Monday, January 14, 2013

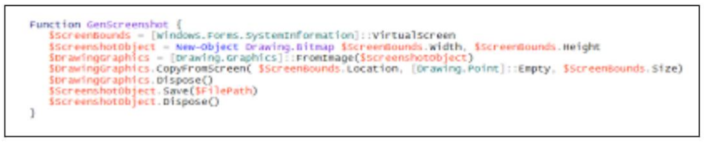
Automating Screenshots with PowerShell

Penetration tests can become very hectic at a moment's notice. One second you are casually reviewing HTML source for a target website and the next dropping a webshell and hooking browsers before staying up all night trying to gain persistent domain-admin access to the enterprise. Keeping notes during hectic times can be difficult, tedious and potentially distracting. Sometimes, it pays to have something taking notes for you. I like to utilize both a key-logger that does time stamping and take frequent screenshots.

There are applications that can take screenshots for you at regular intervals and in the past I used an AutoIt macro to printscreen and save. That works well when I am on my own machine, but what if I was at a kiosk or doing an insider assessment from one of their workstations? I needed a PowerShell script that could take a screenshot at regular intervals, time stamp it, save it to a file and not tamper with the contents of the clipboard.

While looking for a good script to start from, I found this one that uses inline C# which seemed a little over-the-top. Another one seemed simple and straight-forward so I started working with it. After getting the function built, I was quickly annoyed with data from the clipboard disappearing. I knew I had to find another way. After digging through MSDN for an hour, I found the Bitmap Class and the System Info Class.

After loading the System.Windows.Forms assembly, I created a function that will be called to take the screenshot and save it to the disk:



```
Function GenScreenshot {
    $ScreenBounds = [Windows.Forms.SystemInformation]::VirtualScreen
    $ScreenshotObject = New-Object Drawing.Bitmap $ScreenBounds.Width, $ScreenBounds.Height
    $DrawingGraphics = [Drawing.Graphics]::FromImage($ScreenshotObject)
    $DrawingGraphics.CopyFromScreen( $ScreenBounds.Location, [Drawing.Point]::Empty, $ScreenBounds.Size)
    $DrawingGraphics.Dispose()
    $ScreenshotObject.Save($FilePath)
    $ScreenshotObject.Dispose()
}
```

Next we need a way to distinguish each file and a way to stamp them with the time it was taken:

Twitter

obscuresec

Other Content

Recommended Books

Recommended Links

Standard Disclaimer

Presentation Slides

RTFM

amazon

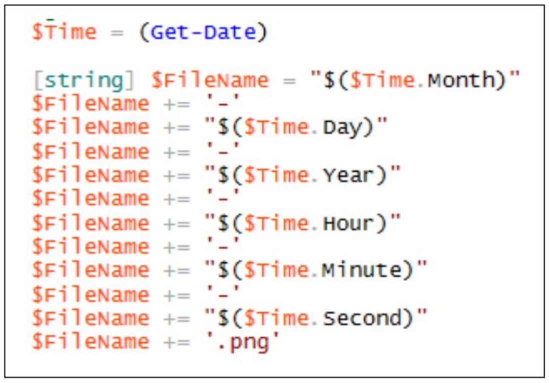
Rtfm: Red Team Field Manual

$9.00 Prime

Shop now

Blog Archive

- ► 2014 (6)
- ▼ 2013 (12)
  - ► September (1)
  - ► July (2)
  - ► June (1)
  - ► April (1)
  - ► March (2)
  - ► February (2)
  - ▼ January (3)
    - Automating Screenshots with PowerShell
    - Test Antivirus with


```
$Time = (Get-Date)

[string] $FileName = "$($Time.Month)"
$FileName += '-'
$FileName += "$($Time.Day)"
$FileName += '-'
$FileName += "$($Time.Year)"
$FileName += '-'
$FileName += "$($Time.Hour)"
$FileName += '-'
$FileName += "$($Time.Minute)"
$FileName += '-'
$FileName += "$($Time.Second)"
$FileName += '.png'
```

Now we just need to settle on parameters, add this to a do-while loop and wrap the whole thing in a try-catch block. The result is Get-TimedScreenshot:

Get-TimedScreenshot

```
Function Get-TimedScreenshot {
<#
.SYNOPSIS

    Get-TimedScreenshot

    Author: Chris Campbell (@obscuresec)
    License: BSD 3-Clause

.DESCRIPTION

    A function that takes screenshots and saves them to a fo

.PARAMETER $Path

    Specifies the folder path.

.PARAMETER $Interval

    Specifies the interval in seconds between taking screens

.PARAMETER $EndTime

    Specifies when the script should stop running in the for

.EXAMPLE

    PS C:\> Get-TimedScreenshot -Path c:\temp\ -Interval 30

.LINK

    http://obscuresecurity.blogspot.com/2013/01/Get-TimedScr
    https://github.com/obscuresec/random/blob/master/Get-Tim

#>

    [CmdletBinding()] Param(
            [Parameter(Mandatory=$True)]
            [ValidateScript({Test-Path -Path $_ })]
            [string] $Path,

            [Parameter(Mandatory=$True)]
            [int32] $Interval,

            [Parameter(Mandatory=$True)]
            [string] $EndTime
            )

        #Define helper function that generates and saves scr
        Function GenScreenshot {
```

Instead of downloading or installing additional software, we now have a script that will

EICAR and PowerShell

Resolve Shortened Links in Bulk with PowerShell

► 2012 (15)

► 2011 (7)

Learn PowerShell

amazon

Learn Windows PowerShell in...

$39.79 Prime

Shop now

About Me

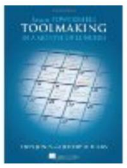
Chris

Just another obscure security professional.

View my complete profile

PowerShell Toolmaking

amazon



Learn PowerShell...

$35.22 Prime

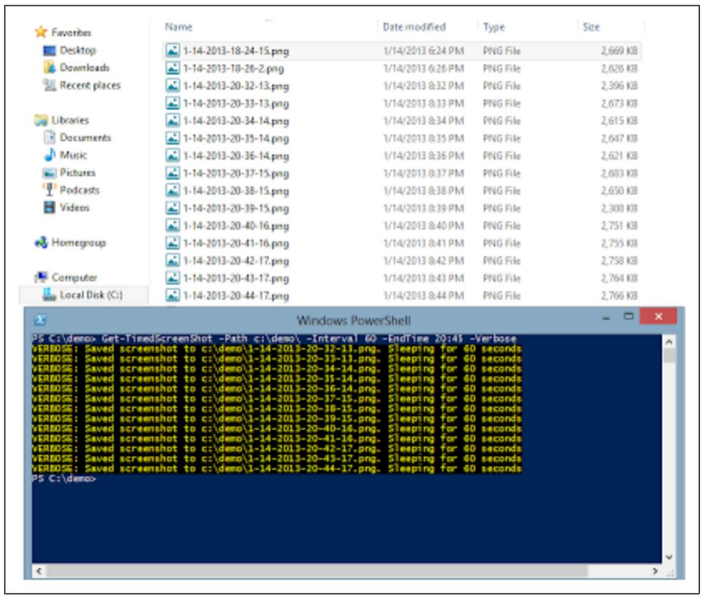
Shop now

Blogs I Read

Carnal0wnage & AR

DevOoops: Client Provisioning (Kickstart take periodic screenshots. The images can be large so I wouldn't recommend leaving it running overnight, but its great to help you fill in gaps in note-taking at the end of a long hacking session.

***Updated 8/6/2013: The maintained version of this script can by found within the PowerSploit framework here.



Name	Date modified	Type	Size
1-14-2013-18-24-15.png	1/14/2013 6:24 PM	PNG File	2,669 KB
1-14-2013-18-26-2.png	1/14/2013 6:26 PM	PNG File	2,626 KB
1-14-2013-20-32-13.png	1/14/2013 8:32 PM	PNG File	2,396 KB
1-14-2013-20-33-13.png	1/14/2013 8:33 PM	PNG File	2,673 KB
1-14-2013-20-34-14.png	1/14/2013 8:34 PM	PNG File	2,615 KB
1-14-2013-20-35-14.png	1/14/2013 8:35 PM	PNG File	2,647 KB
1-14-2013-20-36-14.png	1/14/2013 8:36 PM	PNG File	2,621 KB
1-14-2013-20-37-15.png	1/14/2013 8:37 PM	PNG File	2,603 KB
1-14-2013-20-38-15.png	1/14/2013 8:38 PM	PNG File	2,650 KB
1-14-2013-20-39-15.png	1/14/2013 8:39 PM	PNG File	2,308 KB
1-14-2013-20-40-16.png	1/14/2013 8:40 PM	PNG File	2,751 KB
1-14-2013-20-41-16.png	1/14/2013 8:41 PM	PNG File	2,755 KB
1-14-2013-20-42-17.png	1/14/2013 8:42 PM	PNG File	2,758 KB
1-14-2013-20-43-17.png	1/14/2013 8:43 PM	PNG File	2,764 KB
1-14-2013-20-44-17.png	1/14/2013 8:44 PM	PNG File	2,766 KB

```
PS C:\demo> Get-TimedScreenShot -Path c:\demo\ -Interval 60 -EndTime 20:45 -Verbose
VERBOSE: Saved screenshot to c:\demo\1-14-2013-20-32-13.png. Sleeping for 60 seconds
VERBOSE: Saved screenshot to c:\demo\1-14-2013-20-33-13.png. Sleeping for 60 seconds
VERBOSE: Saved screenshot to c:\demo\1-14-2013-20-34-14.png. Sleeping for 60 seconds
VERBOSE: Saved screenshot to c:\demo\1-14-2013-20-35-14.png. Sleeping for 60 seconds
VERBOSE: Saved screenshot to c:\demo\1-14-2013-20-36-14.png. Sleeping for 60 seconds
VERBOSE: Saved screenshot to c:\demo\1-14-2013-20-37-15.png. Sleeping for 60 seconds
VERBOSE: Saved screenshot to c:\demo\1-14-2013-20-38-15.png. Sleeping for 60 seconds
VERBOSE: Saved screenshot to c:\demo\1-14-2013-20-39-15.png. Sleeping for 60 seconds
VERBOSE: Saved screenshot to c:\demo\1-14-2013-20-40-16.png. Sleeping for 60 seconds
VERBOSE: Saved screenshot to c:\demo\1-14-2013-20-41-16.png. Sleeping for 60 seconds
VERBOSE: Saved screenshot to c:\demo\1-14-2013-20-42-17.png. Sleeping for 60 seconds
VERBOSE: Saved screenshot to c:\demo\1-14-2013-20-43-17.png. Sleeping for 60 seconds
VERBOSE: Saved screenshot to c:\demo\1-14-2013-20-44-17.png. Sleeping for 60 seconds
PS C:\demo>
```

There is also a clear post-exploitation use for the function. You can schedule it to run and maybe add a check to see if the screensaver is running to make sure you aren't wasting space. I think the function is pretty flexible and with event triggering and an email function could potentially be used as a simple parental alert system. As is, it works for my purposes which is to remind me what I did today. I hope you find it useful and thanks for reading. In case you were wondering, it works well with multiple monitor setups:

Please let me know if you have any issues, bugs or questions. Hopefully, I will see you at Shmoocon and Firetalks. Also, if you are in town, check out Shmoocon Epilogue. The other talks look really good, but I get the chance to present "No Tools? No Problem! Building a PowerShell Bot." It will cover chaining simple tasks like this one into a nefarious PowerShell script.

-Chris

Posted by Chris at 9:27 PM

G+1 +9 Recommend this on Google

Labels: .Net, Get-Date, Get-TimedScreenshot, Join-Path, Parameter Validation,

Files)
19 hours ago

Metasploit
Breaking Metasploitable3: The King of Clubs
4 days ago

Dark Operator
Being Thankful for 2016
5 days ago

Exploit Monday
PowerShell is Not Special - An Offensive PowerShell Retrospective
1 week ago

Mubix's Room362
2017 Shmoocon Hiring List
1 week ago

Still Passing the Hash 15 Years Later
I'm PKDC, your Personal Kerberos Domain Concierge for the Whatever_domain
7 months ago

Command Line Kung Fu
Episode #180: Open for the Holidays!
2 years ago

Pentest Geek
Phishing Frenzy: SSL Support on Rails 4 with Syntax Highlighting
2 years ago

clymb3r
Cracking Open PowerShell's Constrained Runspace
2 years ago

Blackthorne R&D
Quick and Dirty Windows Malware Analysis
2 years ago

Colligo Mentis

Violent Python

Main Site

PowerShell, PowerSploit, Screenshots, Start-Sleep, System.Windows.Forms

11 comments:

Anonymous January 15, 2013 at 5:22 AM

this is just cool

Reply

Anonymous January 15, 2013 at 10:42 AM

Issue running this with provided example. . .\script -Path C:\temp -Interval 30 -EndTime 10:54

Reply

Replies

Chris January 15, 2013 at 2:14 PM

There are two things you should probably check. The first is that you have a "C:\temp" directory since that is not a default directory and that you have the ability to write to it under the context that PowerShell is currently running. You should be provided a warning if either of those conditions exist.

The next and more likely thing to check is the time. The time is in 24-hour time so if 10:54 AM has passed for the day, it will only run once and then quit. Try 22:54 if you would like it to be PM. Feel free to edit the time to allow for multi-day runs, but I prefer it to not be able to fill my hd by continuing to save screenshot images.

Thanks for you commend and let me know if that fixes your issue.

Anonymous June 6, 2013 at 4:06 AM

Script crash then capture UAC message

Reply

JamesT January 16, 2013 at 4:34 PM

Paste the content of the entire file in PS. Hit rreturn twice to return to prompt. Run command again:
Get-TimedScreenshot -Path C:\temp -Interval 30 -EndTime 10:54
If command executes (which it should) you have a permission issue. Check permissions and execution policy.

Reply

Replies

Chris January 16, 2013 at 6:08 PM

It certainly could be a permissions issue on c:\temp, but I don't think the ExecutionPolicy comes into play since the file is just a function and call to that function. The ExecutioPolicy restrictions are only enforced on scripts.

Reply

Anonymous May 21, 2013 at 2:27 AM

Thanks Chris for the great post.
Here is an alternative for the filename.
$FileName = [DateTime]::Now.ToString("MM-dd-yy-hh-mm-ss")
MM=Month and mm=minutes

Jose

Reply

Replies

Chris August 6, 2013 at 9:47 AM

Thanks!

Reply

Anonymous August 6, 2013 at 6:22 AM

Github link is not working.

Reply

Chris August 6, 2013 at 9:47 AM

Thank you. Fixed.

Reply

John New January 29, 2014 at 5:44 PM

Nicely done. Thanks.

Reply

Enter your comment...

Comment as: Unknown (Google) **Sign out**

Publish **Preview** Notify me

Newer Post Home Older Post

Subscribe to: Post Comments (Atom)