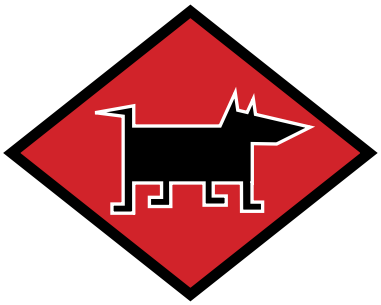
Password Defender

Ensuring Password Quality on NT Networks

A Technical Overview of Password Defender
and other defences against NT Password Cracking

Revision 90, Wednesday, April 7, 1999

Frank O'Dwyer <fod@brd.ie>



RAINBOW DIAMOND
information security

http://www.brd.ie/

info@brd.ie

The most recent version of this document is available from:
http://www.brd.ie/ntsecurity/

CONTENTS

Introduction

Password Cracking tools represent a serious threat to Windows NT networks. Tools such as l0phtcrack are capable of automatically guessing millions of password possibilities per second, running "dictionary attacks", sniffing network login attempts, and extracting passwords from NT emergency repair disks. On typical NT networks, they will recover the cleartext of many account passwords (including Administrator accounts) in less than a day (often significantly less).

This document presents the NT Password Cracking threat, discusses weaknesses in existing responses to it, and gives a technical overview of Rainbow Diamond's product, *Password Defender*.

For an overview of NT Password Cracking tools, *see How NT Password Cracking Tools Work* (page 3 of this document).

For an overview of available defences against NT Password Cracking, and their benefits and limitations, see *Defences against NT Password Cracking* (page 5 of this document).

For an overview of Password Defender's features, including screenshots, see *Password Defender's Approach* (page 9 of this document).

For up-to-date information on Password Defender, visit:

http://www.brd.ie/ntsecurity/

How NT Password Cracking Tools Work

To understand how to defend against NT password cracking tools it is necessary to know something about how they work. The following sections give a brief overview of how NT stores and protects user passwords, and how cracking tools circumvent these controls. Readers who are already familiar with how NT password cracking tools work may wish to skip to the next section (Defences against NT Password Cracking).

How NT Stores Passwords

NT systems store passwords in a portion of the registry known as the "SAM". The passwords are not stored in cleartext, but instead a function of the password is computed, and the result is stored. The functions used are known as "one-way" functions, or "hashes", which means that given the result it is not feasible to directly compute the original cleartext password. In other words, such functions are easy to compute but difficult to reverse, hence "one-way".

NT applies two one-way functions to each password, and stores the results of both in the registry. One is known as the "LANMAN" password, and is based on the DES encryption algorithm. The other is known as the "NT" password, and is based on the MD4 algorithm. The LANMAN password is not case-sensitive, so that passwords of "SECRET" and "secret" will result in the same value being stored in the registry.

How NT Protects Passwords

The security of NT passwords rests partly on the practical difficulty of retrieving the one-way function values from a system, and partly on the computational difficulty of

reversing the one-way functions themselves. NT makes it more difficult to retrieve the one-way values in the first place by two means:

- **Operating System controls**. The portion of the registry that stores the passwords (one-way function results) is normally accessible only to the operating system, and not (directly) accessible to logged in users, including Administrators.
- **"Security through obscurity"**. The portion of the registry that stores the one way function values is encrypted under a fixed or system-specific encryption key. Although this process uses strong encryption, in fact it merely obscures the stored values, since the encryption key is fixed or stored on the system in cleartext.

How Password Crackers Circumvent NT's controls

Step 1: Obtaining the one-way function values

Although the one-way function values are theoretically available only to the operating system, in fact there are a number of ways of bypassing these controls, even if the SAM is "encrypted":

- By temporarily altering the SAM permissions to allow Administrator access (if the SAM is not encrypted)
- By remotely accessing the SAM (if remote administrative access is available, and the SAM is not encrypted)
- By directly calling undocumented internal APIs (if the SAM is encrypted)
- Sniffing of network login attempts[1] (whether or not the SAM is encrypted)
- Theft/copying of emergency repair disks

A remote (unprivileged) attacker can also obtain the one-way function values if they can get an authorised administrator to run a Trojan Horse (e.g. using a Trojan ActiveX control, a Trojan email attachment, a local tool with a back door, or a similar attack). Similarly, an attacker may be able sniff network login attempts remotely by getting a network user to run a Trojan Horse.

It is also worth noting that in practice passwords tend to be reused across many systems (including network services such as web servers, and non-NT systems). This makes the attackers job considerably easier, since if one vulnerable system is discovered, the chances are high that the recovered passwords provide access to other systems too.

Step 2: Using the one-way function values to retrieve the cleartext passwords

Once an attacker has obtained one-way function values, the next step is to use them to recover the cleartext passwords. Although it is computationally infeasible to directly reverse the one-way functions, in practice it is often feasible to recover the cleartext by applying the one-way function to millions of password possibilities, and checking if the results match. There are several variants of this attack:

- **Dictionary Attack.** This approach applies the one-way function to each password in a large dictionary (e.g. of common passwords and jargon), and

[1] This does not obtain the one-way function values themselves, but yields similar material. The LANMAN network login protocol (used by Windows 95 and other clients) is especially weak.

checks if the results match. Since most user-chosen passwords are derived from dictionary words, this approach is often successful, and it is quick.

- **Brute Force.** This approach generates all possible permutations of a certain character set (e.g. alphanumeric), applies the one-way function to each, and checks if the results match. This approach is certain to succeed if the character set used is large enough, but it is CPU intensive and time consuming. Still, all possible alphanumeric passwords can be searched in about 1 day on modest hardware, and most user-chosen passwords are alphanumeric.
- **"Hybrid" Brute Force.** This approach expands a dictionary attack using a combination of heuristics and brute force to derive thousands of additional password possibilities from each dictionary word. Done well, this approach can significantly increase the success rate of a dictionary attack, and is especially effective against dictionary-derived passwords drawn from character sets that are too large to brute force.
- **Pre-computed Dictionary Attack/Brute Force.** This approach pre-computes the one-way functions for a number of password possibilities, and stores them in a database for quick lookup. This approach is superficially attractive for its speed, however the storage overhead means that it lacks the thoroughness of brute force coupled with very large dictionaries. (For example, even pre-computing only 7-character alphanumeric passwords would require storage on the order of a terabyte, and that's without counting storage requirements for cleartext passwords and database overhead.)

Defences against NT Password Cracking

Minimum Password Length

A common defence against password cracking is to enforce a minimum length for user-chosen passwords. The goal is to make it more likely that brute force runs cannot complete before the password expires. Enforcing a minimum length can be done with standard NT tools including "User Manager" (musrmgr.exe). Enforcing a minimum password length is helpful, but only partially effective since:

- Most available tools don't distinguish between privileged and unprivileged users, and require the same minimum length to be enforced for all users. This can lead to a "lowest common denominator" effect, leaving privileged accounts exposed.
- Those tools that do distinguish between privileged and unprivileged users typically do so only on the basis of NT privileges. However in practice this distinction is inappropriate, as "ordinary" NT users may have limited privileges on the box, but can still have extensive privileges on sensitive services such as databases and web servers.
- In practice passwords of *exactly* 7 or 14 characters tend to be strongest, due to weaknesses of the LANMAN one-way function. Passwords of other lengths tend to be more vulnerable to dictionary attack and/or brute force. By requiring only a *minimum* length, most available tools can't enforce this requirement.
- NT is set up to permit blank passwords by default (i.e., a minimum length of 0).

Password Ageing

Exposure to password cracking tools can also be reduced by regularly expiring passwords, a process known as "password ageing". Again, the goal is to make it more likely that password cracking runs cannot complete before the password expires. Password ageing is a standard NT feature, and can be enforced using User Manager. It is helpful in general, but of limited effectiveness because:

- Most user-chosen passwords are dictionary-derived. Dictionary attacks against passwords complete very quickly, and so the assumption that a password can be retired before it is cracked does not hold for most passwords.
- Even passwords that are not dictionary-derived may be drawn from a small character set (e.g. alphanumeric). An attacker with modest resources can brute-force many such passwords in about 1 day.
- By default, NT disables password ageing for some accounts, notably the Administrator account, which has the "Password Never Expires" setting in User Manager. (Password ageing can be enabled for these accounts, however.)
- Standard NT expiry doesn't distinguish between privileged and unprivileged users, and requires the same ageing policy to be enforced for all users. This can lead to a "lowest common denominator" effect, leaving privileged accounts exposed.
- Expiry of account passwords is ineffective if accounts fall into disuse, since the password will never be changed if the user does not log in.

Password History

Many users respond to expiry of their password by "changing" their password to exactly the same password as before. From the attacker's perspective, this extends the time available for cracking the password (effectively the password has not been expired at all).

The usual means of preventing this is to maintain a "password history", preventing users from changing to a password that they have recently used. Some users respond to this by "changing" their password quickly a number of times until the history mechanism "forgets" their password, and then reinstating their expired password. To avoid this, a minimum time (e.g. 1 day) that must elapse between password changes can also be enforced.

Password histories and minimum time between password changes are standard NT features, and can be enabled using User Manager. Some weaknesses of this approach are:

- The purpose of history mechanisms is to close down holes in the password expiry mechanism. The weaknesses of password expiry (see above) still apply.
- History mechanisms are finite, and there is a residual risk that a previously expired password will be inadvertently reinstated (long after it has been cracked by an attacker).
- By storing previously used passwords on the system, history mechanisms can exacerbate the problem of weak passwords, since an attacker can go after previously used passwords as well, and use them to attack services other than NT (we are not aware of any tools that do this, but it is in principle possible). This

makes it all the more urgent that expired passwords are completely retired from use across all network services.

Password Auditing

Password Auditing tests a system for vulnerability to password crackers by attacking it with the same tools that attackers use. Any passwords that are found to be vulnerable are then manually removed from the network (e.g. by expiring them, disabling the accounts, and/or notifying the users).

Password Auditing is an important defence, but nonetheless it suffers from many drawbacks:

- Cracking tools typically capture one-way function values, then begin auditing them. In the meantime, passwords may expire, or users may change their passwords. Thus, auditing does not necessarily reflect the current state of the system.
- Cracking tools are typically CPU intensive and time-consuming to run. As a result they are often used inconsistently, or not at all. Tools using pre-computed brute force are faster and more convenient, but not thorough—for example they cannot assure that an exhaustive brute force of alphanumeric or larger character sets will not succeed.
- Cracking tools typically require that the system being audited is open for network administration, or else the tool must be run locally on the system. This makes them unsuitable for auditing sensitive systems, standalone systems, or laptops.
- Cracking tools are oriented towards attack and not defence. Consequently, they typically require a user to be logged in while the attack is in progress, display cleartext passwords on the screen (where passers-by may see them), and may leave traces of their operation in temporary or intermediate files.
- Many organisations will be reluctant to widely distribute attack tools throughout their IT systems. The routine availability of such powerful tools in the vicinity of sensitive systems can constitute a risk in itself.
- When weak passwords are discovered by an audit, it may be too late. The passwords concerned may already have been compromised, and used by an attacker.
- There is no guarantee that weak passwords discovered by an audit will not simply be reinstated in future, whether deliberately (e.g. in case of a user's "favourite" password), or inadvertently.
- Some passwords may be suitably complex, and immune to password cracking, but may be vulnerable for another reason. For example, they may be well known "default passwords" used locally, and known to attackers. Or, perhaps they were used on another system (such as a web-server) which has been compromised. Auditing cannot uncover the use of such passwords unless they are added to its cleartext dictionary. In some environments the presence of such passwords in cleartext dictionaries may be a risk in itself, since compromised passwords remain sensitive until they are fully removed from a network.

Password Filtering

Password Filters (e.g. NT's "passfilt.dll") enforce strong passwords when a password change occurs. Typically they enforce a minimum password length, and require the

password be drawn from a large character set, but they can also enforce other requirements. The goal is to catch weak passwords *before* they are installed, and to allow only passwords that require significant brute force effort.

In conjunction with appropriate password expiry and password auditing, password filters are a highly effective defence against password cracking. However, password filters have a number of problems:

- Most available filters are primitive and will still allow dictionary-derived passwords to be set, so long as they use the required character set. These passwords are still vulnerable to a dictionary attack and/or hybrid brute force.
- Most available filters suffer from the "minimum length" problem. That is, they cannot enforce the requirement for passwords of exactly 7 or 14 characters.
- Password filters are only effective from the date they are first installed. Any existing weak passwords will not be upgraded until they expire, or are manually changed. Accounts that have fallen into disuse may never be upgraded.
- The use of password filters may result in some users choosing passwords that are difficult to remember, and writing them down. Additionally, users may be frustrated by having their password change attempts rejected, resulting in increased support calls.
- Most available filters don't distinguish between privileged and unprivileged users, and require the same policy to be enforced for all users. This can lead to a "lowest common denominator" effect, leaving privileged accounts exposed.
- Those filters that do distinguish between privileged and unprivileged users typically do so only on the basis of NT privileges. However in practice this distinction is inappropriate, as "ordinary" NT users may have limited privileges on the box, but can still have extensive privileges on sensitive services such as databases and web servers.
- Some passwords may be suitably complex, and pass filtering tests, but may be vulnerable for another reason. For example, they may be well known "default passwords" used locally, and known to attackers. Or, perhaps they were used on another system (such as a web-server) which has been compromised. Most available filters cannot prevent the reuse of such passwords.

Other Defences

For completeness, some other defences against password cracking include:

- Disabling LANMAN authentication in environments which do not require it (e.g. NT-only environments)
- Use of firewalls/packet-filters to establish NT-only enclaves
- Use of (encrypted) virtual private networks to defend against network sniffing
- Use of physical network topologies that complicate network sniffing
- Avoiding network use of privileged accounts (e.g. disabling remote administration, etc.)
- Use of hand-held authenticators, and/or one-time passwords for privileged accounts
- Use of public key based login, smart cards, etc.

Password Defender's Approach

Password Defender can be installed on standalone systems, laptops, workstations, servers, and domain controllers (providing protection for all domain logins). It uses a combination of defences including password filtering, password auditing, and automatic countermeasures. By integrating these defences in a single package, Password Defender avoids many of the drawbacks of traditional auditing and filtering approaches.

Password Defender's features are presented in more detail in the subsequent sections.

Smart Password Filtering

Password Defender's password filtering uses a combination of character class testing, dictionary tests (including tests for dictionary derived words and multi-word combinations), length testing, and heuristics to weed out poor passwords. When users attempt to set passwords that look vulnerable to dictionary attack, brute force, "hybrid" brute force, and other common password cracking techniques, the password change is rejected. In contrast, NT's "strong password" filtering and similar filters will let many such weak passwords through.

Password filtering is highly customisable, using a policy mechanism to allow individual filter tests to be enabled or disabled for specific NT user groups, or for individual accounts.

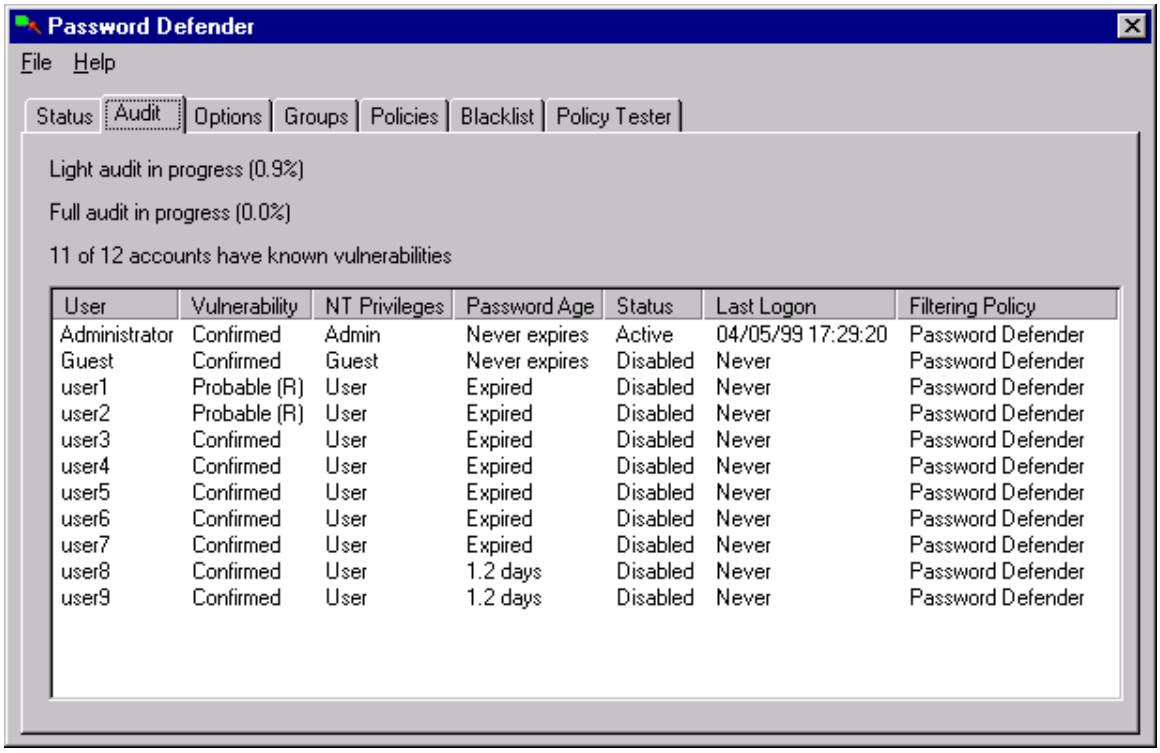
Unattended Password Auditing

Password Defender uses idle computer cycles to automatically schedule password auditing in the background, and does not require a privileged user to be logged on while the audit progresses.

Auditing employs dictionary attack, brute force, hybrid techniques, and other heuristics. The dictionary attack component uses a base UNICODE dictionary of over 1.2 million passwords, drawn from common passwords, natural language words, literature, science fiction and other sources. This basic dictionary is expanded using hybrid brute force and other heuristics, yielding a "virtual dictionary" of over 1.4 billion dictionary-derived passwords.

The brute force component verifies that passwords are not vulnerable to exhaustive alphanumeric brute force (over 80 billion passwords), and also verifies that passwords do not fall in the early range of in-depth exhaustive brute force by running a l0phtcrack-style attack with a 68-character set (alphanumeric and specials).

Auditing is designed for unnoticeable impact on computer performance, and works only when you are not using your computer. The CPU and memory are released to other programs when the computer is busy.



Integrated Filtering and Auditing

Tight integration of filtering and auditing means that audits can be intelligently scheduled to run only when passwords are changed, and cracked passwords can be automatically and immediately added to the filter list, preventing their future reuse. Also, this integration means that if a password expires or is changed while an audit is in progress, auditing works with the most up-to-date passwords, and reflects the current state of the system.

Defence-oriented Design

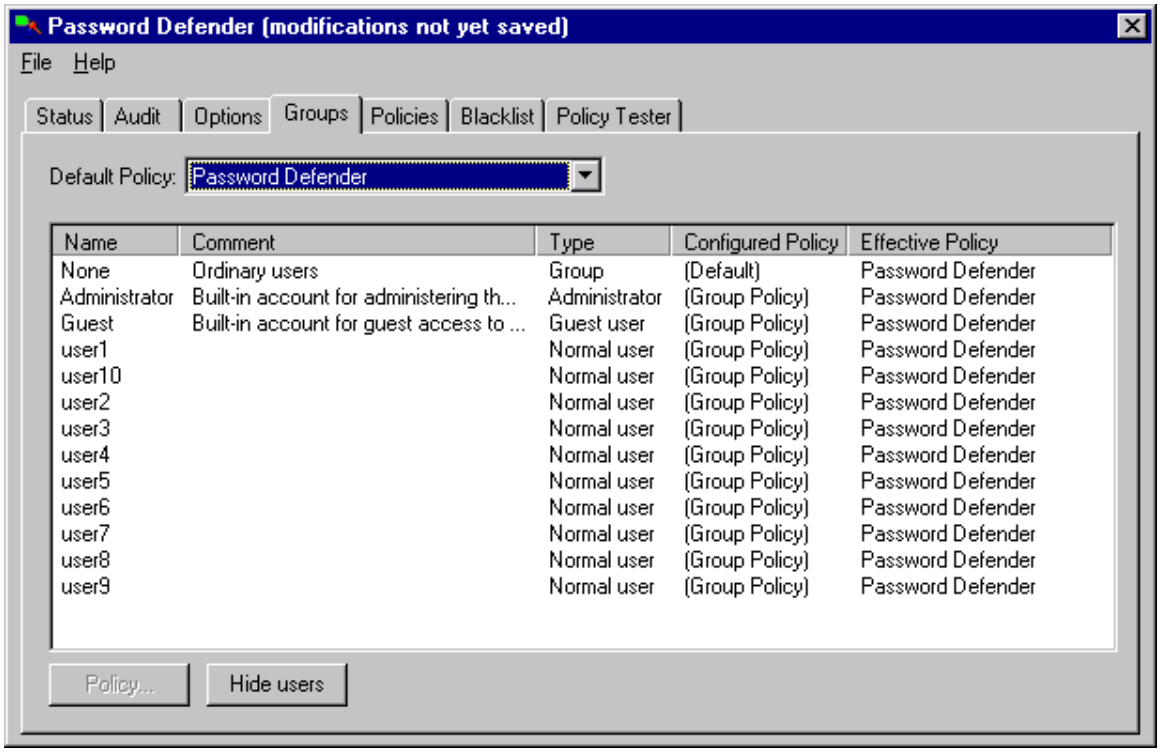
Password Defender is explicitly defence-oriented, in contrast to alternative password auditing and policy enforcement tools, which are primarily attack tools. Password Defender does not require a privileged user to be logged on, does not display the cleartext of recovered passwords on the screen, and does not leave traces of cleartext passwords in temporary or other intermediate files.

To achieve this, Password Defender operates with one-way functions of passwords at all times, uses no temporary files, and never stores or displays cleartext passwords. When Password Defender discovers a vulnerable password, it destroys the cleartext, notes the vulnerability, exercises any configured countermeasures, adds a one-way function of the vulnerable password to the filter list (if configured to do so), and moves on.

Per-user and Per-group Password Strength Policies

Password Defender allows for per-user and per-group policy settings. This avoids the "lowest common denominator" effect, and allows (e.g.) more stringent policies to be set for NT administrators, high risk groups, or administrators of sensitive services such as databases or web servers.

Password Defender also comes with a number of predefined policies, designed to provide maximum resistance to tools such as l0phtcrack.

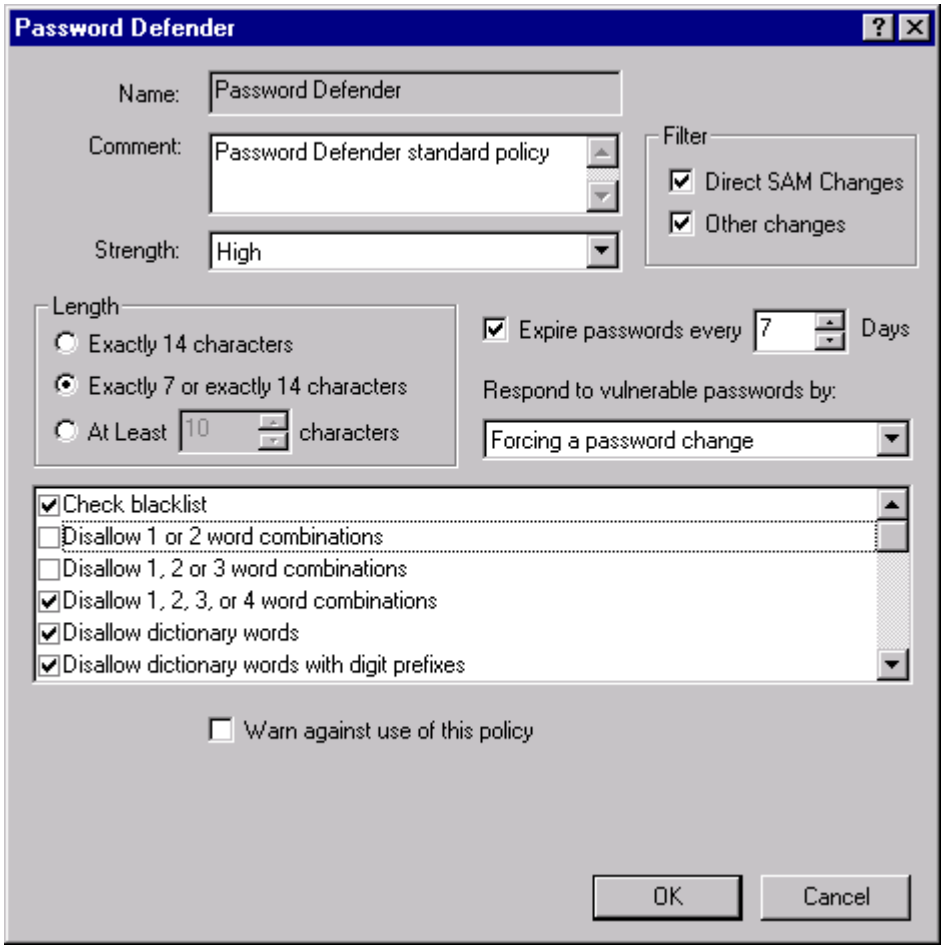


Custom Filtering Policies

Password Defender's filtering component is highly customisable. A Password Defender policy includes the following filtering details:

- Details of filtering tests to be applied when a password change is attempted. A custom policy can be built from a wide range of character class tests, dictionary tests, and other heuristics.
- Length requirements (including the ability to specify a requirement for passwords exactly 7 or 14 characters in length)
- Password Ageing details (allowing different expiry requirements to be set for different groups or accounts)
- Auditing countermeasures (allowing different responses to be set for different groups or accounts)
- A strength estimate, used to resolve conflicts when a user is a member of multiple groups (Password Defender uses the strongest policy in such cases)
- Whether or not to apply filtering to direct SAM changes, and/or network password changes.

An existing password filter DLL, if you have one, can also be used as part of a Password Defender policy. This allows you to continue using your existing DLL if you need to, but with the benefits of Password Defender's policy mechanism. This means that your existing tests can be applied to some users and groups, and not others.



Automatic Countermeasures

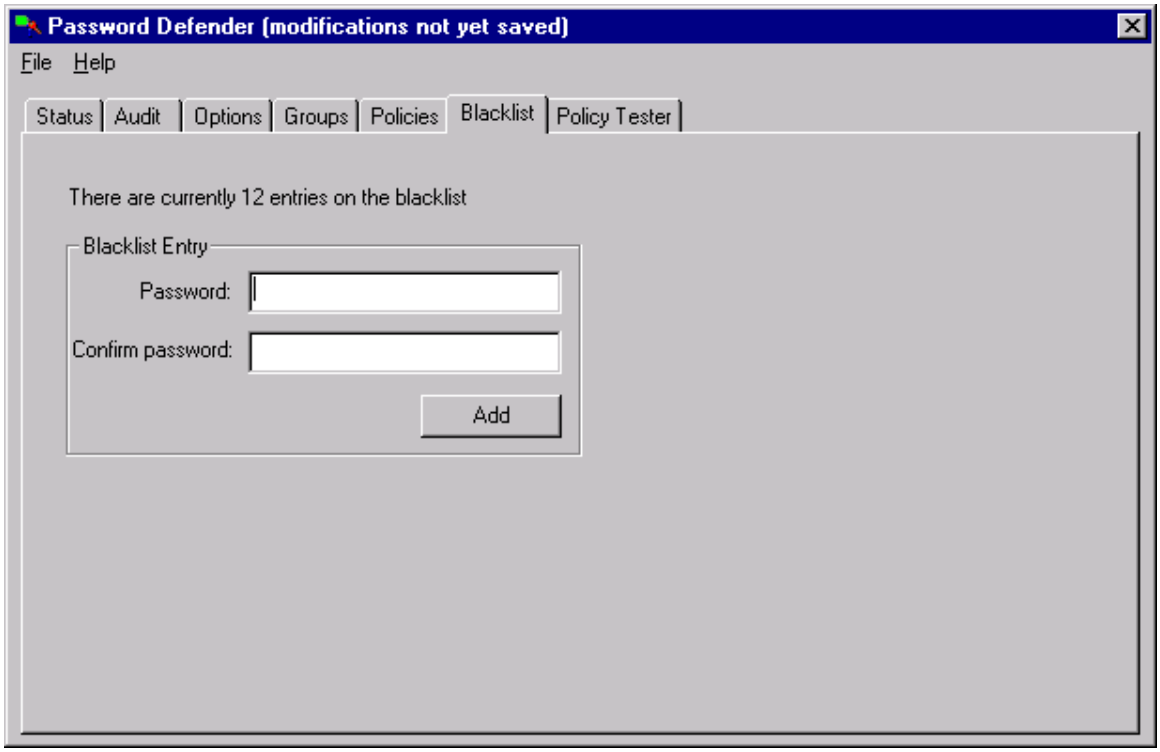
When auditing detects a weak password, it can automatically respond by forcing a password change, or disabling the account. By forcing a password change, Password Defender accelerates the removal of weak passwords from the system and brings password filtering into play as quickly as possible. Tight integration of filtering and auditing means that when the password change occurs, the vulnerable password can be excluded from reuse, and the new password will be subject to testing to ensure that it is not also vulnerable.

Password Blacklisting

Some passwords may be suitably complex, and not vulnerable to password cracking, however they may be compromised for some other reason. For example, many sites fall into the habit of using well-known default passwords on systems, and these may become well-known to attackers. Or, a password may have been used on another service (such as a web-server) which has been compromised. Password Defender allows such passwords to be explicitly blacklisted, so that filtering will prevent their future reuse.

Since passwords on the blacklist are potentially sensitive, Password Defender does not store them in cleartext, and instead uses a strong one-way function (SHA1) together with a cryptographically random salt value and an iteration count. This

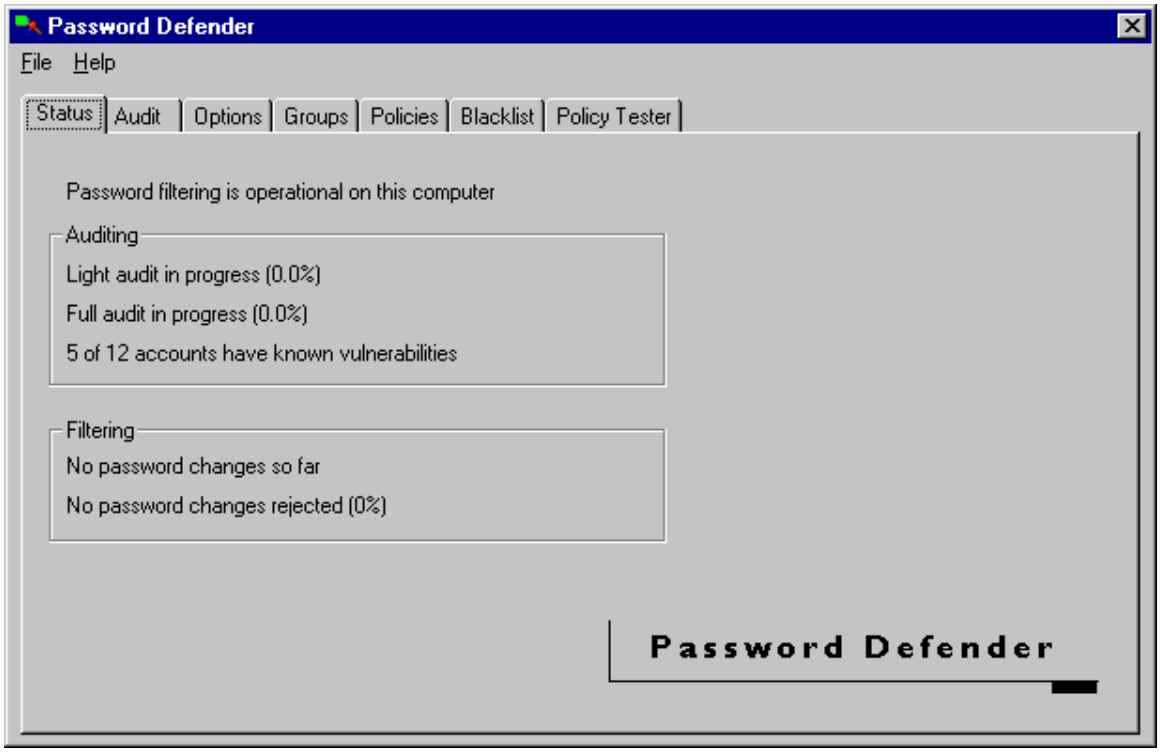
construction is designed to ensure that it is infeasible to brute force the blacklist itself, should it be stolen from a system.



Real-time Status Monitoring

Password Defender provides real-time updates on the status of both filtering and auditing, including:

- Number of password changes
- Percentage of password changes rejected by filtering (allowing you to gauge if your filtering policy is causing users excessive difficulty in changing their password)
- Number of vulnerable accounts
- Password ageing details for vulnerable accounts (whether the password is expired, how long the password has been in use, whether or not the password can be automatically expired)
- Account status for vulnerable accounts (active/disabled)
- Last login time for vulnerable accounts (allowing you to gauge whether or not vulnerable accounts have fallen into disuse)
- Filtering policy for vulnerable accounts (allowing you to gauge whether or not filtering requirements need to be changed)



Integrated Policy Testing

Password defender provides the ability to test predefined and custom filtering policies by allowing you to try out various passwords, and find out which policies permit them to be set and which reject them.