

[TOOL] ComLog.pl, a WIN32 Command Prompt Logger

Source: <http://www.derkeiler.com/Mailing-Lists/Securiteam/2002-08/0088.html>

From: support@securiteam.com

Date: 08/21/02

From: support@securiteam.com

To: list@securiteam.com

Date: Wed, 21 Aug 2002 21:21:49 +0200 (CEST)

The following security advisory is sent to the securiteam mailing list, and can be found at the SecuriTeam web site: <http://www.securiteam.com>

-- promotion

When was the last time you checked your server's security?

How about a monthly report?

<http://www.AutomatedScanning.com> -- Know that you're safe.

ComLog.pl, a WIN32 Command Prompt Logger

DETAILS

The goal of this paper is to present a new Perl tool that Floydman made to monitor DOS sessions on Windows NT/2K. This tool can be used by administrator to keep a history of commands typed in the DOS command prompt and the associated output, for example on an IIS server. This can help administrators to figure out what the attacker has done after compromising the machine via one of the numerous vulnerabilities this software present. Of course, like any key logger out there, it can also be use for malicious purpose, but Floydman thinks other programs out there make better spying tools.

Tool source:

```
#!D:\dev\perl\bin\perl.exe
```

```
# ComLog 1.0
```

```
#####
```

```
# ComLog 1.0 #
```

```
# by Floydman floydian\_99@yahoo.com #
```

```
# Copyright 2002 SecurIT Informatique Inc. http://securit.iquebec.com #
```

```
##
```

```
# This program captures the input/output of the Windows NT Command Promt
```

Securiteam: [TOOL] ComLog.pl, a WIN32 Command Prompt Logger

```
(cmd.exe). #
# It does so by pretending to be the real prompt and forwarding the
commands to the #
# real (and renamed) cmd.exe. I/O is stored in a random-generated log file

in #
# \WINNT\Help\Tutor.
#####

#####
# LICENSE #
# This software is Open Source. This means that its source code is open,
free and avai-#
# lable for anyone to look into, make modifications, correct bugs (let me
know, please) #
# and use for personal and commercial use. You can create your own
binaries with #
# perl2exe (www.indigostar.com), or download the install package from #
# securit.iquebec.com/download.html. #
#####

#####
# Main Program #
# This is the main structure of ComLog. #
# This programs is a loop that will be passed only once if arguments are
passed with #
# ComLog, since it means it was not launched for interactive use. Else,
the loop will #
# go on until the user types 'exit' or breaks otherwise the program
execution (CTRL-C).#
# The program then presents the prompt to the user, and gets the command
typed at the #
# keyboard. Change of drives or directories are checked for, since ComLog
have to #
# provide these parameters to the real command prompt. Everything is
logged in a #
# random generated filename to allow for multiple instances. Output is
sanitized to #
# hide ComLog's presence. #
#####

use Win32;

# Initialization of entrant parameters, current dir, command, exit and
currentdrive
$input = join (' ',@ARGV);
$index = 0;
$currentdir[$index] = '.\';
$command = "";
$exit = 1;
$currentdrive[$index] = getcurrentdrive ($currentdir[$index]);
```

Securiteam: [TOOL] ComLog.pl, a WIN32 Command Prompt Logger

```
$nothing = 1; # nop
$winnt = "Windows NT Version 4.0";
$wintwok = "Microsoft Windows 2000";
$cleanlog = 0; # Set to 0 to keep the local log files, set to 1 to delete
them after the session is done

$history = randomfilename();

open (WINVER, "cm_.exe /C ver |") or die "Can't open pipe";
@header = <WINVER>;
close (WINVER);

$header = join(" ", @header);

if ($header =~ m/$winnt/) { sendoutput ("Microsoft(R) Windows NT(TM)\n(C)
Copyright 1985-1996 Microsoft Corp.\n");}
if ($header =~ m/$wintwok/) { sendoutput ("Microsoft Windows 2000 [Version
5.00.2195]\n(C) Copyright 1985-1999 Microsoft Corp.\n");}

# Get the path for the fake prompt
getprompt($currentdrive[$index], $currentdir[$index]);

# If no args are passed, we cancel the $exit marker and launch the prompt
if ($input eq " ") {$input = <STDIN>; $exit = 0;}

# Enters in an "interactive" session and will terminate on 'exit', the
loop
will run only once if $exit is set to 1
while ($input ne "exit\n")
{
chomp $input;
$cdcommand = 0; $baddir = 0; $baddrive = 0;

SWITCH: {
# If the command starts with 'cd', then it checks if the next character is
\
# If it is, then the string replaces the $currentdir. If it does not
start with
# a \, then the string is appended at the end of the path in $currentdir
# If nothing follows the cd command, then the command is put back in
$input
# since we modified the string for analysis with the shift command above
$input =~ m/^cd/i && do {
@input = split (//,$input);
if ($input[0] eq "c" or $input[0] eq "C") {shift @input;}
if ($input[0] eq "d" or $input[0] eq "D") {shift @input;}
while ($input[0] eq " ") {shift @input;}

if (@input)
{
if ($input[0] eq "\\")
```

Securiteam: [TOOL] ComLog.pl, a WIN32 Command Prompt Logger

```
{
  $input = join (",@input);
  $direxist = checkdir($currentdrive[$index], "\\", $input);
  if ($direxist)
  {
    $currentdir[$index] = $input;
    $cdcommand = 1;
    $input = "cd ".$input;
  }
  else { $cdcommand = 1; $baddir = 1; $input="cd"; }
}
else
{
  $input = join (",@input);
  $direxist = checkdir($currentdrive[$index],$currentdir[$index],
$input);
  if ($direxist)
  {
    $currentdir[$index] = $currentdir[$index].$input."\\";
    $cdcommand = 1;
    $input = "cd ".$input;
  }
  else
  { $cdcommand = 1; $baddir = 1; $input="cd"; }
}
}
else
{ $input="cd"; }
last SWITCH;
};

# If the command is 'cls', we nullify the command and simulate it from the
perl program
  $input=~m/cls/i && do {
    system("cls");
  sendinput ($input."\n");
  $input = "";
  last SWITCH;
};

# If the command is to change the drive letter (c:, d:, etc), we set our
variables accordingly
# If it's the first time the drive is accessed, an entry is added in the
table to store it with its current directory
$input=~m/^\w:$/ && do {
  $i = 0;
  foreach $drive (@currentdrive)
  {
    if ($currentdrive[$i]=~m/$input/i)
    { $index = $i; last SWITCH; }
  }
  $i++;
}
```

Securiteam: [TOOL] ComLog.pl, a WIN32 Command Prompt Logger

```
    if ($index ne $i)
    {
$driveexist = checkdrive($input);
if ($driveexist)
    {
    $index = $i;
    $currentdrive[$index] = $input;
    $currentdir[$index] = '\\';
    }
    else {$baddrive=1; $input = $currentdrive[$index];}
}

    last SWITCH;
        };
    $nothing = 1;
}

# If it is a 'cd' command, then we simply execute a 'cd' command with the
# $currentdir path
# If not, then we issue the 'cd' command anyway, as it is needed to
# position the summoned instance
# of cm_.exe in the right directory, then we call the command typed at the

prompt
# $command contains the string of the final command to be sent to the real

DOS prompt
if ($command)
    { $command = "cm_.exe /C ".$currentdrive[$index]." && cd
    ".$currentdir[$index]; }
else
    { $command = "cm_.exe /C ".$currentdrive[$index]." && cd
    ".$currentdir[$index]." && ".$input; }

# Writes the input to the history file
sendinput ($input."\n");

# If the command is valid and it's not an empty carriage return, then we
# call the command and capture the output

if ($baddir) { sendoutput("The system cannot find the path specified.\n");
}
if ($baddrive) { sendoutput("The system cannot find the drive
specified.\n"); }

if ($input ne "")
    {
    open (COMMANDOUTPUT, $command." |") or die "Can't open pipe";
    @output = <COMMANDOUTPUT>;
    close (COMMANDOUTPUT);
    sendoutput (@output);
    }
```

Securiteam: [TOOL] ComLog.pl, a WIN32 Command Prompt Logger

```
}

# If the $exit marker is not set, get the fake prompt
if ($exit) {$input = "exit\n";} else {getprompt($currentdrive[$index],
$currentdir[$index]); $input = <STDIN>;}

} # End of While

# sends the final command (exit) to the history log file
sendinput ($input."\n");

if ($cleanlog) {unlink ($history);}

#End Of Main

#####
# procedure getcurrentdrive(currentdir) #
# This procedure gets the drive where is located the current directory. #
#####
sub getcurrentdrive
{ my ($dir) = @_ ;

open (PROMPT, "cd".$dir." && cd |") or die "Can't open pipe";
$prompt = <PROMPT>;
chomp $prompt;
close (PROMPT);

@prompt = split (//,$prompt);
$drive = join ("", ($prompt[0], $prompt[1]));
return ($drive);
}

#####
# procedure getprompt(currentdrive, currentdir) #
# This procedure receives the current drive and directory and fakes a
command prompt. #
#####
sub getprompt
{ my ($drive, $dir) = @_ ;

open (PROMPT, $drive."&& cd ".$dir." && cd |") or die "Can't open pipe";
$prompt = <PROMPT>;
chomp $prompt;
close (PROMPT);
sendoutput ("\n".$prompt.">");
}

#####
# procedure sendinput(line) #
# This procedure writes the input received into the history file #
#####
```

Securiteam: [TOOL] ComLog.pl, a WIN32 Command Prompt Logger

```
sub sendinput
{ my (@line) = @_;
```

\$now = localtime;

Opening of history file
open (HISTORY, ">>".\$history) || die "Can't open session log file";
lock HISTORY;
print HISTORY "\$now\n";
print HISTORY "@line";
close (HISTORY);
}

procedure sendoutput(output) #
This procedure writes the output from commands received into the history

file and to #
the console. It also sanitize the output to conceal the program
presence. #

sub sendoutput
{ my (@output) = @_;

\$now = localtime;

Opening of history file
open (HISTORY, ">>".\$history) || die "Can't open session log file";
lock HISTORY;
print HISTORY "\$now\n";
foreach \$line (@output)
{
\$line=~s/cm_.exe/cmd.exe/i;
if (!((\$line=~m/711,342/) || (\$line=~m/\w{8}.clg/) || (\$line=~m/cm_/)))
{ print "\$line";
print HISTORY "\$line";
}
}
close (HISTORY);
}

procedure checkdir(\$drive, \$path, \$dir) #
This procedure checks for the existence of a directory before changing
to
it. #

sub checkdir
{ my (\$drive, \$dir, \$unknown) = @_;

Securiteam: [TOOL] ComLog.pl, a WIN32 Command Prompt Logger

```
$command = "cm_.exe /C ".$drive." && cd ".$dir." && if exist ".$unknown."  
echo OK";
```

```
    open (COMMANDOUTPUT, $command." |") or die "Can't open pipe";  
    $output = <COMMANDOUTPUT>;  
    close (COMMANDOUTPUT);  
    if ($output eq "OK\n") {return 1;} else {return 0;}  
}
```

```
#####
```

```
# procedure checkdrive($drive) #  
# This procedure checks for the existence of a drive before changing to  
it. #
```

```
#####
```

```
sub checkdrive  
{ my ($drive) = @_;  
  
    opendir (TESTDRIVE, $drive."\\") or return 0;  
    closedir (TESTDRIVE);  
    return 1;  
}
```

```
#####
```

```
# procedure randomfilename() #  
# This procedure generates a random filename for the session log. #
```

```
#####
```

```
sub randomfilename  
{  
    for ($gen=0; $gen<8; $gen++) {$random = int (rand 26)+97; $history =  
$history.chr($random);}  
    $history = $history.".clg";  
    $history = $ENV{SystemRoot}."/Help/tutor/".$history;  
    return $history;  
}  
#EOF
```

ADDITIONAL INFORMATION

Additional details on the tool can be found at:

<http://www.geocities.com/floydian_99>

http://www.geocities.com/floydian_99 or at <<http://securit.iquebec.com/>>

<http://securit.iquebec.com/>

The tool has been provided by <mailto:floydian_99@yahoo.com> Floydman.

=====

This bulletin is sent to members of the SecuriTeam mailing list.

To unsubscribe from the list, send mail with an empty subject line and body to:

list-unsubscribe@securiteam.com

In order to subscribe to the mailing list, simply forward this email to: list-subscribe@securiteam.com

=====
=====

DISCLAIMER:

The information in this bulletin is provided "AS IS" without warranty of any kind.

In no event shall we be liable for any damages whatsoever including direct, indirect, incidental, consequential, loss of business profits or special damages.

-
- ***Previous message:*** support@securiteam.com: "[\[UNIX\] Aquonics File Manager Directory Traversal Vulnerability And Privilege Escalation](#)"
 - ***Messages sorted by:*** [\[date \]](#) [\[thread \]](#) [\[subject \]](#) [\[author \]](#) [\[attachment \]](#)