Code Secure

Browsing the Web and Reading E-mail Safely as an Administrator

Michael Howard
Microsoft Security Engineering

November 15, 2004

Summary: Michael Howard discusses how you can run as an administrator and access Internet data safely by dropping unnecessary administrative privileges when using any tool to access the Internet. (10 printed pages)

Download the DropMyRights.msi file [http://download.microsoft.com/download/f/2/e/f2e49491-efde-4bca-9057-adc89c476ed4/dropmyrights.msi] .

I've said this many times, but I'll say it again, "Running with an administrative account is dangerous to the health of your computer and your data." So, whenever someone says they must operate their computers as administrators, I always try to persuade them it's not the correct thing to do from a security perspective. That said, every once in a while I meet someone who has a valid reason. For example, I use one of the computers in my office to install the latest daily build of Windows, and I need to be an administrator to install the OS. However, and this is a big point, I do not read e-mail, browse the Web, or access the Internet in any form when running as an administrator on that machine. And I do not do so because the Web is the source of most of the nasty attacks today.

What if someone does want to browse the Web? Or read e-mail? Or do Instant Messaging and so on, and for some reason must run in an administrative context? If you look at the major threats to computers, they are from user interaction with the Web through tools like browsers and e-mail clients. Sure, there are non-user interaction attacks, such as Blaster (http://www.cert.org/advisories/CA-2003-20.html [http://www.cert.org/advisories/ca-2003-20.html]) and Lion (http://www.sans.org/y2k/lion.htm [http://www.sans.org/y2k/lion.htm]), but that's in part why we turned on the firewall in Windows XP SP2!

> **Note** For Best practices on running as a non-admin, I urge you to look over Aaron Margosis' blog [http://weblogs.asp.net/aaron_margosis/] to glean tips on running as a non-admin in Windows.

An Example of Why Running as an Admin Is Bad

Some nasty *malware* works only because the user browsing the Web is an administrator. A good example is a recent variation of the Bagle/Beagle worm named W32.Beagle.AV@mm. I would recommend you read up on what the worm does once it is invited onto a computer system. Symantec has a good write-up at http://securityresponse.symantec.com/avcenter/venc/data/w32.beagle.av@mm.html [http://securityresponse.symantec.com/avcenter/venc/data/w32.beagle.av@mm.html] . I say invited because the malware is not taking advantage of a coding or design defect. It is using simple human error to execute.

Amongst the many things this malware does, all of which require admin rights, are:

- Creating files in the system32 directory.
- Terminating various processes.
- Disabling the Windows Firewall.
- Downloading and writing files to the system32 directory.
- Deletes registry values in HKLM.

All these fail if the user running the e-mail client is not an administrator.

So wouldn't it be useful (read: safer) if you could browse the Web, read e-mail, and so on as a non-admin, even though you need to perform your normal daily tasks as an admin? Luckily, Windows XP and Windows Server 2003 and later support this capability using restricted tokens.

Further Detail

Windows XP and Windows Server 2003 and later support functionality called *Software Restriction Policy*, also known as SAFER [http://msdn.microsoft.com/library/default.asp.aspx?url=/library/en-us/security/security/safer.asp] , which allows a user or software developer to run code at a lower privilege without having the user enter credential information when the application starts. For example, an administrator could run an application as a normal user by stripping out certain SIDs and privileges from the application's token as the application is launched. Some applications, most notably Internet-facing applications, such as a Web browser, instant messaging, or e-mail client, should never be run under an administrative context.

The DropMyRights Application

DropMyRights is a very simple application to help users who must run as an administrator run applications in a much-safer context—that of a non-administrator. It does this by taking the current user's token, removing various privileges and SIDs from the token, and then using that token to start another process, such as Internet Explorer or Outlook. This tool works just as well with Mozilla's Firefox, Eudora, or Lotus Notes e-mail.

The code couldn't be simpler. Here's the core code:

Copy Code

```
////////////////////////////////////////////////////////////////////////////////
DWORD wmain(int argc, wchar_t **argv) {

  DWORD fStatus = ERROR_SUCCESS;

  if (2 != argc && 3 != argc) {
    Usage();
    return ERROR_INVALID_PARAMETER;
  }

  // get the SAFER level
  DWORD hSaferLevel = SAFER_LEVELID_NORMALUSER;
  if (3 == argc && argv[2]) {
    switch(argv[2][0]) {
      case 'C' :
      case 'c' :  hSaferLevel = SAFER_LEVELID_CONSTRAINED;
            break;
      case 'U' :
      case 'u' :   hSaferLevel = SAFER_LEVELID_UNTRUSTED;
            break;

      default  :   hSaferLevel = SAFER_LEVELID_NORMALUSER;
            break;
    }
  }

  // get the command line, and make sure it's not bogus
  wchar_t *wszPath = argv[1];
  size_t cchLen = 0;
  if (FAILED(StringCchLength(wszPath,MAX_PATH,&cchLen)))
    return ERROR_INVALID_PARAMETER;

   SAFER_LEVEL_HANDLE hAuthzLevel = NULL;
   if (SaferCreateLevel(SAFER_SCOPEID_USER,
                  hSaferLevel,
                  0,
         &hAuthzLevel, NULL)) {

      //  Generate the restricted token we will use.
      HANDLE hToken = NULL;
      if (SaferComputeTokenFromLevel(
         hAuthzLevel,    // SAFER Level handle
         NULL,           // NULL is current thread token.
         &hToken,        // Target token
         0,              // No flags
         NULL)) {        // Reserved

       STARTUPINFO si;
       ZeroMemory(&si, sizeof(STARTUPINFO));
       si.cb = sizeof(STARTUPINFO);
       si.lpDesktop = NULL;
```

```
          // Spin up the new process
          PROCESS_INFORMATION pi;
          if (CreateProcessAsUser(
             hToken,
             wszPath, NULL,
             NULL, NULL,
             FALSE, CREATE_NEW_CONSOLE,
             NULL, NULL,
             &si, &pi)) {

                CloseHandle(pi.hProcess);
                CloseHandle(pi.hThread);

          } else {
             fStatus = GetLastError();
             fwprintf(stderr,L"CreateProcessAsUser failed (%lu)\n",fStatus);
          }
       } else {
          fStatus = GetLastError();
       }

       SaferCloseLevel(hAuthzLevel);

    } else {
       fStatus = GetLastError();
    }

    return fStatus;
 }
```

The source code and executable are available at the top of this article. Now let's look at configuring the application to run applications in lower privilege.

Setup

Simply copy DropMyRights.exe to a folder. Then for each application you want to run in lower privilege, follow the steps in the next three sections.

Create a Shortcut

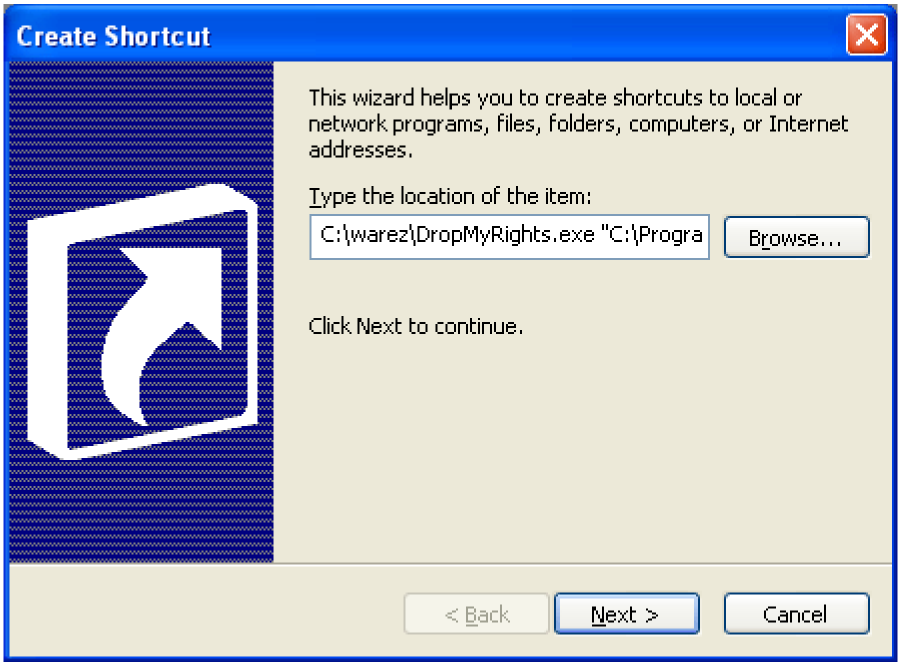
Create a shortcut and enter DropMyRights.exe as the target executable, followed by the path to the application you want to execute in lower privilege.

For example:

Copy Code

```
C:\warez\dropmyrights.exe "c:\program files\internet explorer\iexplore.exe"
```

Figure 1 shows what this will look like on your screen.



Create Shortcut

This wizard helps you to create shortcuts to local or network programs, files, folders, computers, or Internet addresses.

Type the location of the item:

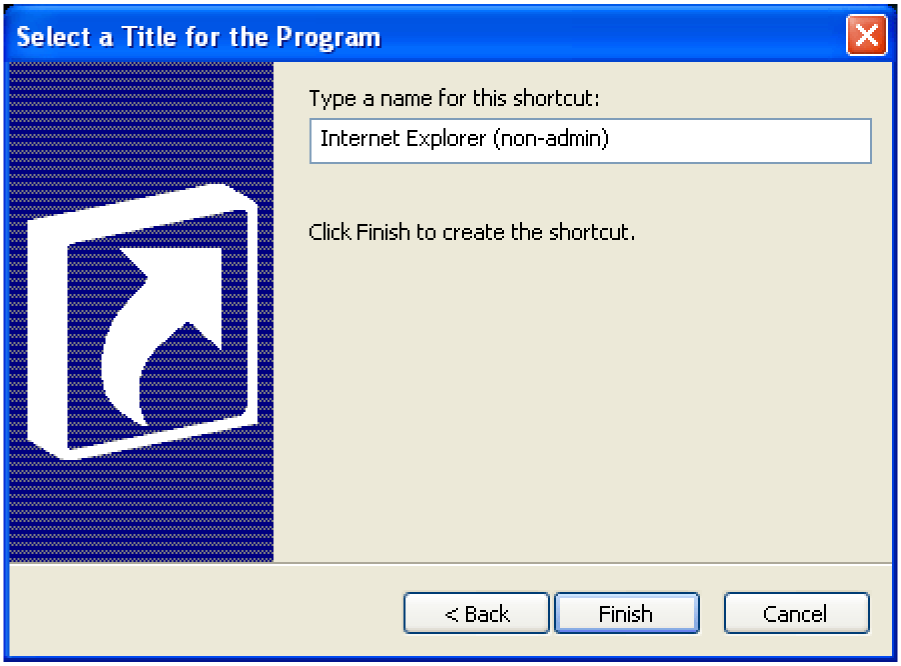
C:\warez\DropMyRights.exe "C:\Progra

Click Next to continue.

Figure 1. Path to application that you want to run in lower privilege

Updating the Shortcut Name

Next, update the name of the shortcut to represent the executable target, and not *dropmyrights*. I usually put the word "(Safer)" after the application name to denote this application will run in a safer security context. "(Non-admin)" is another common addition, as shown in Figure 2.



Select a Title for the Program

Type a name for this shortcut:

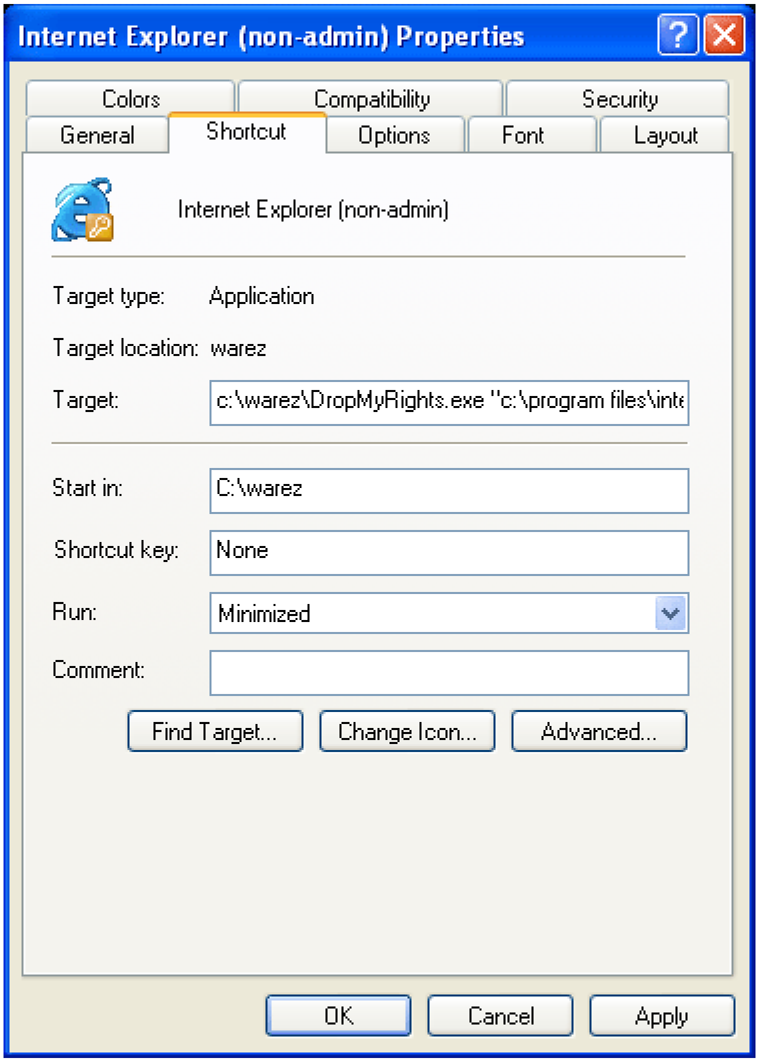
Internet Explorer (non-admin)

Click Finish to create the shortcut.

Figure 2. Updating the shortcut name

Setting the Icon and Run Mode

Finally, once the shortcut is created, set the **Run** option for the shortcut to **Minimized** and if you want, select a new icon.



Internet Explorer (non-admin) Properties

Internet Explorer (non-admin)

Target type: Application

Target location: warez

Target: c:\warez\DropMyRights.exe "c:\program files\inte

Start in: C:\warez

Shortcut key: None

Run: Minimized

Comment:

Figure 3. Setting the Run option to Minimized and optionally changing the icon

Advanced Options

The arguments to **DropMyRights** are:

Copy Code

```
DropMyRights {path} [N|C|U]
```

The meanings of the variables are:

- `Path`

  is the full path of the application to launch.
- `N`

  means run the application as a normal user. This is the default if you provide no argument.
- `C`

  means run the application as a constrained user.
- `U`

  means run the application as an untrusted user. Chances are, this will cause some applications to fail.

The best way to identify what each of these settings does is to look at the resulting process token. The following tables show the changes made to the process token.

Table 1. Administrative Account

SIDS	Restricting SIDS	Privileges
DOMAIN\Domain Users Everyone BUILTIN\Administrators BUILTIN\Users NT AUTHORITY \INTERACTIVE NT AUTHORITY \Authenticated Users \LOCAL	None	SeChangeNotifyPrivilege SeSecurityPrivilege SeBackupPrivilege SeRestorePrivilege SeSystemtimePrivilege SeShutdownPrivilege SeRemoteShutdownPrivilege SeTakeOwnershipPrivilege SeDebugPrivilege SeSystemEnvironmentPrivilege SeSystemProfilePrivilege SeProfileSingleProcessPrivilege SeIncreaseBasePriorityPrivilege SeLoadDriverPrivilege SeCreatePagefilePrivilege SeIncreaseQuotaPrivilege SeUndockPrivilege SeManageVolumePrivilege SeCreateGlobalPrivilege SeImpersonatePrivilege

Table 2. Normal User ('N')

SIDS	Restricting SIDS	Privileges
DOMAIN\Domain Users Everyone BUILTIN \Administrators ☒ BUILTIN\Users NT AUTHORITY \INTERACTIVE NT AUTHORITY \Authenticated Users LOCAL	None	SeChangeNotifyPrivilege

Table 3. Constrained ('C')

SIDS	Restricting SIDS	Privileges
DOMAIN\Domain Users Everyone BUILTIN \Administrators ☒ BUILTIN\Users NT AUTHORITY \INTERACTIVE	DOMAIN\Domain Users Everyone BUILTIN\Users NT AUTHORITY \INTERACTIVE NT AUTHORITY \Authenticated Users	SeChangeNotifyPrivilege

NT AUTHORITY \Authenticated Users LOCAL	LOCAL NT AUTHORITY \RESTRICTED	

Table 4. Untrusted ('U')

SIDS	Restricting SIDS	Privileges
DOMAIN\Domain Users ☒ Everyone BUILTIN \Administrators ☒ BUILTIN\Users NT AUTHORITY \INTERACTIVE NT AUTHORITY \Authenticated Users LOCAL ☒	NT AUTHORITY \RESTRICTED Everyone NT AUTHORITY \INTERACTIVE NT AUTHORITY \Authenticated Users BUILTIN\Users	SeChangeNotifyPrivilege

The red cross mark means the SID is still in the token, but it is a deny SID. A SID with this attribute is a deny-only SID. When the system performs an access check, it checks for access-denied ACEs that apply to the SID, but it ignores access-allowed ACEs for the SID.

The biggest privilege and SIDs delta is between the administrative account and the normal user account. As you can see, all privileges are stripped from the token except the Bypass Traverse Checking privilege (also know as **SeChangeNotifyPrivilege**.) Constrained and untrusted are smaller deltas from normal user, and you may start to see some applications fail with security restriction errors. My opinion is use Normal (the default) for most things, and Constrained if you think you'll be browsing hostile or potentially dangerous Web sites.

Spot the Security Defect

A good number of people worked out the bug in my last article. The **CreateFile** function is opening the file for all access, when the code only reads from the file. FILE_ALL_ACCESS should be replaced with GENERIC_READ or similar. This is bad because in all likelihood only an administrator can use this code, not a normal user. I see this error often.

Can you spot this code flaw? This came across my desk the other day as a bug in some Java DNS stuff. It's an interesting bug, which I rewrote in C# and generalized the flaw.

Copy Code

```
Int16 req;
...
while (true) {
   getRequest();
   req++;
   arr[req] = DateTime.Now;
}
```

Michael Howard is a Senior Security Program Manager in the Secure Engineering group at Microsoft and is the coauthor of Writing Secure Code [http://www.microsoft.com/mspress/books/5957.asp] , now in its second edition, and the main author of *Designing Secure Web-based Applications for Windows 2000*. He is also a co-editor of *Basic Training* in IEEE Security & Privacy Magazine. His main focus in life is making sure people design, build, test, and document nothing short of a secure system. His favorite line is "One person's feature is another's exploit."