

Star Hacks, Episode V - Answers and Winners

digg this story

Discuss in Forums {mos_smf_discuss:May 06 - Star Hacks, Episode V: The Empire Hacks Back}

To all of my Jedi and Sith friends,

I am pleased to announce the winners of our "Star Hacks, Episode V: The Empire Hacks Back" Skillz H@ck1ng Challenge. It was indeed a heated battle, with brilliant Jedi* and Sith fighting it out for domination of that poor Windows 2003 box aboard the Millennium Falcon using your wiliest command-line kung fu. We received over one hundred entries, and I carefully analyzed each and every one of them, running every single command you guys had suggested for dealing with the vaderbot. Some of your commands were studies of brilliant simplicity, others funky and complex but workable, and some others just didn't do the trick. It was really hard to pick winners from so many really solid entries, but I worked very hard to find the best of the best. Several of you did not win, but deserve honorable mentions for your Jedi skills and cleverness. Note that in choosing the winners, I used Occam's Razor to narrow down the results; simpler but correct answers were preferred to the more complex ones.

Skillz Sponsored by Core Security Technologies

But, before we announce the winners, I'd like to give you my favorite answers for the challenge.

1) How can R2D2 kill all of the processes named "vaderbot.exe" with a single command?

There were many approaches to this one. My two favorites were:

```
C:\> wmic process where name="vaderbot.exe" delete
```

```
C:\> taskkill /f /im vaderbot.exe
```

2) Unfortunately, as the last vaderbot.exe process is about to be killed, it spawns a group of new Vader Bot processes, but each with a new name, called "vaderbot0.exe", "vaderbot1.exe", "vaderbot2.exe", and so on up to "vaderbot9". How can you kill all of these processes based on their process name in one command?

Again, while there were many approaches, two answers that I liked were:

```
C:\> wmic process where (name like "vaderbot%") delete
```

```
C:\> taskkill /f /im vaderbot*
```

I rather like that wildcard * capability in taskkill and the "like" and % options in wmic. Nice stuff! But, note that wmic does not support *, nor does taskkill support %. So, you have to use the right wild card with the proper command!

3) Unfortunately, as the last Vader Bot numbered process ("vaderbot9") is about to be killed, it generates a whole bunch of new Vader Bot processes, with apparently random names, such as QnV5I.exe, ENvdW.exe, 50ZXI.exe, gSGFj.exe, ayBSZ.exe, WxvYW.exe, RIZCw.exe, gUGxl.exe, YXNII.exe, and finally, Q==.exe. How can you kill all of these processes in one command without knowing their Process IDs?

It pains me to say it, but no one managed to decipher those apparently random names. Note that whenever I say something like "apparently random" you can be sure that they are not random at all! They were some form of encoded information. But what encoding scheme was used? Check out that last process name, "Q==.exe". The == often implies that something is Base-64 encoded. But encoding of what, and how do you unencode it? Well, if you use a web-based Base-64 encoder/decoder, such as the one at <http://makcoder.sourceforge.net/demo/base64.php>, or a local tool that does such encoding/decoding (like Paros Proxy with its Encoding/Hash tool), you'll see that these process names really decode to:

Buy Counter Hack Reloaded, Please!

You are probably thinking, "Gee, Ed, could you put that in a larger font?"

Why, yes, I can…

Buy Counter Hack Reloaded, Please!

But, no one got that in any of the entries… Sometimes, advertising can be a little too subtle when buried in a challenge base-64 encoded. As you can see, I’ve learned my lesson and won’t make such ads as subtle going forward. ;)

Anyway, there were several solutions to this one, but one of my favorites is this, based on something that the vaderbots still all have in common, their user name:

```
C:\> taskkill /f /fi &ldquo;username eq vader&rdquo;
```

I’m pretty bummed that wmic cannot be used to refer to a process based on user name. Whenever I’m using wmic, I typically start by running the following command to see all attributes of a given area I can interact with:

```
C:\> wmic [area] list full
```

So, for processes, if you run the following, you’ll see all attributes of a process you can interact with using wmic:

```
C:\> wmic process list full
```

If you look through that output, you’ll see that wmic process knows gobs of details about every running process on the box, but it just does not know about username or SID. What a major bummer! Oh well.

Several of you got around this bummer by utilizing wmic with either the CreationDate or ParentProcessID items. That’s not a bad approach, but it is more complex than relying purely on taskkill, as above.

4) And yes again unfortunately, as the last apparently random-named bot process is about to be killed, it generates one more process for Vader Bot, named smss.exe. How can you kill this final Vader Bot process in a single command without knowing its Process ID?

And now, here’s where things got tricky. About three quarters of our respondents tried to use taskkill to end smss.exe, thinking that question number 4 was pretty much the same as question number 3. Sadly, that is not the case. In these challenges, if it seems to be trivially easy, it might just be hard. The taskkill command, like the Windows Task Manager, cannot kill a process named smss.exe, not matter how you refer to it (by user name, process name, etc.)

Quite a few of you tried doing this, which caused major problems:

```
C:\> taskkill /f /t /im smss.exe
```

That had the effect of killing the entire process tree (/t) of smss.exe, with the exception of smss.exe itself! Now, smss.exe is rather like the init daemon of Windows, so this command killed almost all processes on the box. It would not kill the legit smss.exe, nor the evil one, because taskkill cannot do that. But, it would have killed everything else, including such vital processes as lsass.exe and the DCOM Server Process Launcher. When a Windows box notices that its own lsass.exe or DCOM Server Process Launcher is gone, it forcibly reboots within 60 seconds. So, these answers were problematic, because they would leave only the good and evil smss.exe processes standing, while forcing the box to crash.

So, with taskkill out of action, how could you have done it? There were several approaches. You might think about just using wmic, as in:

```
C:\> wmic process where name="smss.exe" delete
```

That's a nice approach, and qualified Ni9e, Morgan Bailey, Scott G, Matt, Moses Hernandez, and Michael McAndrews for an honorable mention. However, this approach, while simple and functional on Windows 2003, did have a problematic side-effect: it kills the real smss.exe, a vital system process. Now, a system can still run without an smss.exe, but I have seen such machines become unstable. That's why I preferred solutions that kept the good smss.exe running, while killing only the evil smss.exe.

Also, in my own work, I've noticed an unusual behavior in this command. I've tried it on about 20 fully patched Windows XP boxes and 10 fully patched Windows 2003 boxes. It always works on Windows 2003, but on about half of the XP boxes, wmic could not kill a process named smss.exe for me. Go figure! I gave credit for this answer, because in my work, it has always functioned on Windows 2003, which was our target environment.

However, using the Hippocratic oath, approaches that left the legit smss.exe running are preferred to the above. Here they are…

One of the simplest was to run wmic in interactive mode, and to say "yes" to killing the evil smss.exe and "no" to killing the good one, based on the legit process having a lower PID. That's simple and works. Cor1nne did it that way, and kudos to her.

```
C:\> wmic process where name='smss.exe' delete /interactive:on
```

Another approach was based on the fact that the legit smss.exe always has a ParentProcessID of 4, given that it is started always at the beginning of the boot process by the "System"; faux process, with a PID of 4. So, solutions of the following were quite good:

```
C:\> wmic process where "name=\"smss.exe\" and parentprocessid!=4" delete
```

Justin Snyder, Travis, and Timothy Maletic used this approach. I especially like the way you had to use the \ escape sequence before the quote, the WQL "and" operator, and the != operator to pull this off. Nice work!

Another approach that I like, because it illustrates an interesting point, relies on the commandline invocation of the vaderbot, which always runs with a -d flag, as I outlined in the story. A really nifty capability of wmic involves pulling out the commandline invocation of each running program. So, you could have killed the evil smss.exe by running:

```
C:\> wmic process where commandline="smss.exe -d" delete
```

A variation of this was entered by Steve Hunter. That commandline viewing option within wmic is very very helpful to incident handlers investigating how various processes were started on a box.

Another approach was narrowing down the evil smss.exe by using the path to the real one, a tactic used by Marcelo Borges and others.

Others, including Alex Di Giuseppe, relied on the ParentProcessID being the value that was printed out from the previous command (the answer to question 3). That's a fine approach, but not as elegant as the others. The question mentions not knowing its process ID, and didn't rule out knowing its parent process ID, so I gave you credit if you used this approach.

A far more complex way of doing this was to base it on the CreationDate and/or memory usage of the process, something that wmic does let you interact with. This approach is fine, but is more complex than other approaches. Raul Siles and Jason Adamson used this approach.

Another method that was complex but worked was to use a small single-line or even multiple line script, wrapping wmic up so you could kill it based on process ID determined by tasklist in the same command. Variants of this solution were entered by Stephane Grobety, Dean De Beer and Tyler Hudak.

Another script-based approach that did not use wmic was to rely on the Windows 2003 built-in debugger, ntsd. A couple of people used that approach with somewhat complex scripts, including Mary Arras, whose scripting kung fu is a site to behold!

Another radically weird yet very clever approach was offered by Bob Dehnhardt. Instead of killing the evil smss.exe, he took the hyperdrive.exe program, and gave the vaderbot a taste of its own medicine. He renamed hyperdrive.exe with a new name… yup… you guessed it: smss.exe! Then, the vaderbot might not be able to kill it any more. Of course, if Lord Vader had programmed it with some of the commands above, he could still kill the newly named hyperdrive.exe process, but I thought this was a very interesting way of turning the tables on the bad guys.

But, there was an easier and more elegant way than using massive wmic kung fu, scripting, or playing tricks with the vaderbot. This approach, used by Jason, Dan Roberts, pcsneaker, and Erik C, relied on the tskill command. While taskkill cannot kill a process named smss.exe, tskill can, and it is built into Windows 2003 and Windows XP Pro. Interestingly, the following command will kill the evil smss.exe, but leave the legit smss.exe running. When tskill tries to kill the legit smss.exe, it gets an “Access is denied” message, and it only hits the evil one:

```
C:\> tskill smss /a /v
```

This rather obscure Windows command was the simplest way of dealing with this problem. The other approaches were totally valid, and I gave credit for all of them. Please note that the winners are based on a cumulative score for all of the questions, including this last one:

5) Finally, instead of spawning separate processes, the Vader Bot could have used other techniques to survive on the machine, continuing to run in light of R2D2’s process-killing assault. Please describe techniques for malware (or even non-malicious code) to continue running without having to spawn new processes.

For this one, I was looking for a description of various hiding and process shielding techniques, including rootkits, DLL injection, and so on. I won’t expound upon this one in detail, but instead refer you to the winners to read their crafty evil. Please do note that I always include an open-ended question with these challenges, to help differentiate winners from nearly identical correct answers. These open-ended ones are the items that separate the true Jedi from the young padawans.

So, enough pontificating! Let’s move on to the winners. As you recall, there were three winners, one for technical accuracy, one for creativity (and technical completeness), and one chosen at random.

Our Technical Winner is:

Erik Schroeder. His technical answer, which relied on tskill for number 4, was impeccable. Also, his descriptions of the

additional ways vaderbot could have been more evil were really solid and earned him the win.

Click [HERE](#) for the Technical Winning Entry.

Technical honorable mentions for their great work go to:

Dan Roberts

Matt

Jason Morrow

Scott G

Moses Hernandez

Cor1nne

Marcelo Borges

Jason Adamson

MBailey

Ni9e

Mary Arras (the force is strong with this one!)

Justin Snyder

PCSneaker

Bill Richards

Steven Hunter

Stephane Grobety

Joshua Hudson

Dean De Beer

Alex Di Giuseppe

David Barroso

Travis

Steven Hunter

Mike Cardosa

Our Creative Winner is…

Timothy Maletic. His answer is funny, technically brilliant, and heavily laced with the movie’s plot. I like how he

had R2 “fat finger” a command, causing the command line to go away, forcing him not to use the parent process id from the previous command. Nice way of integrating plot and technical sophistication. However, I have no idea how a droid like R2 fat fingers anything. And, here’s a money quote from Obi-Wan in Timothy’s write-up: “Call me a defunct process. I'm here to tell you that the Falcon's Operating System has just spawned a new smss.exe process. I don't understand why this OS is trusted to run mission-critical systems. You'll never find a more wretched hive of scum and villainy...”

Click [HERE](#) for the Creative Winning Entry.

For creativity, honorable mentions go to:

Raul Siles

Michael McAndrews

Bob Dehnhardt

Ian Mitchell

Tyler Hudak

gNick

And, our randomly chosen winner is:

Stephen Cottrell, chosen by the pseudo-random number generator on my Casio calculator!

Sorry, but no honorable mentions among the random folks!

Congrats to all of our winners, who will get an autographed copy of my book Counter Hack Reloaded.

That’s all for this time, folks. Around July 1, we’ll have our next challenge, being crafted as I type by the honorable Mike Poor. He’s whipping up a challenge based on the Quentin Tarantino film, Kill Bill. You can only guess what that plot will involve.

Thank you again for your interest and hard work!

--Ed Skoudis.

Intelguardians

*By the way, did you know that MS Word's dictionary flags "jedi" with a small j as a spelling error, but "Jedi" with a cap J is ok? Note that Sith, regardless of capitalization, is always flagged as a spelling error. Perhaps Bill himself is the ultimate Jedi master, or possibly the ultimate Sith Lord looking for plausible deniability.