# Automated Checking for Windows Host Vulnerabilities [1]

Matin Tamizi, Matt Weinstein, and Michel Cukier

Center for Risk and Reliability,
Department of Mechanical Engineering
University of Maryland, College Park,
College Park, MD 20742
{matin, weinmatt, mcukier}@umd.edu

## Abstract

*Evaluation of computing system security requires knowledge of the vulnerabilities present in the system and of potential attacks against the system. Vulnerabilities can be classified based on their location as application vulnerabilities, network vulnerabilities, or host vulnerabilities. This paper describes Ferret-Windows, a new software tool for checking host vulnerabilities on the Windows platforms. This tool helps system administrators by quickly finding vulnerabilities that are present on a host. It is designed and implemented in a modular way: a plug-in module is used for each vulnerability checked and each possible output format is specified by a plug-in module. Moreover, several vulnerability fixing plug-in modules exist to help users remove specific vulnerabilities. As a result, Ferret-Windows is extensible, and can easily be kept up-to-date through the addition of checks for new vulnerabilities as they are identified. Finally, Ferret-Windows is a freely available open-source software.*

***Keywords:*** Security auditing tool, host vulnerabilities, Windows.

**Submission category**: regular paper.

**Contact author**: Michel Cukier

Department of Mechanical Engineering
2100H Marie Mount Hall
University of Maryland
College Park, MD 20742-7531, USA

Email: mcukier@umd.edu
Phone: 301-314-2804, Fax: 301-314-9601

[1] This research has been supported by NSF CAREER award 0237493.

## 1. Introduction

The security community had a great interest in host vulnerabilities about ten years ago. Since then, the focus has shifted towards network vulnerabilities because of the assumption that once an outsider finds a way to penetrate the network, it is very difficult to keep him or her from getting administrative privilege on hosts [Vie02]. This argument may make sense if one is focusing on the problem of attacks from outside, but it becomes less convincing when one is trying to evaluate the security of a computer network with respect to the possibility of both internal and external attacks. In the latter case, host as well as network and application vulnerabilities need to be considered.

Host vulnerabilities are understood in a broad sense in this paper. Some might argue that some of the vulnerabilities are in fact poor configurations or that some of the vulnerabilities are features provided to the user. Despite such arguments, we simply use the term "vulnerability" for anything identified as potentially exploitable by attackers.

Several tools were developed a decade ago to check UNIX host vulnerabilities (e.g., COPS [Far90] and Tiger [Tiger]). COPS and Tiger check for vulnerabilities that are no longer relevant, offer no simple way to modify the list of checked vulnerabilities, and do not check certain important current vulnerabilities. Furthermore these tools were not designed using a modular approach, making updates to checked vulnerabilities difficult. For all these reasons, we developed a new tool, called *Ferret*, which is designed to check UNIX host vulnerabilities [Sha04]. Ferret is designed in a modular way: a plug-in module is used for each vulnerability checked, and the format of the output is specified by different plug-in modules. As a result, Ferret is extendible, and can easily be kept up-to-date through addition of modules to check for new vulnerabilities as they are discovered. The modular approach enables the provision of specific configurations of Ferret tailored to specific operating systems or user environments. Ferret is a freely available open source software implemented in Perl.

After focusing on UNIX host vulnerabilities, we targeted host vulnerabilities on Windows platforms. Several tools have been developed to check for Windows host vulnerabilities. Microsoft's Baseline Security Analyzer [BSA] checks for unapplied patches, misconfigurations, and is a general use vulnerability assessment tool for Windows. However, the Baseline Security Analyzer only focuses on Microsoft products and is not open source. WinGuides Tweak Manager [Tweak] creates a graphical interface for many local settings and allows manual inspection for potential vulnerabilities. The inspection is facilitated since the configurations are placed in categories. However, Tweak Manager is neither free nor open source. ISS Internet Scanner [ISS] and eEye Retina Network Security Scanner [Retina] check for different host and network vulnerabilities. The evaluation version of Retina just lists the number of vulnerabilities found and does not provide information on how to fix these vulnerabilities. Moreover, Internet Scanner and Retina, which are commercial products, are also not free or open source. After reviewing existing tools that check for host vulnerabilities on Windows, the need for a tool that covers a wide range of vulnerabilities, supports the addition of new modules, automates what Windows system administrators are doing manually, and is free and open source becomes apparent and will greatly benefit the security community.

Thus, this paper describes Ferret-Windows, a host vulnerability-checking tool specifically for Windows vulnerabilities. The structure of Ferret-Windows includes four parts: the Ferret-Windows core, vulnerability checking plug-in modules, vulnerability fixing plug-in modules, and output plug-in modules. The Ferret-Windows core is responsible for checking the version of the computer's operating system, checking for dependencies, running the vulnerability checking plug-in modules, and running the output plug-in modules. Each vulnerability checking plug-in module performs one action, such as checking for one vulnerability or listing users in one specific group. These plug-in modules can be classified into five families: storage, security policies, domain settings, local settings, and passwords. Plug-in modules have been included that fix vulnerabilities found by the vulnerability checking plug-in modules. Ferret-Windows creates a consolidated presentation of the individual plug-in module outputs in the form of text and HTML.

The remainder of the paper is organized as follows. The detailed design of Ferret-Windows and an overview of Ferret-Windows' architecture are described in Section 2. Section 3 lists the current set of available vulnerability-checking plug-in modules classified into five families. Section 4 describes how to run Ferret-Windows and provides sample output generated by the output plug-in modules. Finally Section 5 concludes the paper.

## 2. Ferret-Windows Detailed Design

The first part of this section describes the design choices made for Ferret-Windows, highlighting the ones that differed from Ferret. The second part of the section presents an overview of the Ferret-Windows architecture. For the rest of the paper, we will use "module" and "plug-in" for "plug-in module."

## 2.1 Some Design Choices on Ferret-Windows

**Fundamental Architectural Differences between Windows and UNIX-based Platforms.** Windows and UNIX share several basic computing principles of common client-server architecture. However, the implementation of these features in both systems is vastly different. One of the areas in which these differences are most apparent is security [Windows]. As opposed to UNIX, the Windows NT security architecture offers an extremely high degree of granularity with regard to users and objects. The NT security model is built to handle all sizes of networks from a simple single server office to a multinational corporation. However, the fundamental design of the security model remains constant. This leads to a very specific and secure model for access control when employed properly. Some variations of UNIX offer similar capabilities, but these types of architectures typically are not portable and are proprietary in nature. For instance, permissions utilities for UNIX network file systems such as AFS must be written differently than, for example, Apple's "Windows-compatible" permission scheme in OS X Server. The Windows architecture has a single code source, as opposed to UNIX, which has several major variations (SCO, BSD, Mach, and Linux) of the kernel that perform the same functionality. Therefore, Ferret-Windows provides host-vulnerability checking to a larger and more diverse user base, as Windows is the dominant desktop and server operating system used by businesses.

**Substituting for Perl.** While Perl is a commonly available open-source, cross-platform scripting language, it has several significant flaws when it comes to a Windows installation. Most importantly, there is no commonly accepted non-commercial, stable build of Perl available on Windows. ActivePerl [ActivePerl], which is the most prevalent version of Perl for Windows, is not open-source. The goal of Ferret-Windows is to provide a trusted easy-to-use but comprehensive host vulnerability checker that is open source. Therefore using ActivePerl may compromise this trust because it consists of an untrusted binary from an external source. Additionally, configuring ActivePerl for use with Windows may be too troublesome for average Windows administrators to handle, thus preventing them from running Ferret-Windows.

**Administrative Capabilities and their Effect on Ferret-Windows.** Windows was designed to be easier to configure and manage than a comparable UNIX-based system. This concept has been furthered significantly with the introduction of Windows Server 2003. Thus, it is apparent that Windows system administrators can possess a more limited level of technical expertise than a UNIX system administrator and yet perform the same job. This may have a negative impact on security: these administrators have less knowledge of the architecture and capabilities of the operating system they are administrating. Thus, in porting Ferret to Windows, we must take into account that the general level of user expertise may be less than of UNIX users. Therefore using Ferret-Windows must be concise and uncomplicated, while providing the user with useful information.

**Finding a Suitable Replacement.** Several methods are available to access system Application Program Interfaces (APIs) in Windows: the use of utilities; C++ access to APIs; and Windows Script Hosting (WSH, which encompasses the use of VBScript and Jscript).

We eliminated the use of utilities because we need scripting capabilities greater than simple batch files and executable commands. C++ was ruled out because one of the main design tenets of Ferret-Windows is the trust that the code run on the system is not malicious. In the UNIX version of Ferret, by using Perl, all code is freely visible and compiled at runtime and can be reviewed prior to launch. C++ requires the users of Ferret-Windows to either compile the code themselves or risk running an untrusted executable.

Thus, Windows Script Hosting (WSH) provides the best platform for Ferret-Windows. Because WSH is an integral part of all Windows variants, there is no need to install programs or make configuration changes to a machine to run Ferret-Windows. WSH contains an extensive library of functionalities, including text processing, regular expressions, and a simple syntax structure. WSH is not an open source product; however, in the interest of trust, WSH is authored by Microsoft exclusively and is fully integrated with all versions of Windows. Additionally, because WSH is not a scripting language, modules can be written in other languages and modules that require the installation of additional tools and packages (for example, the Group Policy Management Console) can also be developed.

**Need for Python**. Due to the shortcomings in VBScript's socket API in WSH, Python is used for any interface with the network. According to Microsoft, VBScript requires a compiled socket interface that is not included with Windows to provide access to the socket interface during run time. To create such an interface for Windows, a licensed version of Microsoft Visual Studio is needed. Python comes with an open source interface to the socket library, so instead of creating a socket interface using a programming language like Visual Basic .NET, we decided to use a standard socket interface.

## 2.2 Architecture Overview

The architecture of Ferret-Windows consists of four components: the Ferret-Windows core, vulnerability checking modules, vulnerability fixing modules, and output modules (see

Figure 1). Note that vulnerability fixing modules were not provided in the UNIX version of Ferret. While fixing vulnerabilities are straightforward in UNIX, some of the vulnerabilities checked in Ferret-Windows require a high level of expertise to fix. When designing Ferret-Windows, we assumed that some users would not have that level of expertise and therefore provided a family of vulnerability fixing modules. Since the remaining part exclusively focuses on Ferret-Windows, from this point forward we refer to Ferret-Windows as simply Ferret.

The Ferret core runs the vulnerability checking modules that scan the computer for vulnerabilities (one module per vulnerability) and sends the results to a text file. The core then runs the output modules that parse the text files created by the vulnerability checking modules and produces a readable output for the user. Users must run the vulnerability fixing modules individually to avoid any unwanted changes to the machine and separate fixes that require privileges from fixes that do not.

All of Ferret modules are self-reliant and can run independently without the Ferret core or the other modules. The modularity of Ferret provides advantages for both developers and users. Developers can write modules for Ferret without regard to any of the other modules, allowing contribution to the project from the Open Source Community. Users can run any one of the Ferret modules individually within the Ferret package or distribute a vulnerability checking module to other computers without having to use the entire Ferret package.

**Ferret core.** The Ferret core is responsible for checking the version of the computer's operating system (OS), checking for dependencies, running the vulnerability checking modules, and running the output modules. Users can modify the presentation of the output through the output modules without having to change the Ferret core. After checking the OS version, Ferret determines which modules can run on the current system. Ferret notifies the user it cannot run any modules if the system uses an OS other than Windows NT 4.x or 5.x, which includes Windows NT 5.0 (Windows 2000), Windows NT 5.1 (Windows XP), and Windows NT 5.2 (Windows 2003). Since some possible modules are not applicable to some versions of Windows NT, functionality that allows Ferret to discriminate between modules that depend on a specific version of Windows NT can be easily added (e.g., a vulnerability that only exists on Windows XP). Dependency checking allows Ferret to determine whether the current system can run certain modules that require installation of packages that Ferret does not provide. We currently check for the Python package to run the modules that use the Python interpreter since this is a package not provided in the Ferret package. However, modules themselves could also check the installation of external packages or executables required.

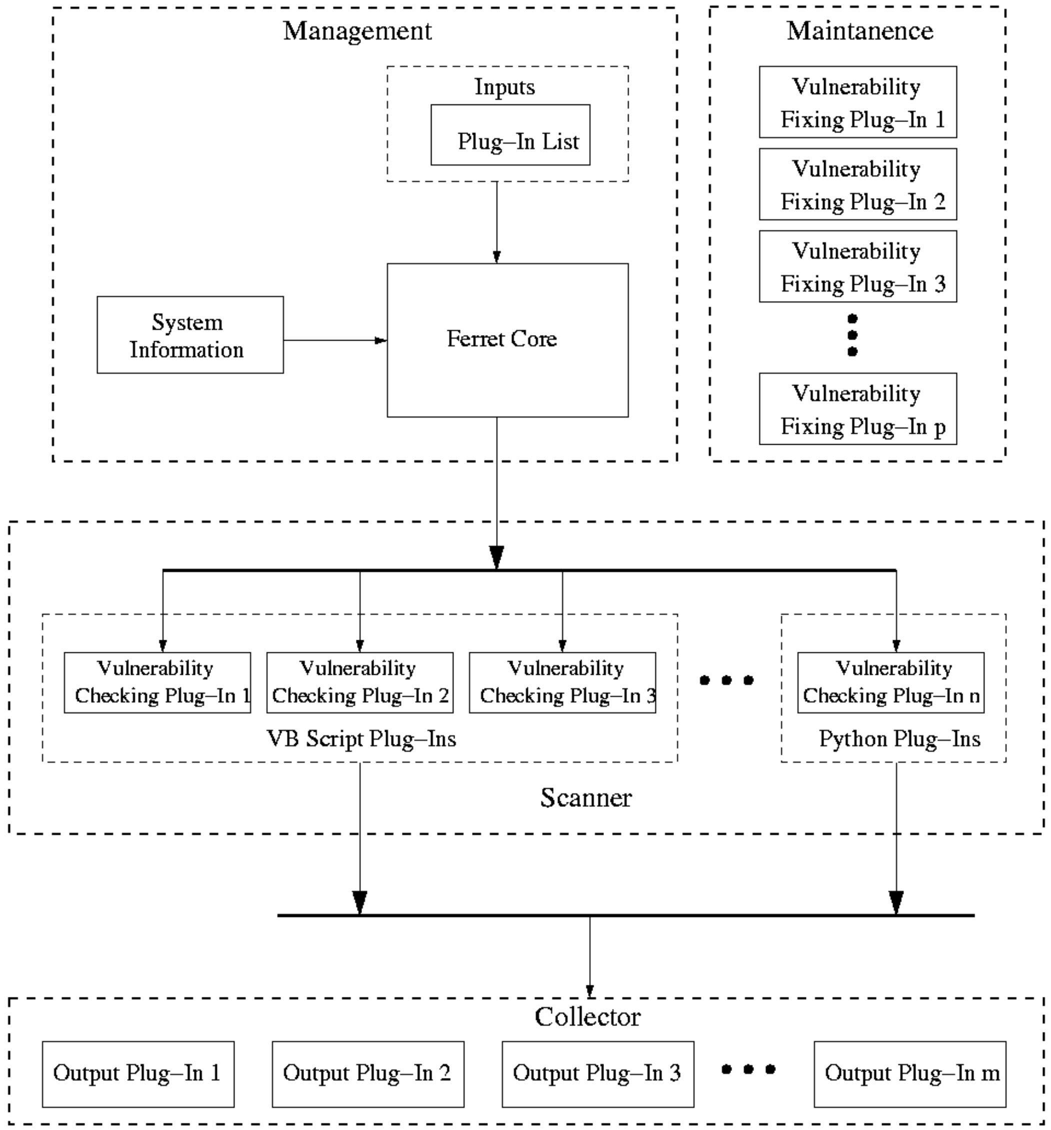

Management
Inputs
Plug–In List
System Information
Ferret Core
Maintanence
Vulnerability Fixing Plug–In 1
Vulnerability Fixing Plug–In 2
Vulnerability Fixing Plug–In 3
Vulnerability Fixing Plug–In p
Vulnerability Checking Plug–In 1
Vulnerability Checking Plug–In 2
Vulnerability Checking Plug–In 3
Vulnerability Checking Plug–In n
VB Script Plug–Ins
Python Plug–Ins
Scanner
Collector
Output Plug–In 1
Output Plug–In 2
Output Plug–In 3
Output Plug–In m

Figure 1. Overview of Ferret Architecture

**Vulnerability Checking Modules.** Each module performs one action, such as checking for one vulnerability or listing users by categories. Vulnerabilities can be classified into five different families (see the discussion in Section 3): storage, security policies, domain settings, local settings, and passwords. The modules can use any scripting language as long as the interpreter is installed, but as discussed in Section 2.1, we currently use Visual Basic Script and Python. Support for JavaScript, which WSH supports natively, can easily be added. The modules receive a command line argument containing the name of the directory in which the modules create a file containing the output; otherwise, the modules create the output file in the current directory. By allowing the modules to execute independently, the modules can execute concurrently, which relieves the bottleneck created by modules that have a long execution time.

**Vulnerability Fixing Modules.** Some modules that fix configuration vulnerabilities are included. Fixing vulnerabilities can be troublesome when registry modifications are required. While a script can easily (in terms of lines of code) modify the registry, it can be time consuming or unintuitive to fix manually. For example, a module that can configure Windows to clear the Page File when shutting-down (an action not enabled by default in Windows) is available. Without this module, correcting the Page File vulnerability would require directly modifying the value of a registry key. Most of the vulnerability fixing modules requires administrator-level privilege. Modules that do not require extra privileges are also supported.

**Output Modules.** Users can create a consolidated presentation of the individual modules outputs in the form of text and HTML. To keep the vulnerability checking modules independent of the output modules, they create a consolidated presentation by iterating over the individual outputs created by the vulnerability checking modules. Output modules can also be added that create a presentation in LaTeX, which can be converted to PDF, Postscript, RTF, and many other formats.

## 3. List of Windows Vulnerabilities Checked

In this section we list the different vulnerabilities currently checked by Ferret. The modules listed in this section are the “low hanging fruits.” The modules implemented thus far center on the most common and critical areas that attackers might choose to probe for access. For each vulnerability, we motivate our choice of checking this particular vulnerability and describe how we implemented the module. Ferret modules are organized into several families, each of which corresponds to a certain functionality. Currently, there are five distinct families, which are the areas of focus that represent critically important spheres of vulnerability for a Windows-based system: storage, security policies, domain settings, local settings, and passwords.

## 3.1 Storage

Windows traditionally runs on NTFS-based (NT File System) storage equipment. The actual storage is accomplished through several means, including RAID arrays, Network Attached Storage (NAS), and Storage Area Networks (SAN), but the Windows Active Directory and File Permissions systems are designed in such a manner to allow for discrete, granular permissions to be applied on individual users, large groups, and entire enterprises, no matter the medium. Windows uses the principle of individual network “shares” to provide network access to a given repository of files, from a single directory or an entire disk drive. Administrators of Windows systems will need to verify that network shares present on their systems are secured and not vulnerable to unauthorized usage. Thus, Ferret contains several modules that are designed to give a useful overview of file shares, permissions inherent to them, and permissions of their root directories.

**Check_Shares.** This module is designed to determine the names, UNC paths, roots, and types of each file share present on the machine. This module lists all shares including hidden ones, as well as important information about each one. The following pseudo-code shows how the module accomplishes these tasks.

```
  get system UNC name;
  retrieve WMI object for this system using winmgmts
interface;
  execute query to retrieve collection of all file shares
  for each share
  {determine share display name;
   if share has maximum connections set, display this value;
   display UNC path, Root path;
   determine type of share (File, Administrative, Printing)
and display;
  }
```

**Check_Shares_GROUPNAME_PERMISSION.** This module determines which of the shares have the desired PERMISSION set for the GROUPNAME. The module is implemented such that users of Ferret can easily duplicate it and substitute the desired GROUPNAME or PERMISSION that they wish to verify. This module takes advantage of a tool that has been incorporated for use with Ferret, included from the Windows 2003 Resource Kit published by Microsoft. The tool returns a raw list of all shares and permissions, which this module filters appropriately.

```
  determine system UNC name;
  execute and redirect output of tool;
  loop until end of stream
  {read output, detect new share;
```

```
   check listed permissions for desired PERMISSION and GROUPNAME;
   if present, display;
   increment count of detections;
  }
  return shares detected and total number of detections;
```

**Check_Shares_Root_GROUPNAME_PERMISSION.** Windows maintains a separate permission list for share objects that are different than permissions present on the root directory of the share itself. In other words, this separate permission list controls access to the share itself but not the files within. This two-tier concept can be confusing, so it is important to give administrators the ability to determine permissions. This module also employs a tool from the Windows 2003 Resource Kit, which when passed a file system path, returns permissions present on that path.

```
  get system UNC name;
  retrieve WMI object for this system using winmgmts interface;
  execute query to retrieve collection of all file shares
  for each share
  {   execute and retrieve output from utility for share path;
   parse permissions for GROUPNAME and PERMISSION
   if present, display
   increment count of detections
  }
  return share UNC paths and root paths, as well as total number of detections;
```

Ferret contains several duplications of the Check_Shares and Check_Shares_Root modules that detect common and presumably insecure GROUPNAME and PERMISSION combinations. Some of these include Everyone/Full Control, Everyone/Change, Domain Users/Full Control, Domain Users/Change. The Windows permissions system prioritizes DENY permissions over any ALLOW permissions, and because these DENY permissions can be inherited from parent objects, administrators can have trouble identifying the cause of mismatched permissions, the cause of which is often inherited DENY. Thus, the Check_Shares_Root module can also detect and identify DENY permission that are present on roots.

Additional modules in the family for Storage include FAT_FS, which detects the presence of the FAT filesystem. Windows servers and desktop operating systems are capable of being run on this older filesystem, as this was the default filesystem for prior Windows versions (3.1, 95, 98, Me). The FAT filesystem is an important vulnerability to detect, because FAT does not have a discrete permissions structure like the NT file system, and thus does not have an adequate level of security.

## 3.2 Security Policies

The Windows mechanism for distributing security policies and settings, Group Policy, is an important area for survey. When computers are joined to a Windows domain, they retrieve a policy from Domain Controllers that is applied to the local machine.

Because Group Policy is not a native WSH interface, users must install the Group Policy Management Console (GPMC) that is freely distributed by Microsoft. This installation provides a WSH interface, which these modules rely on to function. GPMC is compatible with all versions of Windows on which Ferret runs.

Group Policies contain two major forms of information, Policy-based Settings and registry-based Settings. We will cover the latter in domain settings (Section 3.3), as these settings are not specific to the security policy, but actually are overlays that supercede any existing domain and local Settings. Thus far, we have implemented four modules in this family.

**GPO_Report.** This module is responsible for querying the Group Policy Management (GPM) interface on the executing machine and displaying the base information about each Group Policy Object (GPO) in effect, including name, GUID, and number of Scopes of Management (SOMs) of which the policy is a member. Because domain controllers maintain a copy of all GPOs, this module shows all group policies available to the domain, while on other machines, this module shows only the group policies currently in effect.

```
retrieve and qualify local domain name from LDAP;
query GPM interface, retrieve collection of policies;
for each policy
{   determine name and GUID;
    query SOM interface, retrieve collection of SOMs;
    display name, GUID, and count of SOMs;
}
```

**GPO_Permissions.** This module is responsible for retrieving the list of GPOs and the permissions specified for each user or group. This is done by querying the security information stored for a given GPO. The code for this module is similar to the GPO_Report module, using the same interface for retrieving the data.

```
retrieve and qualify local domain name from LDAP;
query GPM interface, retrieve collection of policies;
for each policy
```

```
  {   retrieve security info;
      determine number of security entries;
      for each entry
      {    determine user or group name;
           classify permissions applied;
           display name and permission type;
      }
  }
```

**GPO_Password_Policy and GPO_Lockout_Policy.** These two modules are similar thus they can be explained concurrently. When a GPO is retrieved from the GPMC, it can be parsed into a XML format, which contains the policy-based settings. Some of these settings, which contain important information about access control, are the description of settings, which bind password length, complexity and age requirements as well as lockout applicability, and timeframe settings. Because Windows prioritizes the "stricter" settings, we poll all GPOs and compare the results to obtain the actual effective setting. These settings are displayed by the following code.

```
  retrieve and qualify local domain name from LDAP;
  query GPM interface, retrieve collection of policies;
  for each policy
  {   retrieve XML policy;
      parse policy to retrieve desired information;
      compare settings to placeholders, if stricter, replace
in placeholder;
  }
  display results of module;
```

### 3.3 Domain Settings

Any Windows-based computer has a number of inherited settings from GPOs. This family of modules identifies specific policies in the registry of local machines that are bound by Group Policy. They are not local settings in the sense that control over them is inherited by the local machine from the domain. Each of the modules is similar, as the creation of these module required more research than code (one needs the appropriate registry key to retrieve the setting). A total of eight modules are currently available.

- **Reg_Drive_Restrict.** Check if CD-ROM/Floppy is accessible only to console user
- **Reg_No_C-A-D.** Check if Ctrl-Alt-Delete is required for authentication
- **Reg_Logons_Cache**. Check number of cached logons
- **Reg_Anony_Shares.** Check for Anonymously Accessible File Shares
- **Reg_AutoPlay.** Check if AutoPlay is enabled
- **Reg_LocalProfiles**. Check if Local Profiles are required

- **Reg_TSC_Restrict**. Check if users are restricted to a single TS session
- **Reg_Port_Redirection.** Check if COM, LPT, or Client Drive Redirection is enabled

Each of the modules is written to retrieve and report the value and meaning of the registry key desired. For example, on Reg_Port_Redirection the module can determine whether any of the three redirections, COM, LPT, or Client Drives, are disabled, and which if any are enabled.

```
access local registry, retrieve key and poll key for value;
display key name, value;
determine value properties and meaning;
display results;
```

## 3.4 Local Settings

Local settings include many potential configuration vulnerabilities, but the local settings family focuses on configurations that lead to vulnerabilities only for the local machine and local users. Storage issues (Section 3.1) and passwords problems (Section 3.5) are not included in the local settings section since they also deal with the domain and the network.

Most of the vulnerability checking modules included in the local settings family use the registry to find configurations that could lead to vulnerabilities. Windows does not offer a GUI to change the local settings. The user has to change the local settings by directly modifying the registry through regedit.exe, or by using a script or a third-party application. These modules thus allow the user to automatically check for a potential vulnerability based on the value of the registry key instead of having to review the registry to find the right key and its value among thousands of registry keys. The following modules check for configurations issues leading to vulnerabilities based on the registry key values.

- **Reg_MessengerServ.** Check if Windows Messenger Service is disabled
- **Reg_AutoUpdate-Check.** Check if Windows checks for updates automatically
- **Reg_AutoUpdate-Install.** Check if Windows installs updates automatically
- **Reg_ClrMRU.** Check if Windows clears the cached run commands
- **Reg_ShellExt.** Check which users Windows enforces shell extensions for applications
- **Reg_RemoteAuthenAttempts.** Check how often Windows allows a failed remote authentication attempt before the remote connection is disconnected
- **Reg_RemoteAccessLockout.** Check how often Windows allows a failed remote authentication attempt before locking out the account
- **Reg_LastUser.** Check if Windows shows the last user in the authentication screen

- **Reg_UserTrack.** Check for which users Windows keeps track of the users' actions
- **Reg_WindowsFirewall.** Check if the Windows firewall is active
- **Reg_ShutdownNoLogin.** Check if Windows allows users to shutdown without login
- **Reg_LockComp.** Check which users Windows allows to lock the computer
- **Reg_MyCompProp.** Check for which users Windows makes available the properties option on My Computer
- **Reg_FolderOptions.** Check for which users Windows makes available the folder options menu in Explorer
- **Reg_FileMenu.** Check for which users Windows makes available the file menu in Explorer
- **Reg_NewMenu.** Check for which users Windows makes available the new menu item in Explorer
- **Reg_ExpToolbars.** Check for which users Windows allows to customize the Explorer toolbars
- **Reg_ModToolbars.** Check for which users Windows allows to change or hide toolbars
- **Reg_ActvDeskRestriction.** Check which of the following rights are available in Active Desktop: change wallpaper, change user components, add components, delete components, add Taskbar's toolbars, drag Taskbar's toolbars, drop Taskbar's toolbars, close the Taskbar's toolbars, modify the desktop toolbars, and use HTML for wallpaper
- **Reg_SecureDesktop.** Check if Windows has secure desktop restrictions enabled
- **Reg_DesktopHide.** Check for which users Windows does not hide all the programs and items on the desktop
- **Reg_AuthForUnlockScreenSave.** Check if Windows requires a full login when unlocking or returning from screen saver
- **Reg_ClrPF.** Check if Windows clears the Page File on shutdown
- **RemDesktopAccess.** List all users with Remote Desktop access
- **AdminRights.** Display all of the users with administrative privileges
- **RebootAfterCrash.** Check if Windows reboots the system after a crash or displays a blue error screen.

The following vulnerability checking modules are part of the local settings family but are not based on the use of registry keys.

**Disable "Adminstrator" Account.** This module refers to the use of the local account Administrator, which exists on all Windows NT machines. Enabling the default administrator creates a high privilege account with a known user name. If an attacker does not know which account has privileged rights on the computer, it is more difficult for the attacker to gain privileged access. The default administrator account becomes the first account that attacker try to break into. To check for this vulnerability we use the local Windows Management Provider in VBScript to obtain a Windows Management Interface (WMI) to the Administrator account and check whether or not the account is active.

**Only Administrators**. This module refers to the case when the machine only has users in the Administrators group. In this case, all programs on the machine are being run with privileged access, which causes a vulnerability in the system. Even if each person has to use two user accounts (i.e., one with privilege and one without), most operations, such as checking email and using a web browser, should be accomplished by an unprivileged user while the person can use the Run As feature in Windows for actions that require escalated privilege, such as installing a program or modifying the Control Panel. The module uses the local Windows Management Provider in VBScript to obtain a WMI interface for every user on the local machine and checks if all users, excluding guests, have administrative privilege.

**Set the Minimum Password Length.** This module defines the minimum password length for local users by machine and user. If the local machine allows passwords shorter than eight characters, we use WMI to obtain a list of all local users and display the username for all users that may have passwords shorter than eight characters. The fix for this vulnerability is to change the machine minimum password length to eight characters.

## 3.5 Passwords

The vulnerability checking modules focusing on passwords address actual passwords that are used in the system as opposed to the configurations that govern those passwords. Password modules check for weak passwords (including blank passwords) used in the Windows system and in other programs like Remote Desktop, VNC, and Telnet that allow remote access to the system. Ferret is not designed to crack passwords nor does Ferret obtain the password hashes stored in Windows. Instead the Windows API and the network interface are used to try to find the password. Ferret's approach is similar to the approach many attackers would take to break into the system by finding a weak or blank password (i.e., attempt simple passwords and move on hoping one of the users has a weak enough password). Three modules are implemented that check for blank passwords amongst Windows users, extremely weak passwords amongst Windows users, and blank passwords used for VNC (Virtual Network Computing). Note that VNC is currently the only remote access application supported.

**Windows Blank Passwords**. This module iterates through all of the active users and checks if the user has a blank password by using the following algorithm.

```
  use the Windows Management Provider to obtain the all active
user objects with the WMI interface;
  for each active user
      if user-->password is blank
           add user to list;
  display list
```

**VNC Blank Passwords.** This module uses a more complicated approach to determine whether the VNC service, if present, allows connections without a password. The module requires that a connection from the local host is enabled and that VNC runs on the default port (5900). The following algorithm was implemented in Python due to the limitations of VBScript discussed in Section 2.1.

```
  tries <-- 0;
  while (tries < 10) or (no error in VNC connection)
      connect to socket 5900 on the current computer;
      use RFB protocol to receive message from VNC server
      if VNC server replies with 1
           no authentication required;
      tries <-- tries + 1;
      sleep(tries)
  end while
  if error in VNC connection
      print error message;
  return authentication type
```

The RFB (Remote Frame Protocol) used by VNC sends the authentication level (authentication required, authentication not required, or connection not allowed) during the second transaction. Unfortunately, the server tends to fail on connection occasionally, so the module tries to connect ten times with an increasing delay between each connection attempt.

**Extremely Weak Windows Password.** This module uses the Windows API to attempt passwords for each user. Unlike PWDump [PWDump] that dumps the hashes of the passwords, this module actually attempts each password in a novel manner. Where PWDump tricks LSASS.exe (the application that interfaces with the secured password database) to dump the hashes of all user passwords, LSASS.exe is used to attempt passwords much faster than a user could by hand. The module can only check approximately three passwords per seconds, so we cannot check a large list of passwords. However, we can check the most likely passwords, such as the username, the user's first name and last name in different combinations, and a few very common passwords. Common passwords include "password", "12345678", and "abcdef", are stored in a file and checked for every user. The following algorithm was implemented in VBScript.

```
  use the Windows Management Provider to obtain the all active
user objects with the WMI interface
  passwords <-- read in password list from file;
  for all active users
      obtain the WinNT Provider for the user
      pwd_found <-- False;
      for each passwd
           ChangePassword passwd, passwd;
           if no errors
                pwd_found <-- True;
      end for
  end for
```

## 4. Using Ferret

In this section we briefly describe how to run Ferret and provide sample output generated by the text file and HTML output modules.

**Running Ferret.** Ferret is easily downloadable at http://www.sourceforge.net/projects/ferret. The contents of the "Ferret Archive" can be unzipped into any folder. Ferret is run by double clicking on "ferret.vbs". Some anti-spyware programs such as Ad-aware or Microsoft Anti-Spyware may prompt to allow "unauthorized" scripts (i.e., Ferret) to run.

Customizing Ferret is accomplished by editing the "plugins.txt" file in the "plugins" folder to include each module, by name, one per line. Ferret comes by default with a "plugins.txt" file that includes all available modules. Modules can be "commented out" by adding a double slash ( "//" ) to the line that a module is listed on.

**Ferret Output.** Ferret prepares an output directory which contains two forms of output: a text file of the output of each module and an easy to read HTML file which contains expandable boxes for each module that contain the module's output.

For example, when run on one of the machines at the University of Maryland, the output of the Check_Shares_Root_Everyone_Readable module was:

```
  Shares whose root directories have Everyone, Read [R] or Read [RX]
permissions:
  \\COBALT\microsoft  frontpage,  Path:  "C:\Program  Files\microsoft
frontpage"

  There are 1 total shares with these permissions.
```

The output of the Shares_Everyone_Writable module was:

```
Shares that have Everyone, Full Control permissions:

\\COBALT\downloads
\\COBALT\Invoices
\\COBALT\hpdeskje

There are 3 total shares with these permissions.
```

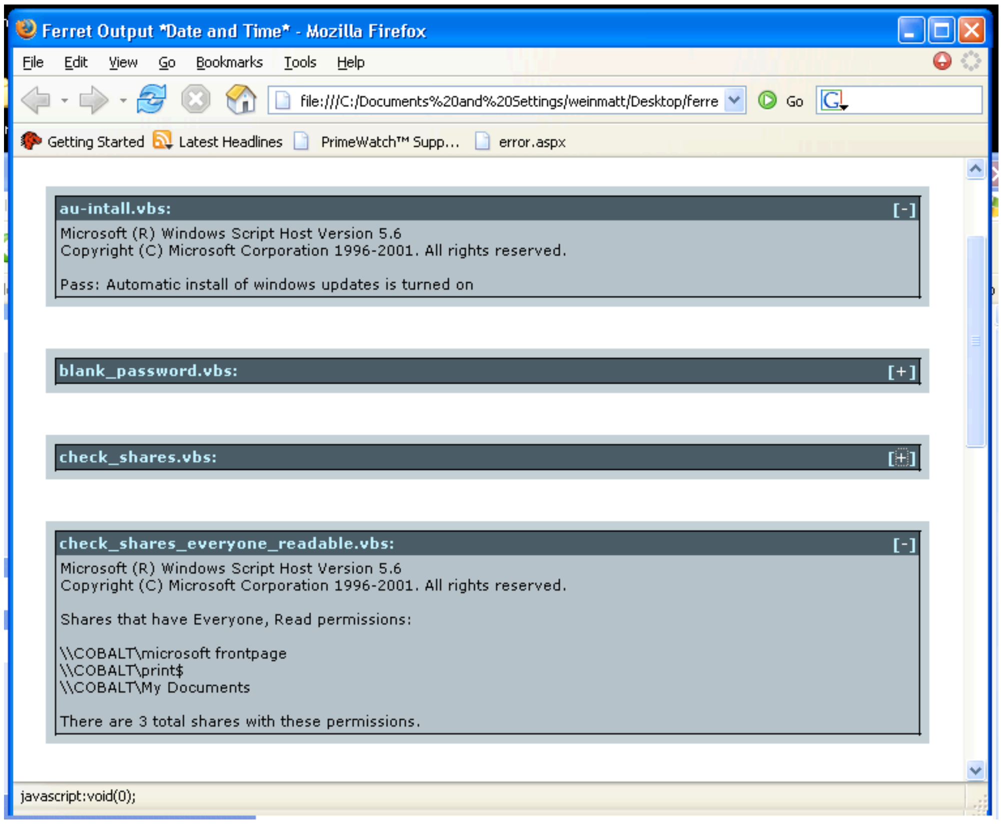
And, finally, the first lines of the output of the Check_Shares module were:

```
Share Name: microsoft frontpage
UNC Path: \\COBALT\microsoft frontpage
Path: C:\Program Files\microsoft frontpage
Type: File Share

Share Name: My Documents
UNC Path: \\COBALT\My Documents
Path: C:\Documents and Settings\weinmatt\My Documents
Type: File Share

Share Name: Remote IPC
UNC Path: \\COBALT\IPC$
Path:
Type: Administrative Share
```

Finally, Figure 2 presents an example of the HTML output file.


Ferret Output *Date and Time* - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
file:///C:/Documents%20and%20Settings/weinmatt/Desktop/ferre
Go
Getting Started
Latest Headlines
PrimeWatch™ Supp...
error.aspx
javascript:void(0);

```
au-intall.vbs:                                                    [-]
Microsoft (R) Windows Script Host Version 5.6
Copyright (C) Microsoft Corporation 1996-2001. All rights reserved.

Pass: Automatic install of windows updates is turned on

blank_password.vbs:                                               [+]

check_shares.vbs:                                                 [+]

check_shares_everyone_readable.vbs:                               [-]
Microsoft (R) Windows Script Host Version 5.6
Copyright (C) Microsoft Corporation 1996-2001. All rights reserved.

Shares that have Everyone, Read permissions:

\\COBALT\microsoft frontpage
\\COBALT\print$
\\COBALT\My Documents

There are 3 total shares with these permissions.
```

Figure 2. Ferret HTML Output Format

## 5. Conclusions and Future Work

This paper described Ferret-Windows, a host vulnerability checking tool specifically designed to find Windows vulnerabilities. Ferret-Windows covers a wide range of vulnerabilities, supports the addition of new plug-in modules, automates what Windows system administrators are doing manually, and is a free and open source. The structure of Ferret-Windows includes four components: the Ferret core, vulnerability checking plug-in modules, vulnerability fixing plug-in modules, and output plug-in modules. The Ferret core is responsible for checking the version of the computer's operating system, checking for dependencies, running the vulnerability checking plug-in modules, and running the output plug-in modules. Each vulnerability checking plug-in module performs one action, such as checking for one vulnerability or listing users in one specific group. These modules can be classified into five families: storage, security policies, domain settings, local settings, and passwords. About 50 different vulnerability checking plug-in modules have been developed. Several plug-in modules are included that fix vulnerabilities that vulnerability checking plug-in modules have found. Users can create a consolidated presentation of the individual plug-in modules outputs in the form of text and HTML.

Ferret-Windows was implemented using Windows Script Hosting (WSH) and more specifically VBScripts. Due to the shortcomings in VBScript's socket API in WSH, Python was used to interface with the network. Ferret-Windows is accessible at sourceforge.net (http://www.sourceforge.net/projects/ferret) where we expect the security community to use Ferret-Windows and to develop new vulnerability checking modules. Our research team will continue developing new vulnerability checking and fixing plug-in modules. In addition, we will develop new output plug-in modules including a LaTeX module.

**ACKNOWLEDGEMENT**

The authors would like to thank Andy Chou, Daniel Ilkovich, and Paige Smith for their helpful comments on previous versions of this paper.

## References

[ActivePerl] http://www.activestate.com/Products/ActivePerl/
[BSA] http://www.microsoft.com/technet/security/tools/mbsahome.mspx
[Far90] D. Farmer and E. H. Spafford, The COPS Security Checker System, in Proc. Summer Usenix Conference, Berkeley, CA, USA, pp. 165-170, 1990.
[ISS] http://www.iss.net/products_services/enterprise_protection/vulnerability_assessment/scanner_internet .php
[PWDump] http://www.bindview.com/Services/RAZOR/Utilities/Windows/pwdump2_readme.cfm
[Retina] http://www.eeye.com/html/Products/Retina/index.html
[Sha04] A. Sharma, J. R. Martin, N. Anand, M. Cukier, and W. H. Sanders, Ferret: A Host Vulnerability Checking Tool, in *Proc. 10th IEEE Pacific Rim International Symposium on Dependable Computing (PRDC-10)*, Papeete, Tahiti, French Polynesia, pp. 389-394, March 2004.
[Tiger] Tiger Analytical Research Assistant home page. http://wwwarc.com/tara/index.shtml
[Tweak] http://www.winguides.com/tweak/
[Vie02] J. Viega and G. McGraw, *Building Secure Software*,Addison-Wesley, 2002.
[Windows] "Understanding Key Architectural Differences"
http://www.microsoft.com/ntserver/ProductInfo/Comparisons/UNIX/NTappdev/2_ArchitecturalDiff.asp