Foundations of Network and Computer Security

John Black

Lecture #22
Nov 11th 2004

CSCI 6268/TLEN 5831, Fall 2004

Announcements

- Proj #2 – Due week from today

- Following Thurs is Thanksgiving – No Class
- Following Tuesday is just Tuesday – No Class

- Final is about a month from now (time flies)

- Brian discovered a typo on shellcode slides: load addr of “/bin/sh” into %ebx and addr of array into %ecx (I had it the other way around)

Password Crackers

- Unix approach: store one-way hash of password in a public file
  - Since hash is one-way, there is no risk in showing the digest, right?
  - This assumes there are enough inputs to make exhaustive search impossible (recall IP example from the midterm)
  - There are enough 10-char passwords, but they are NOT equally likely to be used
    - HelloThere is more likely than H7%$$a3#.4 because we're human

Password Crackers (cont)

- Idea is simple: try hashing all common words and scan for matching digest
  - Original Unix algorithm for hash is to iterate DES 25 times using the password to derive the DES key
    - $DES^{25}(pass, 0^{64})$ = digest
    - Note: this was proved secure by noticing that this is the CBCMAC of $(0^{64})^{25}$ under key ‘pass’ and then appealing to known CBCMAC results
    - Why is DES iterated so many times?

Password Crackers (cont)

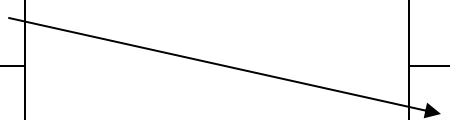
- Note: Actually uses a variant of DES to defeat hardware-based approaches
- Note: Modern implementations often use md5 instead of this DES-based hash
- But we can still launch a ‘dictionary attack’
  - Take large list of words, names, birthdays, and variants and hash them
  - If your password is in this list, it will be cracked

Password Crackers: example

word	*digest*
alabaster	xf5yh@ae1
albacore	&trh23Gfhad
alkaline	Hj68aan4%41
wont4get	7%^^1j2labdGH

Pasword file
/etc/passwd

jones:72hadGKHHA%
smith:HWjh234h*@!!j!
jackl:UwuhWuhf12132^
taylor:Hj68aan4%41
bradt:&sdf29jhabdjajK22
knuth:ih*22882h*F@*8haa
wirth:8w92h28fh*(Hh98H
rivest:&shsdg&&hsgDGH2



Making Things Harder: Salt

- In reality, Unix systems always add a two-character “salt” before hashing your password
  - There are 4096 possible salts
  - One is randomly chosen, appended to your password, then the whole thing is hashed
  - Password file contains the digest and the salt (in the clear)
  - This prevents attacking all passwords in /etc/passwd in parallel

Password Crackers: with Salt

Table for Salt Value: ***A6***

word	*digest*
alabaster	xf5yh@ae1
albacore	&trh23Gfhad
alkaline	U8&@H**12
wont4get	7%^^1j2labdGH

no match

Pasword file
/etc/passwd

jones:72hadGKHHA%**H7**
smith:HWjh234h*@!!j!**YY**
jackl:UwuhWuhf12132^**a$**
taylor:Hj68aan4%41**y$**
bradt:&sdf29jhabdjajK22**Ja**
knuth:ih*22882h*F@*8haa**U%**
wirth:8w92h28fh*(Hh98H**1&**
rivest:&shsdg&&hsgDGH2***1**

Fighting the Salt: 4096 Tables

- Crackers build 4096 tables, one for each salt value
  - Build massive databases, on-line, for each salt
    - 100's of GB was a lot of storage a few years ago, but not any longer!
    - Indexed for fast look-up
    - Most any common password is found quickly by such a program
    - Used by miscreants, but also by sysadmins to find weak passwords on their system

Getting the /etc/passwd File

- Public file, but only if you have an acct
  - There have been tricks for remotely fetching the /etc/passwd file using ftp and other vulnerabilities
  - Often this is all an attacker is after
    - Very likely to find weak passwords and get on the machine
  - Of course if you are a *local* user, no problem
  - Removing the /etc/passwd from global view creates too many problems

Shadowed Passwords

- One common approach is to put just the password digests into /etc/shadow
  - /etc/passwd still has username, userid, groupid, home dir, shell, etc., but the digests are missing
  - /etc/shadow has only the username and digests (and a couple of other things)
  - /etc/shadow is readable and writeable for root only
    - Makes it a bit harder to get a hold of
    - Breaks some software (including the buggy web server) which wants to authenticate users with their passwords
      - One might argue that non-root software shouldn't be asking for user passwords anyhow

Last Example: Ingres Authorization Strings

- Ingres, 1990
  - 2nd largest database company behind Oracle
- Authorization Strings
  - Encoded what products and privileges the user had purchased
    - Easier to maintain this way: ship entire product
    - Easier to sell upgrades: just change the string
- Documentation guys
  - Needed an example auth string for the manual

Moral

- There's no defending against stupidity
- Social engineering is almost always the easiest way to break in
  - Doesn't work on savvy types or sys admins, but VERY effective on the common user
  - I can almost guarantee I could get the password of most CU students easily
    - "Hi this is Jack Stevens from ITS and we need to change your password for security reasons; can you give me your current password?"

Social Engineering: Phishing

- Sending authentic looking email saying "need you to confirm your PayPal account information"
  - Email looks authentic
  - URL is often disguised
  - Rolling over the link might even pop-up a valid URL in a yellow box!
  - Clicking takes you to attacker's site, however
    - This site wants your login info

Disguising URLs

- URI spec
  - Anything@http://www.colorado.edu is supposed to send you to www.colorado.edu
    - Can be used to disguise a URL:
      - ***http://www.ebay.com-SECURITYCHECKw8grHGAkdj>jd7788<Account Maintenace-4957725-s5982ut-aw-ebayconfirm-secure-23985225howf8shfMHHlUBd889yK@www.evil.org***
    - Notice feel-good words
    - Length of URI exceeds width of browser, so you may not see the end
    - www.evil.org could be hex encoded for more deception
      - %77%77%77%2e%65%76%69%6c%2e%63%6f%6d = www.evil.com

Disguising URL's (cont)

- This no longer works on IE
- Still works on Mozilla
- In IE 5.x and older, there was another trick where you could get the toolbar *and* URL window to show "www.paypal.com" even though you had been sent to a different site
  - Very scary
- Moral: don't click on email links; type in URL manually

Digression: Character Encodings

- Normally web servers don't allow things like this:
  - http://www.cs.colorado.edu/~jrblack/../../etc/passwd
    - The ".." is filtered out
  - Character encodings can sometimes bypass the filter
    - Unicode is a 16-bit code for representing various alphabets
    - . = C0 AE
    - / = C0 AF
    - \ = C1 9C
  - In Oct 2000, a hacker revealed that IIS failed to filter these encodings
    - …/~jrblack/%C0AE/%C0AE/%C0AE/%C0AE/etc/passwd

Segue to Web Security

- The web started out simple, but now is vastly complex
  - Mostly because of client-side scripting
    - Javascript, Java applets, Flash, Shockwave, VBScript, and more
  - And server-side scripting
    - CGIs (sh, C, perl, python, almost anything), Java servlets, PHP, ASP, JSP
  - All of these are security-sensitive
    - Ugh

We Can't Hope to Survey all Possible Web Security Issues

- Too many to count
- Goal: look at a few thematic ones
- Cataloguing all of them would not be very instructive, most likely

Typical Server-Side Vulnerability

- PHP: Personal HomePage (?!)
  - An easy-to-use and Perl-like server-side scripting language
  - A “study in poor security” – Gary McGraw
  - Variables are dynamically declared, initialized, and global by default; this can be dangerous:
    - ```
      if(isset($page))
      {
        include($page);
      }
      ```
    - Now we call this script with:
      - `script.php?page=/etc/passwd`
      ```

Javascript

- Javascript (and VBScript) can do bad things
  - Get your cookies, for one, which may include sensitive information
- You don’t want to run scripts unless the site you are visiting is “trustworthy”
  - Javascript has had a large number of security problems in the past; dubious sites might take advantage of these
  - If you set your security level high in IE, it turns off Javascript; that should tell you something

Javascript (cont)

- Turning it off in your browser is one solution
  - But often you lose a bunch of functionality
- How can an attacker get you to run his malicious Javascript code?
  - Gets you to visit his website
    - But you might know better
  - Old trick: post something to a bulletin board with <script>…</script> in it
  - When you view his post, you run his script!

Filtering

- To prevent this, a correct bulletin board implementation always filters stuff that others have posted
- You can post YES! but not <script> evil stuff… </script>
- But until recently we didn't worry about filtering stuff from you to yourself! ☺

XSS Attacks

- XSS: Cross Server Scripting
  - Not CSS (Cascading Style Sheets)
  - Idea: you open a website, passing a value, and the site echoes back that value
    - What if that value has a script embedded?!
  - Example: 404 Not Found
    - Suppose you see a link (in email, on IRC, on the web) saying, “Click here to search Google”
      - The link really does go to google, so what the heck…
      - However the link is www.google.com/badurl%0a%5C...
        - » Above contains an embedded, hidden script
      - Google says, “badurl%0a%5C…” not found
      - Just displaying this to you, executes the script

XSS Vulnerabilities

- They've been found all over the web
  - Fairly new problem
  - Lots of examples still exist in the wild
  - Very tricky to find them all
- Solution is to filter, of course
  - Need to filter inputs from users that server will be echoing back to the user