Local Access Can Nullify The Strongest Passwords by **Jason Mansfield** on **18/06/04** ←back

Contents

Everyone in the IT field knows or should know that passwords are almost always the weakest link in the security chain, but the focus is almost always on choosing passwords and policies that enforce good choices. In this article I'll outline how a malicious user with physical access to a computer can, within time, gain any or all passwords for that machine.

Password Primer

Those familiar with how passwords are stored and authenticated can skip this section; everyone else should read on as this is one of the most important concepts in computer security.

A password is sequence of characters (letters, numbers, punctuation, etc) intended to prove that a user should have access to a resource. This is security through a 'secret': as long as only you know the secret and no one else can guess it, that resource is hypothetically secure. An authentication system accepts the supplied password and compares it to some stored version to determine whether or not the user is authorized to access the resource.

Intelligently created authentication systems do not simply store the password but encrypt or mangle it in some way so that someone with access to the stored passwords can't read them. This not only discourages attackers, it prevents malicious administrators from seeing the password and posing as the user. Most authentication systems mangle the password in a process called Hashing. Hashing is generally the process of taking something and sending it through a process that is extremely difficult to reverse (try assembly a potato from hash browns). There are many different hashing algorithms (hashing processes) and their explanations are beyond the scope of this article. When a user's password is changed, the supplied password is hashed and the product, the hash, is stored to a file or database. Ideally, the process cannot be reversed to obtain the original password. For authentication, the user supplies the attempted password, this is hashed, and the attempted hash is compared to the stored hash to see if they match. If they match, the attempted password must match the original and authentication is successful. A single mismatched bit results in failed authentication.

To break a password, a computer program must take what is suspected to be a password, hash it with the same hashing algorithm, and compare the hash it generated to the stored hash. If they don't match, a new suspected password is tried. This can go across billions of password and tie up even a fast computer for a great deal of time. There are shortcuts, but the overall difficulty for the attacker depends on how hard it is to guess a password. The first and most important line of defense in password security is choosing hard to guess passwords.

Obtaining Hashes With a LiveCD

If you want to have a secure system you have to assume the perfect security is impossible and you can only get so close for dollars and man power available. With this in mind, every system has a vulnerability including your passwords. I have little income but can, for free, assemble a 70 Ghz cluster for about 12 hours a week or more. Truly resourceful attackers will get more and for more time. A truly determined attacker will break your hashes.

But first the attacker must gain access to your hashes. The most profitable for the attacker is to get the hashes locally, capturing all for the machine at once. It's as simple as having a Linux LiveCD such as Knoppix. The attacker need only boot the CD, mount your hard drive partitions, and copy your SAM, shadow, or other password file to a USB flash drive or other computer on the Internet. NTFS is no longer an obstacle as NTFS read support is solid with write support improving.

Once a SAM file is obtained, the hashes can be retrieved with samdump. This doesn't apply the same way to Windows 2000 and Windows XP.

Obtaining Hashes With pwdump2

The above methods will give you password hashes on Windows 2000 and XP, but those hashes will be invalid. On top of that, there's not really a good way to tell if the hashes you get from samdump are correct without trying to crack a known password. Pwdump2, however, will get the proper hashes, but only through Windows and the user running it has to have the privilege SeDebugPrivilege, which is normally only allotted for administrators.

If the attacker doesn't mind not having the normal administrator password, there are utilities on the net which will boot from a floppy and set the administrator password as desired. Then the attacker can log in as administrator to get the hashes with pwdump2. If it doesn't exist already, there could easily be a tool to add a new user with SeDebugPrivilege so that new user can get the hashes with pwdump2.

Keeping Your Hashes Safe(r)

The simple fix for these issues is solid premise security where users report anyone they don't recognize. This is infeasible in a large corporation. Unfortunately, many employees are resistant to learn and employ good security practices, and one user who doesn't participate in security makes for an unpredictable security hole.

Booting the machine to something other than the primary operating system installation is the primary focus of this vulnerability. This can easily be remedied by either not allowing users to boot from anything but the primary operating system and password protecting the boot options. A drastic solution would be to remove, disconnect, or disable all devices that allow booting, such as CD drives, floppy drives, and USB connections. Passwords in the BIOS transfer the difficulty to opening the machine to reset the BIOS password. Not only is this suspicious, it's easily preventable with computer locks that are inexpensive and should fit most modern computer cases.

Another option is to use an encrypted file system that requires a password for the machine to boot. This is usually impractical and reserved for more paranoid users and administrators.

Conclusion

Software security is almost always inferior to hardware security. Where there is software, there will be a vulnerability. Better security is not only protecting against known vulnerabilities but anticipating and preventing vulnerabilities wherever possible.

About the author: Jason is a computer enthusiast interested in programming, networking, and general exploration. He is currently working on his BS in Computer Science in San Diego. He can be found on the freenode IRC network as crunge.