

# Automatic Creation of Fine-Grained Vulnerable Windows System for Penetration Testing Education

Arati Banerjee, Cliff Zou and Damla Turgut  
Department of Computer Science  
University of Central Florida

## Abstract

In the face of the increasing needs of cybersecurity professionals from US public and private sectors, many universities have created various cybersecurity education programs. Penetration testing, as a critical component in cybersecurity training, often requires setting up virtual machines (VM) with various vulnerabilities. However, it is usually time-consuming and technically difficult to fine tune vulnerabilities in VM systems. In this paper, we present an automatic security patch removal tool that can fine tune various Windows VM systems to precise levels of vulnerabilities, and easily employed by students and educators alike. This tool can create virtual machines that simulate different security states in the Windows operating system timeline and gives a more realistic view of the every-changing state of cybersecurity to the students pursuing an education in the field.

## 1. Introduction

In the current Internet-connected world, most companies, government agencies, and ordinary people heavily rely on the cyber world for information and data management, processing, and exchange. Information leakage and data breaches become increasingly damaging to business, government and people's life, and hackers have more incentive to attack for financial and political gains. In recent years, we are facing an increasing trend of major security breaches and incidents happening in every aspect of our society, including security breaches to major retailers such as Target Corp. in 2013 (Rosenblum, 2014), data breach to government agencies such as Office of Personnel Management in 2015 (Davidson, 2015), data breach to major banks such as the JPMorgan Chase (Crowe, 2015), and data breach to major health insurance companies such as Anthem Inc. (Riley, 2015), etc.

Facing such continuous and costly cyberattacks, there is a tremendous need of education programs on educating and training students and professional capable to conduct cybersecurity analysis and investigation. Penetration testing labs and experiments are best conducted on targets with various vulnerabilities. These computer systems can take the form of virtual machines (VM), to cut down on the cost of obtaining and maintaining additional hardware. However, not many readily available vulnerable VM systems exist. Additionally, it is time-consuming and technically difficult to fine tune vulnerabilities in those systems. For example, to set up Windows XP system as penetration testing target, we only have the WinXP VM with service pack 2, service pack 3, and fully security-patched versions to use. This inevitably misrepresents the gradual state of security of Windows systems over time.

In this paper, we present an easy-to-use and automatic approach to remove security patches in order to create fine-grained vulnerable Windows systems for cybersecurity educators. In the real world, Windows systems do not implement a sweep of security patches all at once. As each

vulnerability is discovered, a new security patch will be released accordingly. Given this reality, cybersecurity educators need to provide a virtual Windows VM system with a fine grain of vulnerabilities to better echo the ever-changing levels of vulnerabilities and security patches in the world. We sought to create an automatic tool able to produce virtual machines that simulate different points in the Windows operating systems life cycle when security patches were implemented. This tool removes the most recent security patches up to a defined point, leaving the system vulnerable to recently-discovered attacks. It uses non-invasive, system-provided methods to ensure that no vulnerabilities are artificially introduced through the process. This essentially returns the system to a state before the vulnerabilities were patched, simulating the system that attackers at that time had access to. This allows students to study the system and exploit the vulnerability with the benefit of hindsight. With the ability to fine-tune the system to various levels of security, educators are able to provide a more realistic and accurate penetration testing target system.

The rest of the paper is organized as follows. We describe work related to this tool in Section 2. In Section 3 we present the detailed design and procedure of the automatic security patch removal tool developed for Windows XP and later Windows systems. Finally, we conclude the paper in Section 4.

## **2. Related Work**

There is a stark skill shortage in the field of cybersecurity (Conklin, Cline, & Roosa, 2014). Educational programs that focus on cybersecurity are often splintered along program lines (for example, information technology, computer science, computer information science, etc.) (Conklin, Cline, & Roosa, 2014). In addition, the majority of those who study cybersecurity go on to work in industry rather than working in research or higher education (McGettrick, 2013). This results in fewer accredited experts available to teach in the field of security. Education in the field of cybersecurity suffers from natural difficulties because of these factors. Any tool that can ease the way of students and educators alike to explore cybersecurity is vital to meeting the demands of a more interconnected world than ever.

Offering courses in cybersecurity is an effective way to educate about the theoretical aspects of security. However, a workshop of recognized experts run by ACM's education board determined without a doubt that studying theory is not enough to prepare a potential cybersecurity professional (McGettrick, 2013). In the light of the shift of infrastructure from physical to electronic control and the rising number and severity of cyber-attacks, the DETER project was created (Mirkovic, et al., 2010). The DETER project was funded by the United States Department of Homeland Security, the National Science Foundation, and several other institutions (Mirkovic, et al., 2010). The project aimed to create resources for cyber security experimentation in the form of the DETER cyber security testbed (Mirkovic, et al., 2010). However, the testbed was difficult to apply to education, because of how difficult it was to use (Mirkovic, et al., 2010). It also struggled to create an environment of realism for rising security professionals (Mirkovic, et al., 2010). The goal of this paper is to offer a simpler tool that can better emulate the ever-changing environment of cybersecurity.

Learning how to create secure systems is key and cannot be done without learning how to exploit vulnerabilities. Fortunately, there are a vast number of exploitative tools available to run penetration testing - a controlled attack to test the security of a system. Kali Linux is an operating

system with a suite of penetration tools (Hayajneh, Denis, & Zena, 2016). Wireshark, and Nmap are valuable tools that can be used to gather information necessary for gathering data necessary to attack a system (Hayajneh, Denis, & Zena, 2016). Metasploit is commonly used to attack a system (Hayajneh, Denis, & Zena, 2016). The tools are easily available to attempt attacks, but what about systems to experimentally exploit? Most educators use available virtual machines to teach the foundations of exploitation, as the main vulnerability in a system is generally its own operating system (Hayajneh, Denis, & Zena, 2016). However, these operating systems serve as a snapshot of the operating system in a singular state. Since the security of an operating system changes with each update, it would better imitate the evolution of security if there was a tool to easily roll back a virtual machine to states before major security patches and follow the timeline of an active operating system.

### **3. Automatic Security Patch Removal**

Different versions of Windows operating systems could have very different architectures and kernel programs. Thus we need to design the automatic security patch removal tool according to the targeted operating system. In this paper, we first design the patch removal tool and process for Windows XP. Because Microsoft stopped supporting Windows XP and its VM in 2014 (Microsoft, 2014), we have also designed and developed the automatic security patch removal tool targeting Windows 7 and later Windows operating systems, since they handle security updates differently.

#### **3.1. Motivation and Objective**

The objective for this project is to create an easily-implemented tool that can set a virtual machine to various vulnerable states based on the timeline of that operating system, along which vulnerabilities were discovered and patched. The first studied and proposed timeline is for Windows XP. Before stopped supporting Windows XP virtual machine (Microsoft, 2014), Microsoft provided three specific states of XP virtual machines: without any service pack, with service pack 2, and with service pack 3. That means we can set up Windows XP VMs with three different sets of vulnerabilities. But in the real world, the OS vulnerability situation is much more complex: for the same OS with the same service pack, different users may have updated the security patches on their computers in different ways. Some users may never install security patches, some users may manually install security patches and thus their computers have not been patched for the last two months, while others may automatically update their security patches.

Therefore, to better train students dealing with real-world penetration testing or security hardening, educators would have to painstakingly and selectively remove security updates in reverse chronological order until they had reached the desired timeline point to create the corresponding vulnerable VM. This process, in addition to being time-consuming, is made more difficult by the built-in expiry in Windows virtual machines. For example, Windows XP VM provided by Microsoft will expire just 30 days after installation. This setup task, therefore, would have to be undertaken repeatedly for each state of security and each time the virtual machine expires. This makes penetration testing at finer grains of security nearly impossible or too time-consuming to set up.

The proposed tool is to automate the process of removing security updates up to a specific key point along the timeline. This is feasible because we can obtain the accurate patch timeline

information from an OS provider. For example, Microsoft Security Bulletin provides the complete list and explanation of its released security patches for Windows OS (Microsoft Security Bulletins, n.d.).

Security Bulletin ID	MS08-067	MS09-001	MS10-018	MS10-046
Date of Patch	10/23/08	01/13/09	03/09/10	08/02/10
KB # of Patch	KB958644	KB958687	KB980182	KB228619

*Figure 1: A timeline of Windows XP security updates*

Let us use Windows XP service pack 3 Operating System as an example. Fig. 1 shows four well-documented security updates of Windows XP that patched critical vulnerabilities, according to the Microsoft Security Bulletins. we can create the following five different VMs with different sets of vulnerabilities by applying our proposed security patch removal tool on a fully-patched Windows XP VM to reversely remove all security patches after a specific time:

- VM 1: has all four vulnerabilities (after removing all security patches whose KB number is equal or larger than KB958644)
- VM 2: is secure against MS08-067 related attacks, but vulnerable to attacks against the other three vulnerabilities (after removing all security patches whose KB number is larger than KB958644)
- VM 3: is secure against MS08-067 and MS09-001 related attacks, but vulnerable to attacks against MS10-018 and MS10-046 (after removing all security patches whose KB number is larger than KB958687)
- VM 4: is vulnerable to MS10-046 related attacks, but secure against all other three attacks (after removing all security patches whose KB number is larger than KB980182)
- VM 5: is secure against all four attacks (without removing any security patches)

Windows XP has more than 300 critical vulnerabilities (Windows XP Vulnerabilities, n.d.). Although some vulnerabilities can be secured with a single security patch, we can still create in the scale of hundreds of different versions of vulnerable WinXP VMs.

### **3.2. Windows XP**

Because of the wide usage of Windows XP for a long time in the past, and its many well-documented security exploits, Windows XP is often the primary target in penetration testing education. In this section we first introduce our developed automated security patch removal tool for Windows XP virtual machine.

Work began on trying to find a way to remove updates in order. The first attempt was to remove them via a command line. This was unsuccessful because the Windows Update Standalone Installer is only available starting from Windows Vista. A seemingly close solution was to run a script that would run the uninstall .exe files in the system files of each update, identified and filed by their Microsoft Knowledge Base ID number. However, this only managed to reduce the workload on the user by one click per update, when compared to manual un-installation from the control panel. From here we considered simulating key presses or other user input via program script, but all potential solutions would be very complex and error prone since the large amount

of security updates in Windows XP have many different forms and behaviors when they are uninstalled. It is clear that this script, while a minor improvement, is not enough and robust for widespread use in cybersecurity education.

Upon further exploration of system files, we discovered that under the hidden system folder, C:\System Volume Information (Fig. 2), Windows XP stores all restore points that could be used by the System Restore Tool (Fig. 3). From this discovery, we took a different approach to creating the patch removal tool. If we created a series of restore points corresponding to different points in the Windows XP timeline, copying one of the specific restore point folders into the system folders of a Windows XP virtual machine, the corresponding timeline could be easily restored via using the System Restore Tool. Early tests of this solution were faced with unusual problems. For example, restore points would alter the certificates that ensured the virtual machine would expire, often shortening the lifespan of the virtual machine. With careful navigation of administrative access and system folder alteration, we finally successfully restored virtual machines to a predefined restore point.

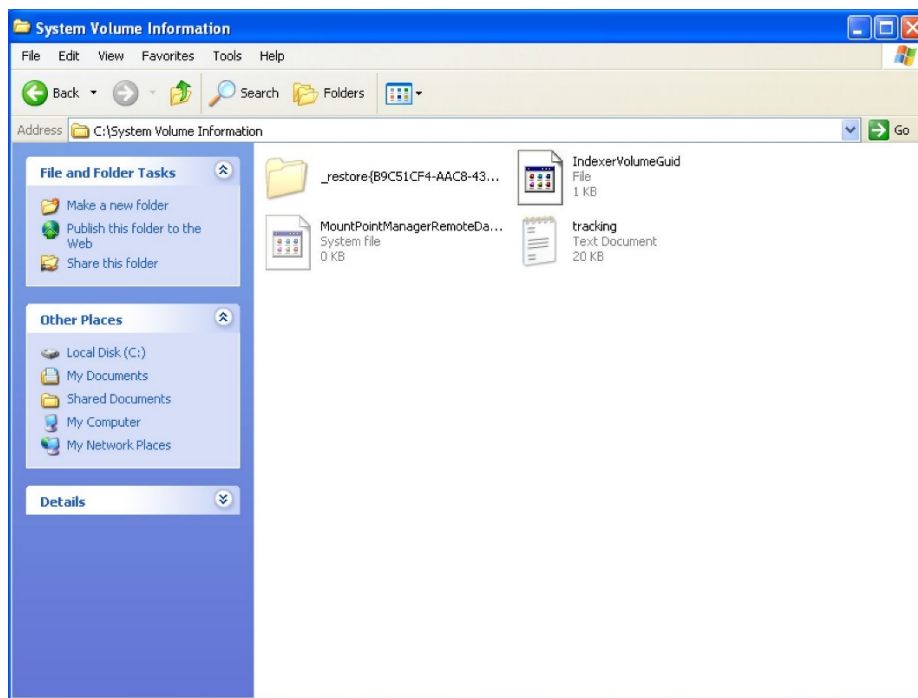


Figure 2: The hidden system folder containing restore points.

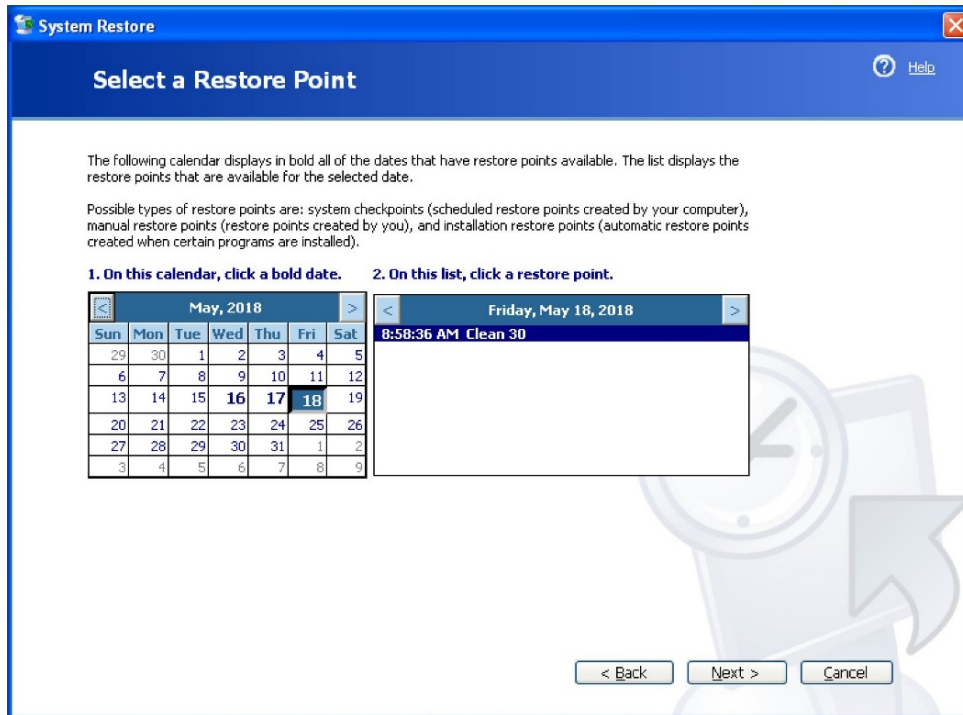


Figure 3: The system restore tool with the imported restore point used to reset the system to a point before all restore points

### 3.3. Windows 7 and Further

In 2014, Microsoft has stopped supporting Windows XP, and removed the XP virtual machine from its VM downloading repository (Microsoft, 2014). In addition, security professionals would benefit more from educating on active and widely deployed operating systems. For these reasons, we refocused our efforts on the newer OS system, a Windows 7 virtual machine. After investigating the system files of the new virtual machine, it became clear that the system files were protected even from administrators, and the file containing the restore points was beyond reasonable access. Additionally, the uninstall .exe files that Windows XP stored according to the KB numbers was not stored in Windows 7. Thus the previous approach used in Windows XP would not work on Windows 7.

However, Windows 7 does have the Windows Update Standalone Installer (wusa.exe). The installer could be launched from the command prompt, has commands for uninstalling updates, and GUI prompts could be suppressed. We experimented with writing a batch file that could uninstall the updates. After batch scripts failed, we switched to PowerShell, which is readily available on the Windows 7 virtual machine.

```

1 echo "Compiling a list of Security Updates"
2
3 wmic qfe get | where {$_-match "Security"} | %{$_.split(' ')[1]} > final.txt
4
5 $file = gc "final.txt"
6
7 if ($file.length -gt 4){ [array]::reverse($file) }
8
9 $file > next.txt
10
11 Remove-Item final.txt
12
13 $KBNum = Read-Host -Prompt "Enter Valid KB number to designate stopping point"
14
15 echo "Removing Security Updates"
16
17 foreach ($line in Get-Content .\next.txt)
18 {
19     cmd /c "wusa /quiet /uninstall /kb:$line /norestart"
20     if ($line -eq $KBNum) { break }
21 }
22
23 Remove-Item next.txt

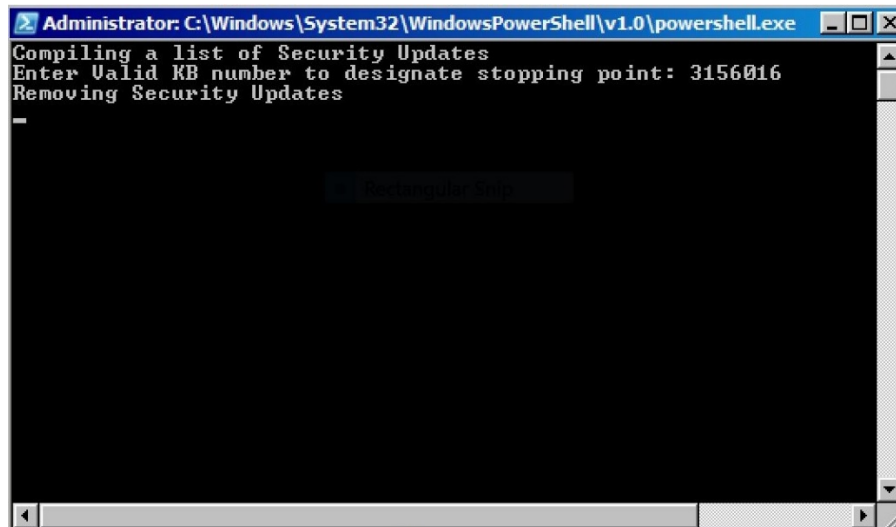
```

Figure 4: Formatted script, without comments or shutdown options.

Fig. 4 shows the bulk of the final script in our automated patch removal tool. The first course of action is to isolate the KB numbers of all already-installed updates (line 3 of Fig. 4). We opt to do this rather than work from a previous list. While using a pre-processed list of updates would have reduced processing time when running this script, generating a new list at the initial stage of the script would make the patch removal tool to function well on a variety of virtual machines, which we believe worth the cost of processing time.

The list of KB numbers is extracted using the *wmic* command, which is already provided by Windows. The list of updates is stored in a temporary file, *final.txt*. The first prototype failed to completely remove some updates, even after running through the script multiple times. In exploring this issue, we checked through the list of updates, and noticed that the KB number increased chronologically. However, when removing security updates, we have to uninstall the newest update first since a security update was programmed in most cases depending or correlating to previous older updates. After finding out this reason, we fixed the issue by simply reversing the list (line 7 of Fig. 4) and storing it in a new temporary file, *next.txt*. By uninstalling the updates in reverse chronological order, we prevented updates that were dependent on previous updates from being corrupted before they could be uninstalled.

Initially, we launched the *wusa* process from PowerShell, but the process tried to run multiple instances of *wusa* at the same time which is not permitted by the Windows System. Instead of inserting a pause, which seemed unreliable, depending on the update being uninstalled, we simply launched a command prompt to run the tool (line 19 of Fig. 4). This would force a wait until the process was completed before running the next instance. We then altered the script to accept user input to choose a stopping point in line 13 of Fig. 4, and looped through the updates, uninstalling them until reaching the given KB number or - if an invalid KB number was provided - until all security updates were removed.

A screenshot of a Windows PowerShell terminal window. The title bar reads "Administrator: C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe". The terminal text shows the script's progress: "Compiling a list of Security Updates", "Enter Valid KB number to designate stopping point: 3156016", and "Removing Security Updates". A single hyphen "-" is visible on the next line, indicating the start of a list of updates. The rest of the terminal is black with no visible text.

```
Administrator: C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
Compiling a list of Security Updates
Enter Valid KB number to designate stopping point: 3156016
Removing Security Updates
-

```

*Figure 5: The script in action.*

The current prototype can uninstall all security updates except for a few that are protected, presumably by Microsoft, after running the script three times with restarting the system after each run. The script will successfully stop removing patches at the given KB number. This means that an instructor (or similar expert) would be able to provide the KB number of the update which patched the desired vulnerability, often found on the security bulletins on the Microsoft developer website.

The script has been tested on Windows 7 and Windows 8.1 virtual machines with success, and initial research into the Windows 10 operating system looks promising - it handles updates in the same way as its tested predecessors, and the script does run successfully. However, because of the newness of Windows 10, there are few confirmed vulnerabilities that a full exploit could test at this time.

## 4. Conclusion

Cybersecurity education is struggling to keep up with the pace at which cybersecurity professionals are needed. This paper presents a flexible and efficient security patch removal tool that can help instructors provide better and faster penetration testing or cyber competition education. The current prototype serves as a promising proof of concept. The script could revert virtual machines by automatically uninstalling security patches, granular enough to fine tune a virtual machine to a desired point, workable on multiple versions of the Windows operating system, and easy to be used by a general user or instructor. The tool enables educators to reverse the clock on the security of Windows virtual machines with ease, making it possible to give students a clearer picture of the ever-changing state of security of a system. An educator could create and quickly configure virtual machines for each penetration testing or competition exercise with the proposed tool. Because of its use of scripts and system tools, the proposed tool can be used on both large and small scales.

## Acknowledgement



This work is supported by the National Science Foundation under grant DGE-1723587, REU Site grant CNS-1560302, and the grant from US Army PEO STRI.

## References

- Conklin, W. A., Cline, R. E., & Roosa, T. (2014). Re-engineering Cybersecurity Education in the US: An Analysis of the Critical Factors. *2014 47th Hawaii International Conference on System Sciences*, (pp. 2006-2014).
- Crowe, P. (2015). *Jpmorgan fell victim to the largest theft of customer data from a financial institution in us history*. Retrieved from <http://www.businessinsider.com/jpmorgan-hacked-bank-breach-2015-11>
- Davidson, J. (2015). *New opm data breach numbers leave federal employees anguished, outraged*. Retrieved from <https://www.washingtonpost.com/news/federal-eye/wp/2015/07/09/new-opm-data-breach-numbers-leave-federal-employees-anguished-outraged/>
- Free Virtual Machines from IE8 to MS Edge*. (n.d.). Retrieved from <https://developer.microsoft.com/en-us/microsoft-edge/tools/vms/>
- Hayajneh, T., Denis, M., & Zena, C. (2016). Penetration testing: Concepts, attack methods, and defense strategies. *2016 IEEE Long Island Systems, Applications and Technology Conference (LISAT)*, (pp. 1-6).
- Holik, F., Horalek, J., Marik, O., Neradova, S., & Zitta, S. (2014). Effective penetration testing with Metasploit framework and methodologies. *2014 IEEE 15th International Symposium on Computational Intelligence and Informatics (CINTI)*, (pp. 237-242).
- McGettrick, A. (2013). Toward Effective Cybersecurity Education. *IEEE Security Privacy*, 66-68.
- Microsoft Security Bulletins*. (n.d.). Retrieved from <https://docs.microsoft.com/en-us/security-updates/>
- Microsoft. (2014) Support for Windows XP ended. Retrieved from <https://www.microsoft.com/en-us/windowsforbusiness/end-of-xp-support>
- Mirkovic, J., Benzel, T. V., Faber, T., Braden, R., Wroclawski, J. T., & Schwab, S. (2010). The DETER project: Advancing the science of cyber security experimentation and test. *2010 IEEE International Conference on Technologies for Homeland Security (HST)*, (pp. 1-7).
- Riley, C. (2015). *Insurance giant anthem hit by massive data breach*. Retrieved from <http://money.cnn.com/2015/02/04/technology/anthem-insurance-hack-data-security/>
- Rosenblum, P. (2014). *The target data breach is becoming a nightmare*. Retrieved from <http://www.forbes.com/sites/paularosenblum/2014/01/17/the-target-data-breach-is-becoming-a-nightmare>
- Turgut, D., Massi, L., Bidoki, N. H., & Bacanli, S. S. (2017). Multidisciplinary Undergraduate Research Experience in the Internet of Things: Student Outcomes, Faculty Perceptions, and Lessons Learned. *2017 ASEE Annual Conference & Exposition Conference*, (p. 19 Pages).
- Windows XP Vulnerabilities*. (n.d.). Retrieved from [http://www.securiteam.com/products/W/Windows\\_XP.html](http://www.securiteam.com/products/W/Windows_XP.html)

