Secure System Design: E-Commerce Web Site

Secure System Design

- Let's design a secure system
- (Actually, this is the first of three)
- Today: an e-commerce site—a *large* e-commerce site called TheLargeRiver.com

Entities

- Users (customers)
- Suppliers
- Shippers (FedEx, UPS, etc.)
- Credit card processors
- And. . .
- The IT department, including the NOC (Network Operations Center)
- The rest of the corporate net

Assets at Risk

- Credit card information
- Mechandise stock—it can’t be stolen
- Pricing data—sell stuff for the right price
- Database integrity

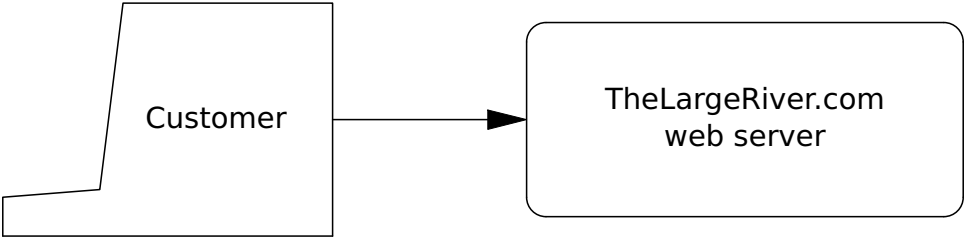
Attackers

- Ordinary hackers
- Thieves
- Probably *not* intelligence agencies

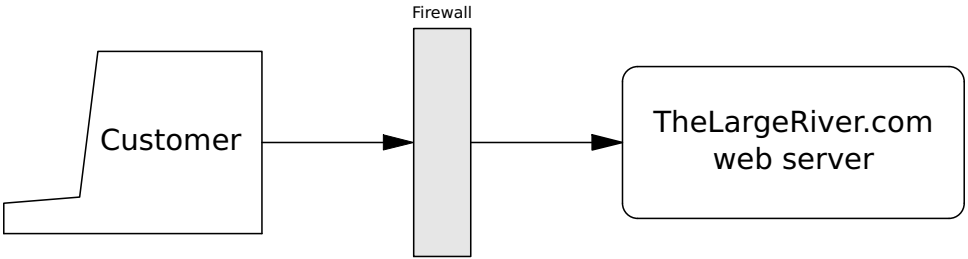
It’s Complex

- We can’t cover all of it
- We can’t even have one diagram showing the pieces we will discuss
- There are many things I won’t cover

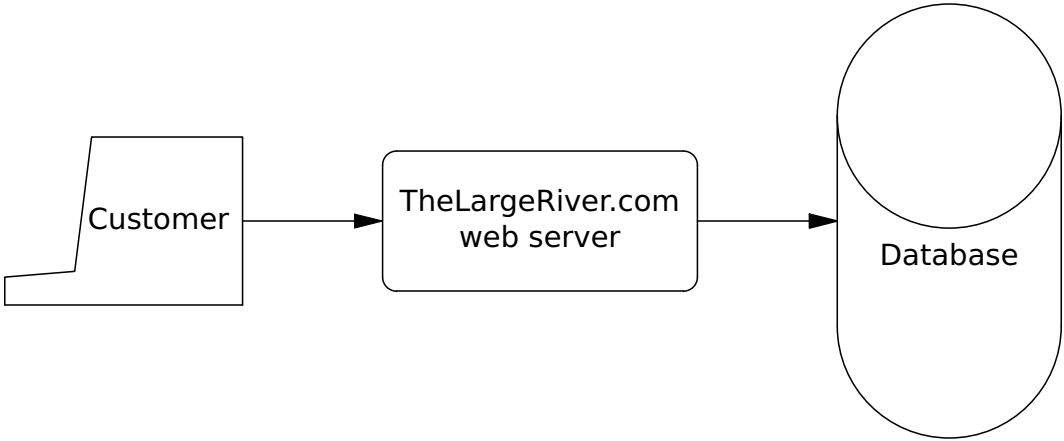
Not TheLargeRiver.com. . .



Not Any Better



Somewhat Better

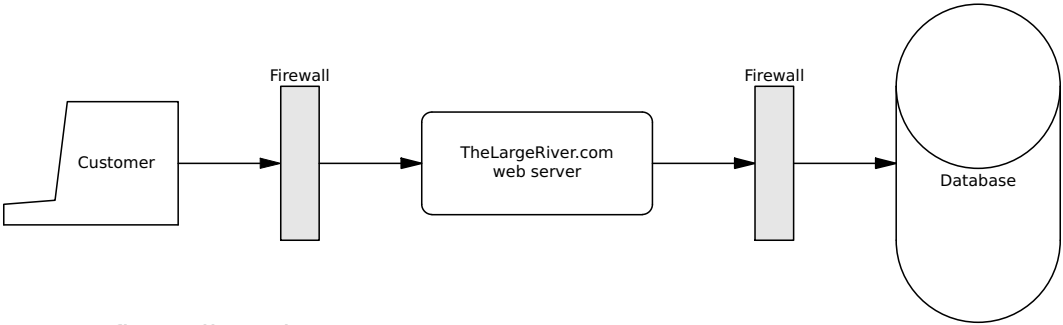


This system is database-driven. (And why have I deleted the firewall?)

The Purpose of a Firewall

- Block the potentially bad traffic
- Allow in the necessary traffic
- The web server's biggest attack risk is the web traffic itself—but we have to let that through

More Realistic



Note: *two firewalls*—why?

This is the Wrong Approach!

- We're sprinkling firewalls around without thinking about why
- What is the threat?
- What is the attacker's execution environment?
- How does a firewall help?
- What else should we be doing?

“The” Database

- Why a database?
- Actually, there are several independent tables—do they all go in one place?
  - Customer profiles
  - Inventory
  - More?

Who Reads, Who Writes?

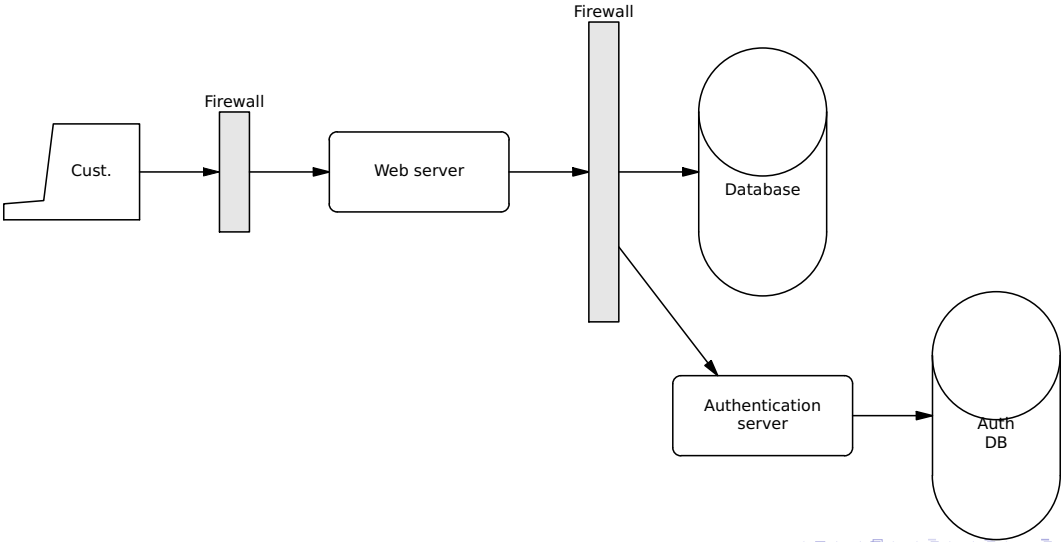
Customer Profile	Used for authentication during login; used to record orders; used for credit card billing; used for order status
Inventory	Updated when orders are placed, and when new stock arrives

These uses are rather different—and there are different security consequences

Customer Passwords

- Passwords are written when they're changed
- They never need to be read by the web server; its need is *verification*
- Password compromise is serious—people reuse passwords
- Even if they don't, compromise of a user's e-commerce password could permit orders billed to that person's credit card
- We need to protect passwords better—and remember that web servers are risky
- ☞ We need to ensure that the password database is not in the web server's execution environment

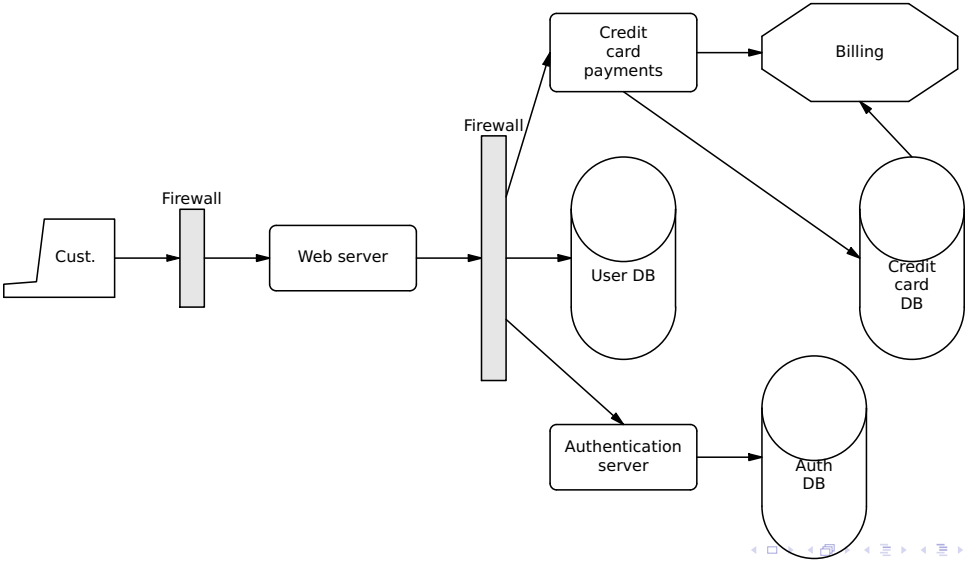
Authentication Server



Separation

- We've moved more-sensitive data to a separate database, accessible only via a less-vulnerable server
- The authentication server (which handles password entry and password change requests) is a protective device, so it *must* have a small attack surface
- ☞ In other words: the interface to that server is utterly crucial from a security perspective
- What other functions should we separate?
- Credit card numbers—they should *never* be readable, but need to be sent to the billing process

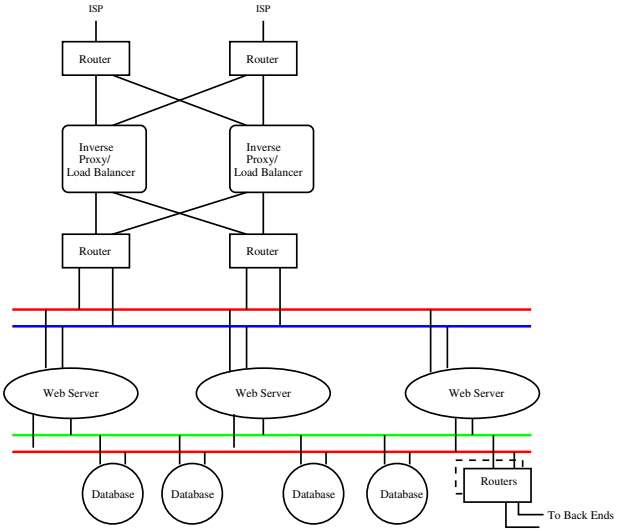
Billing


Credit card payments
Billing
Firewall
Firewall
Cust.
Web server
User DB
Credit card DB
Authentication server
Auth DB

Why the Firewalls?

- The obvious solution is to, e.g., turn off all ports but HTTPS on the web server
- However—how do you *administer* it?
- How do you patch it, update your code, install new certficates, monitor it, reboot it, and more?
- All elements in an operational environment—and that includes the firewalls—must be designed for operation and administration

A Web Server Complex



Why So Complex?

- Availability—primarily against ordinary failures
- *Everything* is replicated: routers, links, Ethernets, servers, etc.
- The only security feature of the redundancy is some protection against DDoS attacks if your two routers connect to different ISPs
- (We'll assume replication without further details on it—but it's not a trivial thing to do properly)

Other Security Functionality

- The inverse proxies are effectively firewalls—they only pass ports 80 and 443
- The database servers are not accessible from the outside—you have to hack through a web server to get any access at all (though that's easier than we would like)

Managing the Network Elements

- Some NOC machine has to be able to talk to the network elements
- The top (“north”) routers and the inverse proxies are exposed to the public Internet
- Must use strong cryptographic authentication
- Add network access control to limit sites that can talk to them
- What about the middle routers?
- Does the inverse proxy permit access to them? That’s a slight weakening of the firewall functionality
- Are they reached from the north LANs? Does that weaken the protection of the database servers?

How Many NOCs?

- The NOC really needs to see all network elements
- To talk to the south and middle routers, it has to be on the inside
- To talk to the north routers, it has to be able to reach the outside
- Some problems are most easily diagnosed if you have the ability to connect to all of the elements
- Conclusion: we need a special firewall for the NOC machines

Firewalls versus Hardened Hosts

- When do we use a firewall? (We can't put them everywhere.)
- When do we simply harden the host?
- What are the tradeoffs?

(Are Firewalls Themselves Secure?)

- In 1994, Bill Cheswick and I wrote

  Exposed machines should run as few programs as possible; the ones that are run should be as small as possible.

  . . .

  Most hosts cannot meet our requirements: they run too many programs that are too large. Therefore, the only solution is to isolate them behind a firewall if you wish to run any programs at all.

- Today, though, host software is better—and firewalls are worse, because they handle too many protocols and are too complex
- What is the tradeoff?

Firewalls versus Hardened Hosts

Use Firewalls If. . .

- Attackers likely mostly on the “bad” side
- At-risk, mandatory services on the hosts that need not be accessed from the bad side
- High-value resources on the “good” side

Use Hardened Hosts If. . .

- Higher probability of attackers inside
- Required protocols are so complex it’s unclear if the firewall does them better
- Only (probably) secure services running
- Rich connectivity needed to the (nominally) protected hosts
- Very complex firewall configuration needed

Firewalls versus Hardened Hosts

- The web server computer can be hardened, except for the web server itself
- But those ports can't easily be protected by a firewall
- The load balancers provide some firewall functionality anwyay
- Conclusion: no extra firewall needed
- Your whole e-commerce complex—many computers, many protocols, etc.—is behind the web server; a good spot for a firewall
- The authentication server is simple, and can protect the database itself
- Similar arguments for the credit card and billing systems

Specialized Servers

- For authentication and credit card payments, the server talks to another server instead of to the database. Why?
- We need a specialized request: not “give me this data”, but “do something with this data” or “send this data elsewhere
- We do not want to trust the web server completely, so we don’t give it the ability to handle too much sensitive data
- Query: should password change requests go through the regular web server, or should be there a separate web server for that, possibly running on the authentication server?

Databases and Access Control

- It’s possible to have separate logins to a database; each login can have different permissions for different rows, columns, and tables
- However: when a web server is accessing a database, it can’t (rationally) have a separate login for each web user—the web server itself is a database user and has a stored login and password
- (Also: remember SQL injection attacks—use stored procedures!)

Shopping and Inventory Management

- Customers shop for items; add them to their shopping cart
- Eventually, they're shipped; inventory needs to be debited
- The inventory system needs to place orders for more items when stock runs low
- Product managers need to add more items to the database, to be ordered as usual
- Do suppliers ever need to call in?

Security Design Decisions

- Can the web server read from the inventory database, or must it ask an inventory server?
- When a customer orders something, does the web server debit the inventory, or does an ordering server do that?
- Who creates the shipping order?
- There isn't necessarily one right answer! There are always tradeoffs

A Separate Inventory/Ordering Server?

- A complex protocol from/to the web server—all sorts of browsing choices
- The shopping cart has to be on or easily accessible to the web server
- The logic for all of that is so complicated that it's very unclear that this inventory/ordering server would be more secure than the web server
- But: what are the risks to the inventory/ordering system if it or the web server are hacked?

Risks

Theft	Order high-value items, e.g., electronics or jewelry, to be shipped to the thief
Mischief	Order many mundane items, to be shipped to someone
Vandalism	Change inventory quantities
Vandalism	Change prices of items—make them lower, for theft; make them higher, for improper competition

How do we counter those risks?

Countermeasures: Theft

- Require extra authentication on problematic transactions, e.g., something very expensive or going to a different address
- Use machine learning algorithms to spot other trouble areas, e.g., shipments to an address that has had too much trouble, or to a neighborhood with too many porch pirates
- Listen to the credit card companies—they're good at spotting this stuff, too, and may ask the customer to verify
- Typical types of extra authentication: re-enter credit card number or perhaps the CVV; message to an app; text message
- Btw: the legal environment matters. In the US—but not in some other countries—the consumer is not liable for fraudulent credit card charges, so merchants have to be extra careful

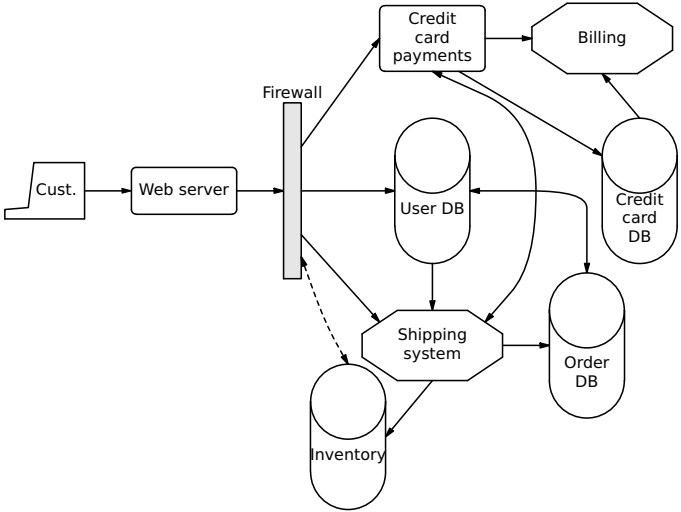
The Role of Logging and Auditing

- All items ordered are logged
- Anomaly detectors look for unusual patterns (three years ago, a run on toilet paper?)
- Reconcile shipped quantities with remaining inventory
- Note that this requires the audit process to be separate and secure—but per the evaluation lecture, that's necessary anyway

Buying Something

- The customer browses product offerings
- The web server does the database queries, and maintains the user's shopping cart (via cookies)
- On checkout, the web server sends the shopping cart to the shipping system, which tells the credit card system "Use card #*n*"
- The shipping system also adjusts the inventory

Buying Things



Security Properties

- The web server—by assumption, less secure—cannot adjust the inventory
- No one but the billing server ever sees a credit card number
- The shipping system adjusts the inventory after getting the go-ahead from the credit card system
- The order database contains all of the relevant details for the customer to view
- (Not covered: returns, restocking, etc.)

Which Are the Most Security-Critical Elements?

And how do we protect them?

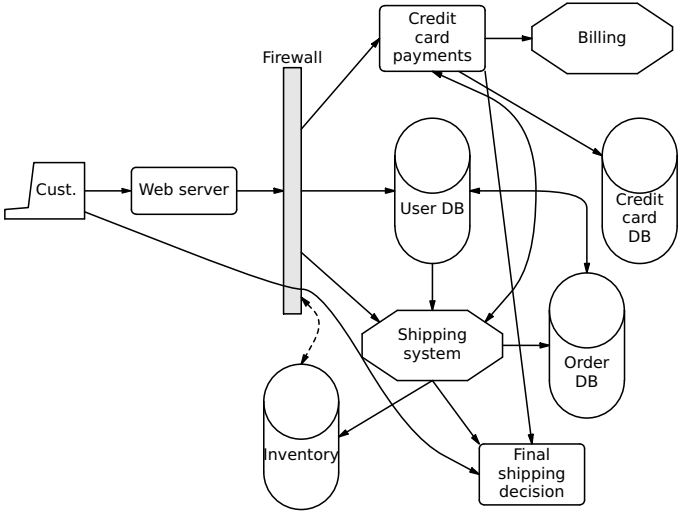
Which Are the Most Security-Critical Elements?

- The shipping system is the most sensitive—it can send goods to people, and bill credit cards
- The credit card payment system is also sensitive, but far less complex
- How can we guard against a compromised shipping system?

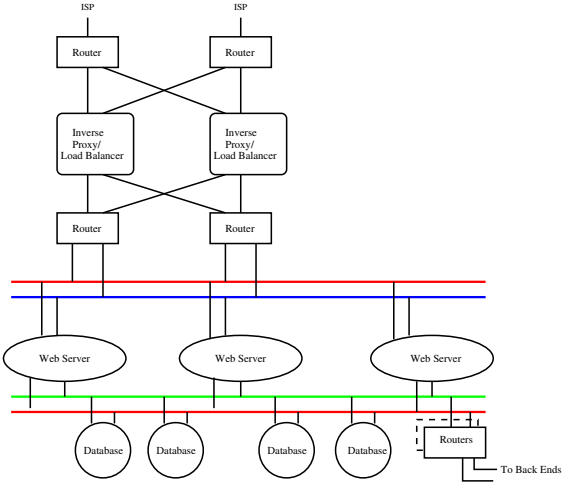
Split Responsibility

- The web server tells the shipping system to prepare the order
- The web server also tells the payment system to do the billing
- The payment system *and* the shipping system both tell a new server to go ahead with the order
- Plus: pass authentication credentials from the *browser* to this new system

Final Configuration



Another Look at the Server Complex



Look at the bottom right: “To Back Ends”. What is that about?

Back End Processes

- Databases don't just appear—they're created, maintained, etc.
- Similarly, transaction history has to be downloaded for routine analysis
- Note well: this is *not* the same as security logging and analysis
- (Why not?)
- A lot of corporate-level employees need access to this sort of data—how do we protect it?

Corporate Net Access to the Sales Complex

- We do *not* want to allow general access to the sales complex from the general corporate net
- But—many employees need access
- Analysts, etc., need read access
- Developers, marketers, etc., often need write access
- How do we do this securely?

No One Solution!

- Different people need very different types of access
- There's no reason to use only one mechanism
- Example: for analysts, copy the necessary files and databases to (a protected location) on the general corporate net
- Example: give NOC and other IT people multiple computers: one on the production net, one on the corporate net, etc.

Moving Files to the Production Net

- First and foremost, there is an important management issue here: *nothing* should be moved to production servers without proper approval, testing, etc.
- This strongly suggests a two-stage process: first, a secure, managed copy to the corporate test environment; second, a secure, managed copy to the production net
- (Use the same process for both copies)
- The requester packages the files and stages them to a border machine
- A privileged user on the inside of that network examines the request, unpackages, tests, etc.
- Not an airgap—but a tightly managed copy
- (Query: why is this border machine secure?)

What Did We Do?

- Identify the valuable resources
- Identify risky systems: ones that (a) had access to those resources, and (b) had a high attack surface
- We then improved security by changing access patterns and/or adding additional controls
- This isn't foolproof—but it is an *engineering* approach. Nothing is risk-free, but we can lower risk.
- N.B. My solutions are far from the only ones!

Questions?

(Evening grosbeak, Riverside Park, February 3, 2019)