infrastructure (PKI) office. In this default configuration, the PIN Caching Service is enabled. This means that once you have typed your PIN, you are not prompted for it again until a certain time period has expired. The purpose for the PIN Caching service is to improve the usability of the software, by not requiring the user to enter the PIN repeatedly for consecutive operations. These settings are part of the Advanced Configuration Manager, shown below:



This shows a default configuration of 15 minutes before the PIN cache is cleared, and that PIN caching is for the machine as a whole, not per process. (Per process caching would mean that if a user digitally signs an email in Outlook, they won't be required to type their PIN again for future Outlook messages during the prescribed interval, but if they then attempt to open a CAC-authenticated website in Internet Explorer, they would be required to enter the PIN again).

The attack vector selected was the PIN caching. We assumed the user could be tricked into loading an executable onto their machine. This could be done via a Trojan horse, a browser vulnerability or some other mechanism. Then, that program simply waits for the user to send a digitally signed message, constructs its own digitally signed message and sends it during the PIN caching window. The targeted user receives no additional PIN prompt, nor any visible evidence that a second message has been digitally signed and sent. The message does appear in the users' sent items folder. We could have chosen to delete the message from that folder, but decided there would be a greater effect if the message remained there.

Microsoft has made it very easy to write programs that interact with Outlook using the Outlook Object Model [14]. Three lines of code connect a program with the currently running Outlook instance:

```
app = new Outlook.Application();
ns = app.GetNamespace("MAPI");
ns.Logon(null, null, false, false);
```

From there, we can access the sent items folder:

```
folder = ns.GetDefaultFolder(
  OlDefaultFolders.olFolderSentMail);
```

This folder has an "items" property referencing all of the messages. We loop through these in reverse order, until we find a signed message, or the messages are over a minute old. If we do not find a sufficiently recent signed message, then we sleep for a few seconds, then try again.

Determining whether a message has been digitally signed (and later digitally signing our constructed message) was the most challenging. The Outlook Object Model does not directly support digital signatures [15]; however, we were able to find a MAPI property that we could check and set on mail items [16]. The code below examines the i^{th} message in folder and the variable "s" holds whether or not that message has been digitally signed.

```
const String PR_SECURITY_FLAGS =
"http://schemas.microsoft.com/mapi/proptag/0x6E010003";
const int SECFLAG_SIGNED = 0x02;
item = (MailItem) folder.Items[i];
int l = (int)
item.PropertyAccessor.GetProperty(
   PR_SECURITY_FLAGS);
```

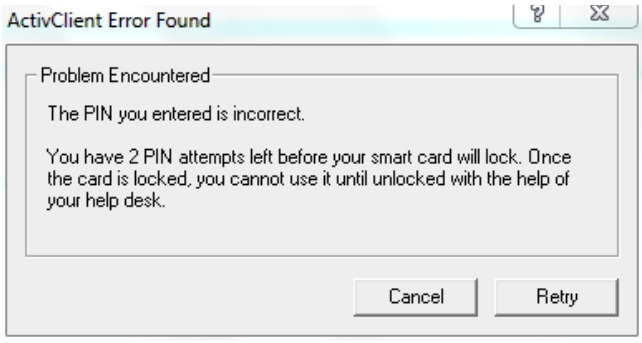
```
int s = l & SECFLAG_SIGNED;
```

Once we have found a sufficiently recent digitally signed message, we then construct a new message, add a recipient, message (with attachment if desired), set the digitally signed property and send it.

```
newMail = (MailItem)
   app.CreateItem(olMailItem);
recipient = newMail.Recipients.Add(
   "recipient@somewhere.mil");
recipient.Resolve();
newMail.Subject = "test";
newMail.Body = "test body";
int l = (int)
newMail.PropertyAccessor.GetProperty(
   PR_SECURITY_FLAGS);
l = l | SECFLAG_SIGNED;
newMail.PropertyAccessor.SetProperty(
   PR_SECURITY_FLAGS, l);
newMail.Attachments.Add("c:\\...pdf",
  olByValue, 1, "name");
newMail.Send();
```

One thing that is striking about this attack is the very few lines of code required to accomplish it. The entire program required only about 100 lines of code.

Since we are automating Outlook directly, switching to per process caching will not stop this from occurring. If PIN caching is disabled entirely, the user would see a second prompt for the PIN occur very soon after the first. We suspect most users would assume that had incorrectly typed their PIN or there was some glitch in the system and simply type it again, even though the incorrect PIN dialog did not appear:



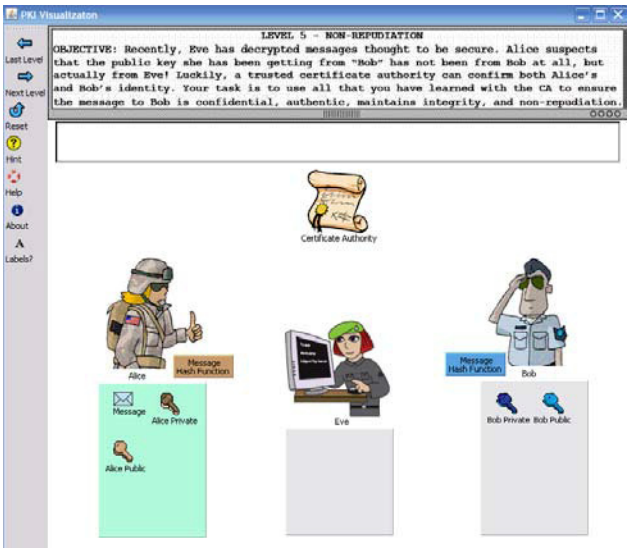
We are able to check and see whether PIN caching is enabled by a simple registry check. If our program has sufficient privileges (which we can also check), we can simply turn on PIN caching by changing a registry entry, and then turn it back off again when our attack has completed. No reboot or restart of Outlook is required for the registry change to have an effect.

IV. CLASSROOM DEMONSTRATION

Rather than simply present the vulnerability to the students in the security class as part of a lecture on digital signatures, we wanted to provide a more impactful lesson. The goal was for them to experience the flaw first-hand without foreknowledge and then use the (hopefully) surprising experience as a basis for discussion on general security issues. In other words, we had to set them up.

To set the stage we began a general lecture on PKI and digital signatures. Since the students were already familiar with the basics of PKI from a conceptual point of view, we focused the lecture on the role that digital signatures and certificate authorities played in the process, and specifically how digital signatures worked at our institution. In order for them to experience the vulnerability on their laptops without prior knowledge, it was necessary to get them to unwittingly load and run software on their machines that contained the "payload."

We took advantage of an interactive visualization tool on PKI written by students in a computer security course [17]. The purpose of the tool, PKIVis, is to introduce students to the concepts of PKI in a game-like environment in which the students need to accomplish certain tasks in order to send messages between Alice and Bob without the evil eavesdropping Eve getting the message. The screen shot below shows the basic elements of the tool.



We have used the tool in our entry-level computer science class to introduce the concepts of encryption, public and private keys, and PKI. The tools goes through five levels of tasks to accomplish, each level supporting one or more of the information assurance goals:

1. Confidentiality
2. Authentication

3. Authentication and Confidentiality
4. Integrity and Confidentiality
5. Non-repudiation

To accomplish a given task, the user drags and drops icons in the tool to the appropriate location. For example, to achieve confidentiality in level 1, the user would first drag Bob's public key onto the message to encrypt it; next drag the encrypted message to Bob; and finally drag Bob's private key onto the message to decrypt it. If a wrong move is made, an error message is generated, or if it allows Eve to compromise the message, a visual indication of disaster is displayed.

We informed the students in the security class that we wanted to use the tool at level 5 to discuss certificate authorities, digital signatures, and non-repudiation. The students downloaded the tool from the course website (containing the malicious payload), and executed it during class. We worked through the individual steps necessary to digitally sign and send a message, how to use the hash function and certificate authority, and what could happen if things were not accomplished in the correct order. Students experimented with the tool and followed along in class. Unbeknownst to them, the malicious code was now running on their machine.

Next, we began discussing how digital signatures and certificate authorities were implemented at our institution. Students were shown how to digitally sign an email and send it. As a demonstration of the difference between signed and unsigned messages, we had each student send themselves two messages, one unsigned and one signed. We discussed the differences in the messages as they were received, and what that should indicate to them about the degree of confidence they should have that the email actually came from them and was not a spoof from someone else. The stage was set.

The students were all convinced from the lecture that if they received a signed email from someone, they could be assured the message actually came from that person. However, using the attack vector previously described, when our students sent themselves a signed message, the malicious code running on their machines sent a second signed message to the instructor informing him that they would bring a dozen donuts to the next class. Students had no indication that a second email, let alone a signed one, had been sent. As the coup de grace, the instructor brought up his email, showed them his inbox, and thanked all of the students for offering to bring donuts. Since he could show them that the emails were signed, they were held by non-repudiation (just discussed) and obligated to follow through on their promise of donuts.

The initial reaction of the students was confusion. This was furthered when some of the students checked their sent items and found their computers reported that they had indeed sent the message! The brighter ones recognized that they had been had and were immediately suspicious of the code they had downloaded and executed, but could not explain the fact that the emails were signed. This led to a discussion of how the attack was accomplished and the vulnerability associated with PIN caching. More importantly, this exercise was used as a seed for discussing the broader issues of trust, security versus usability, social engineering, and the fact that security is relative.

As an issue of trust, students had to rethink the meaning of a signed message. What did it really mean? Not, as they had initially assumed, that the message actually came from the purported sender, but rather simply that the message had been signed by the sender's private key. This led to a discussion of ways that someone could get a message signed without the sender's knowledge, such as the attack executed. The question of whether or not students should trust the certificate authority and the implementation of digital signatures at our institution was also raised. While this particular attack did not try and exploit those, the students were much more skeptical about the whole process. Finally, the issue of trusting the instructor and downloaded courseware was discussed in detail. Just because an instructor claimed something was safe, what was their level of confidence that it really was? Many claimed that after this exercise, their level of trust had diminished--perhaps a lesson learned. In the future they may bring the same level of skepticism to a claim that a system is secure.

The second major concept discussed was the tradeoff of security and usability. The whole purpose of PIN caching is to make it easier on the user not to have to constantly re-enter it. However, as clearly demonstrated, that leads to a vulnerability. Other examples of the tradeoff were discussed. Many students claimed that if PIN caching had been turned off, they probably would have blindly entered their PIN anyways. One student, who was running an additional security check on his machine was actually warned that an email was attempting to be sent from his machine without his knowledge. He assumed it was something normal, and approved it! (A more sophisticated attack could have prevented this dialog from occurring and/or answered it for the user, as shown by Carlisle and Studer [18])

The social engineering aspects of the exercise were also discussed. While the scenario was pretty much guaranteed success given the instructor / student roles, many students recognized that they were very gullible to social engineering attacks if properly crafted. Would the same exercise have been successful, if it were a classmate who had encouraged them to download and execute a program? Most agreed it would. In fact, we have

recently seen examples of students sending links to simple Trojan horses that change the user's background, Start button icon, etc. Many students were taken in by emails claiming the links pointed to collections of pictures, or other innocuous content.

Finally, the relativity of security was discussed. Famous examples, such as the Vigenere cipher and Enigma machine, were brought up to demonstrate that a secure process is only secure until proven otherwise. Absolute security does not exist. Students were under the mistaken assumption that a signed email meant something more than it really does. How much do they really believe security claims? How skeptical should they be? Hopefully, this lesson has contributed to a greater level of healthy skepticism.

V. Conclusions

Students are often overly impressed with the mathematics behind cryptography, digital signatures and other computer security topics. They fail to understand that a system is only as secure as its weakest link, which may be an untrustworthy computer or an overly trusting user.

By creating a simple mechanism for creating digitally signed emails without the sender's knowledge and then surprising students with it during a lecture, we generated a crisis moment that caused students to reevaluate their security beliefs and assumptions. How could it be that their instructor just received a digitally signed message from them that they never sent? Worse yet, how could they then find that message in their sent items? This demonstration caught their interest in a much more significant way than simply presenting the possibility of attacks. Several students commented that this was the best lecture they had all semester and had really caused them to rethink their assumptions.

This exercise led to very profitable discussions on trust, security vs. usability tradeoffs, and the proliferation of social engineering attacks such as spear phishing. These discussions may have long-lasting impacts as these students go on to configure networks and procure and use computer security services.

VI. References

[1] Association of Computing Machinery, Computing Curricula 2005, www.acm.org/education/curricula.html, retrieved February 2010.

[2] Abler, Randal T., Contis, Didier, Grizzard, Julian B., and Owen, Henry L., "Georgia Tech Information Security Center Hands-On Network Security Laboratory", IEEE Transactions on Education, vol. 49 no. 1, February 2006, pp. 82--87.

[3] Crowley, E. 2004. Experiential learning and security lab design. In *Proceedings of the 5th Conference on Information Technology Education* (Salt Lake City, UT, USA, October 28 - 30, 2004). CITC5 '04.

[4] Mattord, H. J. and Whitman, M. E. 2004. Planning, building and operating the information security and assurance laboratory. In *Proceedings of the 1st Annual Conference on information Security Curriculum Development* (Kennesaw, Georgia, October 08 - 08, 2004).

[5] Conklin, A. 2005. The use of a collegiate cyber defense competition in information security education. In *Proceedings of the 2nd Annual Conference on information Security Curriculum Development* (Kennesaw, Georgia, September 23 - 24, 2005).

[6] Schweitzer, D., Gibson, D., Collins, M. "Active Learning in the Security Classroom," Proceedings of the Hawaii International Conference on System Science, HICSS-42 , January 2009.

[7] Schneier, B., Beyond Fear, Thinking Sensibly about Security in an Uncertain World, Copernicus Books, 2003.

[8] Fagin, B., Baird, L., Humphries, J., Schweitzer D., "Teaching Information Security with Skepticism and Critical Thinking," Proceedings of the 11th Colloquium for Information Systems Security Education, June 2007.

[9] Schneier, B. (2000). "Why Digital Signatures are not Signatures." Online [February 18, 2010]. Available at http://www.schneier.com/crypto-gram-0011.html#1.

[10] Spalka A., Cremers, A.B., and H. Langweg. The Fairy Tale of What You See Is What You Sign – Trojan Horse Attacks on Software for Digital Signatures. In IFIP Working Conference on Security and Control of IT in Society-II (SCITS-II) Bratislava, Slovakia, 2001.

[11] Jøsang, A. and AlFayyadh, B. 2008. Robust WYSIWYS: a method for ensuring that what you see is what you sign. In *Proceedings of the Sixth Australasian Conference on information Security - Volume 81* (Wollongong, NSW, Australia, January 01 - 01, 2008). L. Brankovic and M. Miller, Eds. ACM International Conference Proceeding Series, vol. 328. Australian Computer Society, Darlinghurst, Australia, 53-58.

[12] Balacheff, B., Chen, L., Plaquin, D., and Proudler, G. 2001. A trusted process to digitally sign a document. In *Proceedings of the 2001 Workshop on New Security Paradigms* (Cloudcroft, New Mexico, September 10 - 13, 2001). NSPW '01. ACM, New York, NY, 79-86. http://doi.acm.org/10.1145/508171.508184.

[13] Activ Client Active Identity. Online [February 19, 2010]. Available at: http://www.actividentity.com/ products/securityclients/ActivClient.

[14] Microsoft. *Outlook Object Model Overview.* Online [February 19, 2010]. Available at: http://msdn.microsoft.com/en-us/library/ms268893.aspx.

[15] "Programmatically Enable Encryption and Digital Signature." Online [February 19, 2010]. Available at: http://www.eggheadcafe.com/software/aspnet/32938983/programmatically-enable-e.aspx.

[16] Dvespa. "How to sign or encrypt a message programmatically from OOM." Online [February 19, 2010]. Available at: http://blogs.msdn.com/dvespa/archive/ 2009/03/16/how-to-sign-or-encrypt-a-message-programmatically-from-oom.aspx

[17] Ebeling, D. and Santos, R. "Public Key Infrastructure Visualization." J. Computing Science in Colleges, vol. 23, no. 1, October 2007.

[18] Carlisle, M. and Studer, S. "Reinforcing Dialog-Based Security." Proceedings of the 2001 IEEE SMC Information Assurance Workshop, West Point NY, June 2001.