Overview

- First widely reported exploit.
- Technical details of a web exploit.
- Importance of validating data; heartbleed bug.

Morris Worm

- 1988, first hack which received attention in the popular press.
- Replicating worm.
- Overflow'd `gets()` buffer in `fingerd` daemon.
- Misused `DEBUG` command to `sendmail` daemon.
- Ran dictionary attack against publicly readable `/etc/passwd` file.
- Given a password, would use `.forward` and `.rhosts` file to break into other hosts where user had accounts.
- Was supposedly meant for innocent research, but a bug caused indiscriminate propagation.

Morris worm *wikipedia link*

HB Gary Hack Background

- WikiLeaks gained world-wide prominence in 2010 with releasing, among other dumps, US State Department emails in Nov 2010.
- At end of 2010, bowing to political pressure, major payment processors stopped processing donations to WikiLeaks.
- The *hacktivist* group **Anonymous** mounted *distributed-denial-of-service* attacks on web sites of payment processors.
- At beginning of 2011, CEO Aaron Barr of security company **HBGary Federal** publicized the fact that he could reveal the identity of Anonymous.
- Anonymous took down the HBGary Federal website, extraced 40K emails from email server, deleted 1TB of backup data.

Links: *New York Times*, *Ars Technica Story*, sequel and follow-up.

Details of Break-In

- HBGary Federal website was running a proprietary *content-management system* which has an *SQL injection flaw*. By providing specially crafted inputs, attackers were able to run arbitrary queries against database, and accessed login and password table.
- Passwords were hashed using a fast hash algorithm (MD5) without any *salt*, making them amenable to a *rainbow-table attack* (comparing the hashed password with a table of precomputed hashes for passwords).
- CEO and COO had passwords which were cracked. Same passwords were used on other machines and email, twitter, etc.
- Used COO password to access Linux support machine which was accessed using password-based ssh (rather than public/private key access).
- Linux system had not been patched and contained a well-known security flaw which allowed access as root! Deleted backups and research data.

Details of Break-In Continued

- Barr's password allowed access to Google Apps as **administrator**. Allowed attackers to examine email archives.
- Email contained root password for another machine `rootkit`!. But remote root access not possible.
- With access to email account, anonymous sent an email to system administrator requesting ssh access be allowed (gave possible root passwords in email!). Administrator set things up, changed password to `changeme123` and verified user-name!!
- Attackers dumped user database for all users who ever registered on `rootkit.com`. Contained crackable passwords.

Sequel

- Extremely poor security practices for a commercial company and even worse for a security company.
- Anonymous members arrested 2012 (ringleader was arrested in 2011 and was cooperating with the FBI).
- HBGary received additional business!!
- HBGary purchased by defense contractor ManTech.
- CEO Barr stepped down from HBGary. Took position as *cybersecurity director* at another federal contractor. Fired 2012 for concentrating on social media and Anonymous.
- In 2013, list of `rootkit.com` user-names/passwords allowed possible identification of Chinese hackers suspected in other hacking incidents.

The Security Problem

- A user (person or program) with limited authorization interacts with a program which has or may (temporarily) get different authorization.
- By choosing certain inputs or interacting with the program in a certain way, the user forces the program to take an unintended action using the program's authorization.
- Ultimate exploit is to spawn a shell with program's authorization.

Terminology

Some of these definitions are adapted from the *Jargon Dictionary*:

Hacker Person who enjoys exploring the intricacies of programmable systems, skilled in programming (quickly), expert.

Cracker Person who breaks security on a computer system. Journalists often misuse the term *hacker* to refer to a *cracker*.

Root Kit A collection of scripts which allows a cracker to break into a machine and (maybe) get `root` access.

Script Kiddie A cracker who merely uses *root kits* and other pre-written scripts to crack into a system. Does not really have much of an understanding of the technology.

Types of Malicious Programs

Trojan Horse A program which provides normal useful functionality while also performing hidden malicious activity. Eg. a `login` program which allows users to login normally, while storing plaintext passwords in a file.

Worm A program that propagates itself over a network, reproducing itself as it goes.

Virus A program that searches out other programs and *infects* them by embedding a copy of itself in them, so that they become Trojan horses. When these programs are executed, the embedded virus is executed too, thus propagating the *infection*. This normally happens invisibly to the user. Unlike a worm, a virus cannot infect other computers without assistance. It is propagated by vectors such as humans trading programs with their friends.

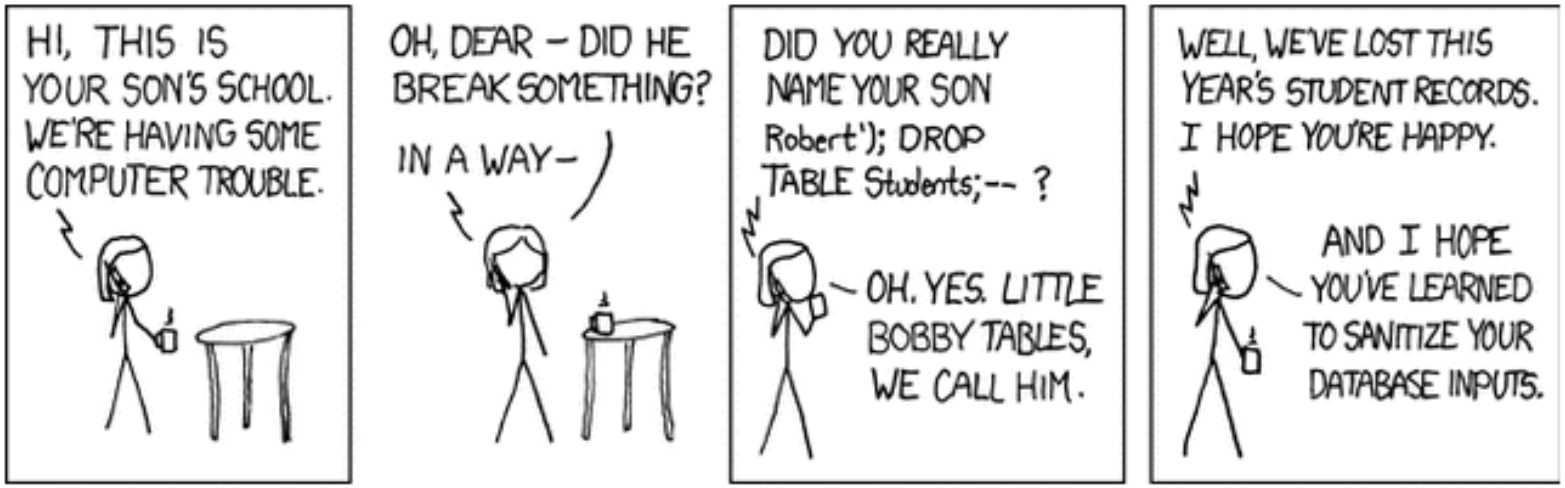
Principle of Least Privilege

- Always use lowest possible privilege.
- When running programs using a privileged program, use privileged user only for portions of execution where it is strictly necessary, reverting to non-privileged user wherever possible.
- When done with privileged operations, revert to non-privileged user permanently.
- Typically use a user with very limited privileges to run externally accessed programs like web servers. Typical user for Unix web servers is `nobody`.

Program Bugs Which Affect Security

- Buffer overflows. No checking on whether a buffer overflows ... hence specially crafted inputs can overflow the buffer overwriting other memory, which can compromise security. Notorious in C/C++ programs. Can be avoided by using safe programming languages like JavaScript, Java or Perl.
- Untrusted data used in executing other programs.
- Environmental variables.
- Race conditions. Program checks something and then acts based on the result of the check. The check and act are not atomic, allowing a malicious user to change the result of the check before the act.
- Randomness. Cryptographic schemes often depend on a source of randomness. If the source of randomness is compromised, then the security of the cryptographic scheme is also compromised.
- Denial of service.

Validating Data



(Source: http://xkcd.com/327/)

Not Validating Data

This page allows attacker to run arbitrary JavaScript on page.

- Input not validated; hence attacker can input arbitrary HTML, including scripts.
- Consider what would happen if this "name" was stored in a db and then output on the page of another user, say Mary with a request: like "john would like to chat with you".
- Script would have access to Mary's session.

Web Programming Errors

- Cross-site scripting (XSS).
- Code injection (SQL, JavaScript, PHP, etc).
- Force unintended execution of code.
- Unintended memory access: buffer overflows or other methods.
- Reveal site info: data, paths.

Heartbleed Bug

- Bug in widely used OpenSSL cryptography library; 2012-2014.
- Normal operation: To check if remote end is alive, a computer is supposed to send a *heartbeat message* consisting of a message payload along with the length of the message. Remote end is supposed to echo back message.
- Attacker sent malformed hearbeat request containing a small message with large length. Validation did not verify that length matched message length. Remote end allocated buffer for incorrect length, filled only initial portion with message and returned full buffer; basically allowing attacker to access secrets which may have been previously stored in the memory used by the unused part of buffer.
- Unintended memory access caused by insufficient data validation.

Cross-Site Scripting

1. A script from an attacker domain is injected into the scripts for the subject domain being attacked.
2. Injected script is executed as though it originated from the subject domain.

Security Scenarios Persona

Following names are often used to describe actors in security scenarios:

Alice	Wants to originate a communication.
Bob	Is the person Alice wants to communicate with.
Mallory	Is a malicious actor.
Eve	Is a eavesdropper.

Non-Persistent XSS Attack

Does not involve server.

1. Alice often logs into Bob's web site.
2. Mallory uses a XSS vulnerability on Bob's site to send an email from Bob's web site to Alice with a *cute cat* link containing a malicious script.
3. Alice clicks on the link and is redirected to Bob's web site and unintentionally executes the malicious script using her login credentials for Bob's web site.
4. The malicious script could take an action like capturing Alice's authentication cookie (if any).

Persistent XSS Attack

Involves persistent storage on server.

1. Mallory creates an account on Bob's server.
2. Mallory realizes that Bob's web site allows comments to contain arbitrary HTML including `<script>` tags. She adds a comment containing a malicious script.
3. When Alice views the comment she unintentionally runs the (normally invisible) malicious script.

Mitigating XSS

1. Always validate all input.
2. Escape any untrusted input (using escaping mechanisms dependent on the context).
3. Make sensitive cookies `HttpOnly`.
4. Do not send sensitive information like passwords or credit card numbers to browser.

RegEx Denial of Service

Wikipedia

```
> function time(fn) {
    t0 = Date.now(); fn(); return Date.now() - t0;
  }
> re = /^(a|aa)+$/
/^(a|aa)+$/
> time(() => ('a'.repeat(30) + 'c').match(re))
18
> time(() => ('a'.repeat(40) + 'c').match(re))
1835
> time(() => ('a'.repeat(46) + 'c').match(re))
32408
>
```

Hashing

- A hash algorithm `H` takes `content` (usually variable length) and produces a hash `h` (usually fixed length).

```
h = H(content)
```

- Minor changes in `content` will produce large changes in `h`.
- Given a `h`, it is very difficult to compute a corresponding `content`.
- Example hash algorithms are MD5 (not secure), SHA-256.
- A password `p` should never be stored in the clear, only `H(p)` should be stored.

Encryption

Plaintext is **encrypted** to **ciphertext**; **ciphertext** is **decrypted** back to **plaintext**.

Symmetric encryption Same secret key used for both encryption and decryption.

Asymmetric encryption separate private key (secret) and public key (not secret). Difficult to compute private key from public key (keys constructed using algorithms which are asymmetric in their computational requirements like multiplying primes versus factoring into primes).

- Message encrypted by public key can be decrypted using private key.
- Also can encode message with private key can be decoded using public key (used for signatures).

HTTPS

Depends on both asymmetric encryption (relatively non-performant) as well as symmetric encryption.

1. Browser visits `https` server.
2. Server presents certificate containing public key.
3. Browser validates certificate using a **certificate authority** like GoDaddy and Verisign.
4. Browser and server cooperate to set up a shared symmetric session key.
5. Browser and server use session key to exchange data securely.

References

Wikipedia article on the *Morris Worm*.

John Viega, Gary McGraw, *Building Secure Software*, Addison-Wesley, 2002.

Peter Bright, *Anonymous speaks: the inside story of the HBGary hack*, Ars-Technica, strongly recommended. Link.

Peter Bright, *With arrests, HBGary hack saga finally ends*, Ars-Technica. Link.

Nate Anderson, *How Anonymous accidentally helped expose two Chinese hackers*, Ars-Technica. Link.

Main-stream press coverage of HBGary hack in the New York Times. Eric Lipton and Charles Savage, *Hackers Reveal Offers to Spy on Corporate Rivals*, New York Times, Feb 11, 2011. Link.

Humorous take on HBGary hack Colbert Report Link.