Security Diagramming Guidance

Overview:

In this document we'll be focused on proper diagramming for threat modeling that conveys a lot of information very quickly that can be used for both in-house developed systems or Commercial Off The Shelf (COTS) products. In threat modeling, security teams must strike a balance between effective risk mitigation and project velocity. All project teams want to deliver their product fast and without security teams getting in the way. Security teams need to identify risks before the company is encumbered with them. Catching problems early in the planning stage can save an organization a lot of time and effort which usually means a lot of money as well.

To be effective at getting security relevant answers out of designs and project teams, we must follow a regimented process for each engagement. We'll be talking about various challenges to the diagramming process. Each of these examples are going to show the bad way to do the diagramming and then the good way for instruction. After those challenges is a checklist that each author should review after each diagramming session to ensure that all the elements are matching the best practices pattern.

Next, we have tables that show what information is needed for each link. Project teams can expect to list out a dozen or two dozen of these tables to properly describe all the data transiting the network.

Finally, we'll have an example of a simple diagram and table set that will help the reader.

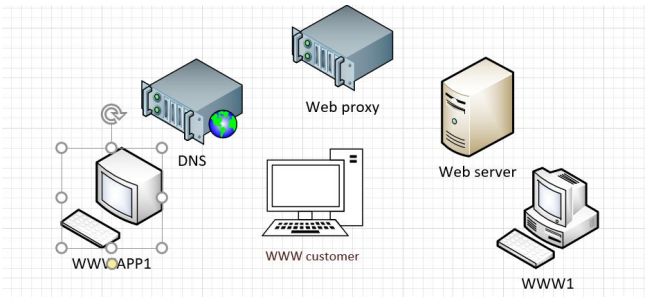
Table of Contents

Security Diagramming Challenges:

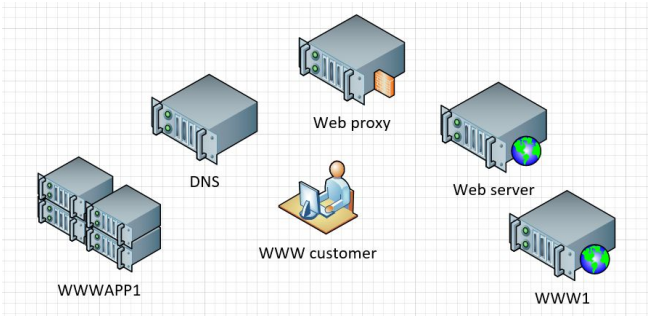
Challenge Diagram Example - All nodes using the same design palette:

Use consistent design palettes and sizes. It makes it easier to read and understand. It has a more professional appearance. As a security person, I want to be able to determine client stations from servers very easily. It's nice to make labels but that forces the user to read all the map at once rather than taking it in one layer at a time.

Bad way:



Better way:

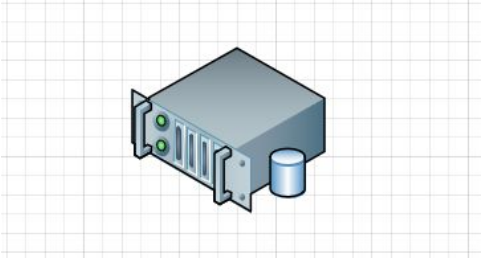


Challenge Diagram Example - Don’t use photos in diagrams:

Sometimes a stencil is not available, and people will turn to pictures of the objects. This is bad for a couple reasons, firstly, it inflates the size of the file and that can cause other problems downstream. Pictures sometimes lack contrast when printing them at a small size. They also use more ink generally than an icon. Pictures lack the functionality needed for flexible connector terminating in older versions of Visio. This also breaks the benefits of consistency that we described earlier.

Bad way:

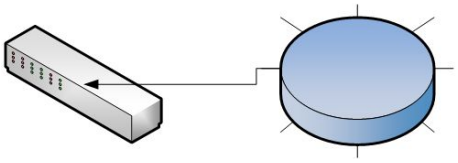
Better way:



Challenge Diagram Example - Terminate all lines and make sure they're visible:

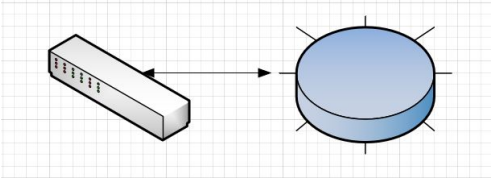
When you've completed your diagram make sure all the lines you've drawn terminate to an object and are connected. You'd be surprised how many times I see lines that don't go anywhere. Also, it's important to terminate the line so that if the object gets moved in the future, maybe to make room for a new object, then the line will stay connected. Also, it's vital to make sure that the end of the line is visible to the reader. Some stencils are not built correctly and are really a collection of grouped objects and those are notorious for not terminating correctly in a spot that's visible; I'm looking at you AWS and Azure. Arrowheads should show the flow of information too. Remember, from a security perspective I need to follow the data and find repositories of data and I'm less interested in the sequence of movement. This is backed up in the security plan document, but it starts here with quick information dissemination. On bigger systems the connectors tend to all be double arrowheads and so we have to rely on the data flow tables to understand what's going on.

Bad way:



This line has two arrow heads!!!

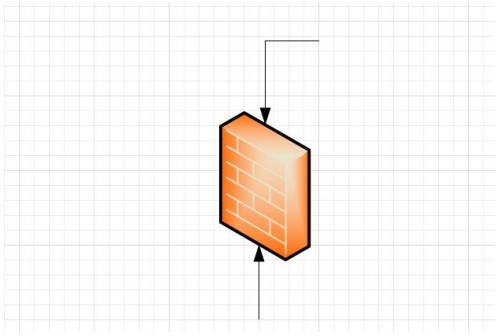
Better way:



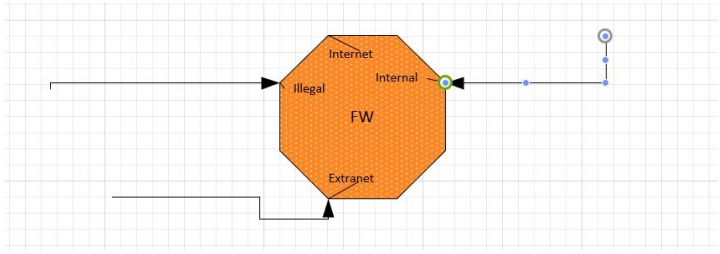
Challenge Diagram Example - Firewall Connections:

This issue is with the default firewall icon in Visio you can diagram the connections in a way that it becomes unclear where the traffic is going. The connected node needs to be clearly designated what network it's on and that the traffic is passing through the firewall.

Bad way:



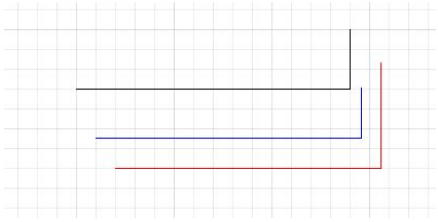
Better way:



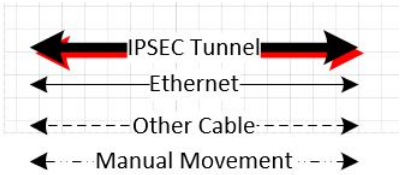
Challenge Diagram Example - Line colors and Dashes:

Many people use coloring to provide context to a link. This is bad for a couple reasons. Firstly, people who are color blind cannot see those colors clear enough or not at all. Using a couple forms of dashes is fine but those have a weakness such as when a line is too short it doesn't clearly show the pattern so don't do too many variations. Shadows can be used to help, but recently there was a bug in Visio 2016 where the shadow was placed too far away, so be flexible with that. Also, build a legend that will be consistent on all diagrams.

Bad way:



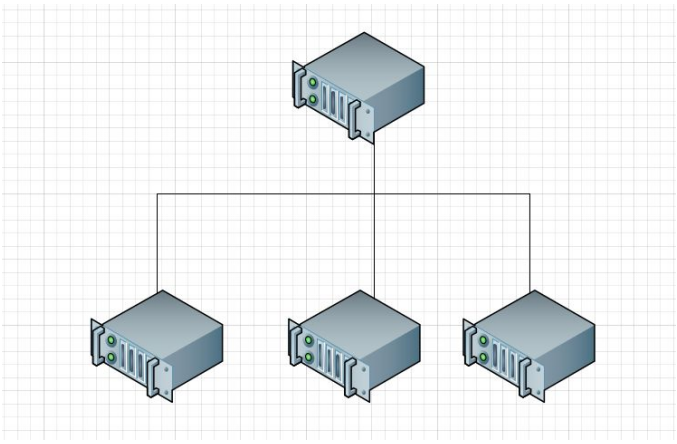
Better way:



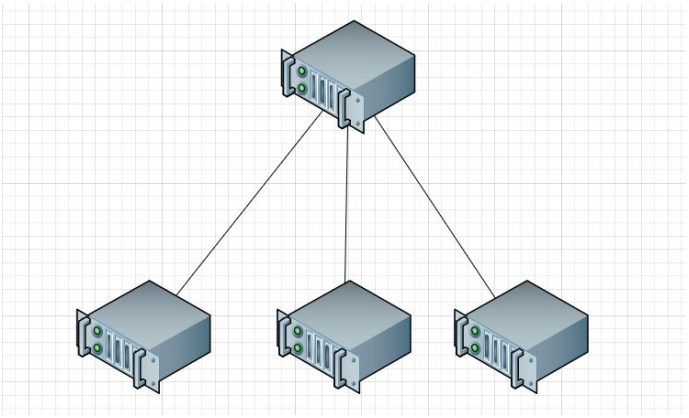
Challenge Diagram Example - Line shaping:

For security the most important piece of information that we need to convey in a diagram is how the data is getting from point A to B and what point A and B are. I've seen so many diagrams that share line paths. This kills the idea that I can know what information is going where. If the lines are redundant get rid of the extra lines.

Bad way:



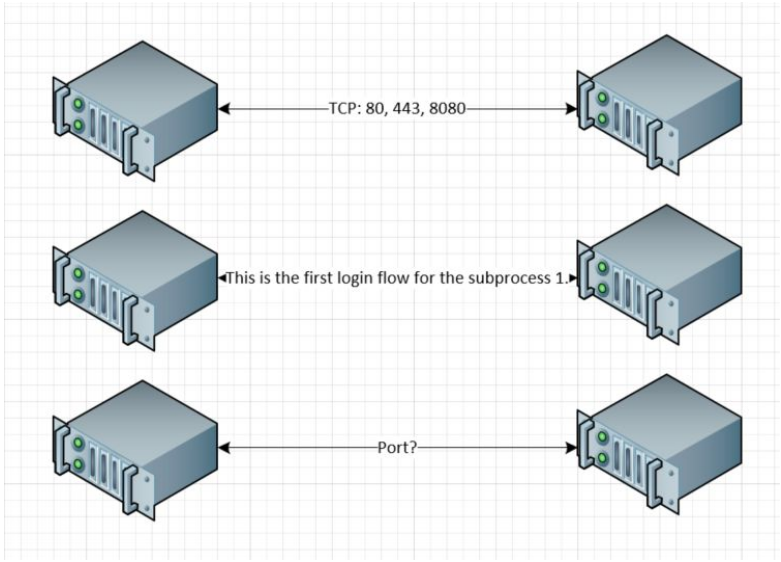
Better way:



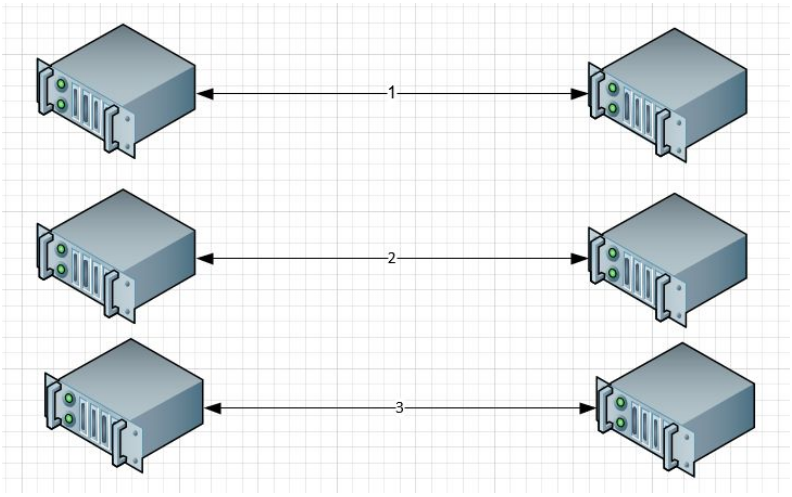
Challenge Diagram Example - Keep the flow labels simple:

People have many different ideas for what go on a link label. I'm going to say don't do it. Label your link with a number that corresponds to a table that describes in detail what's going on. You must do it anyways, so just keep it simple here.

Bad way:



Better way:



Challenge Diagram Example - Add a Title, Date, and Legend:

It's a good habit to put a title, last modified date, and a legend on all your diagrams. Remember that in print form the name of the file doesn't help you. Also, as a side note there are entities out there that require this information for submissions to their system. For example, system designs that are submitted for criminal justice information system(CJIS) access require this information on all diagrams.

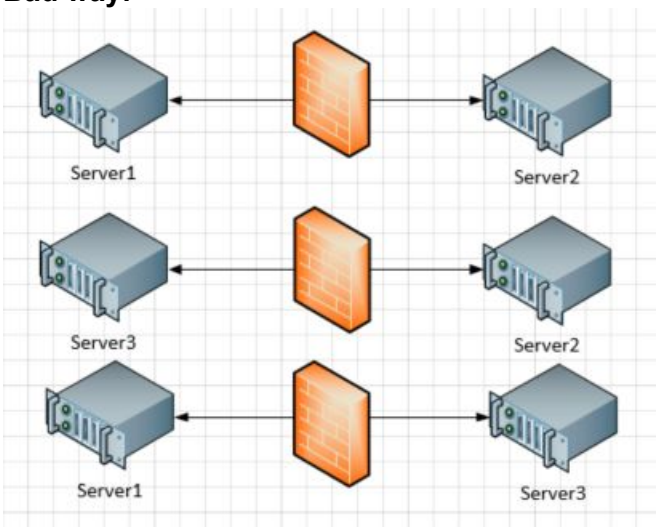
Bad way:
Nothing.

Better way:

Challenge Diagram Example - Use unique names for all entities:

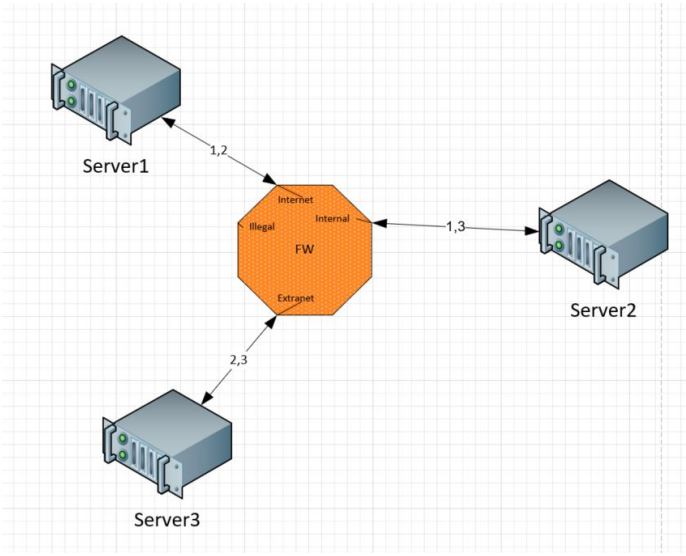
This one gets challenged from time to time but believe me it will help reduce confusion. When collaborating on a diagram it's easier to call out items for discussion if they have unique names. Also, it dispels the idea that it could be the same object. This example below is a real-world example that I had.

Bad way:



Several servers were on the map multiple times and the firewall was the same firewall. However, the diagram that I was given made it look like there were two times the servers and three times the firewalls.

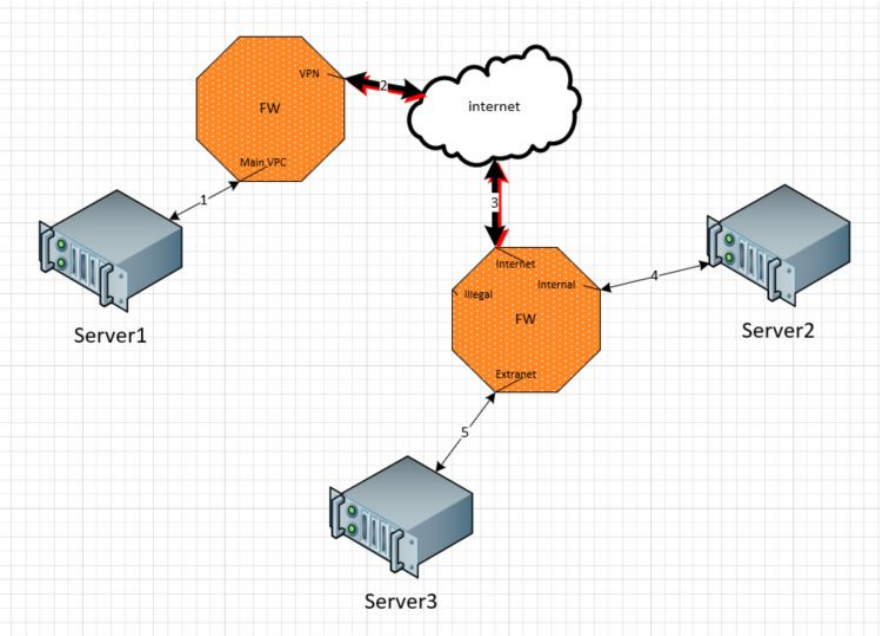
Better way:



Challenge Diagram Example - Link lines are to be labeled NODE-TO-NODE:

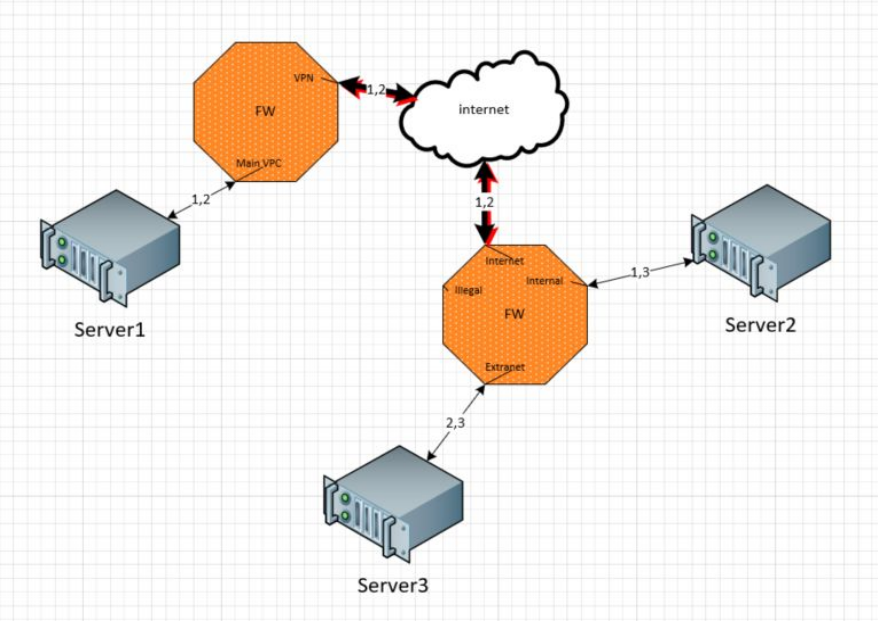
This is the most important note and concept. Always, always, always, label your links with a complete flow from ***node to node***. If the data channel passes through a device, do not waste effort relabeling it. There will be massive confusion and extra work that provides no security benefit. This applies to only node to node scenarios, do not use the same link number on two different links because they use the same protocol or perform the same work. Those are different nodes and should have different link numbers.

Bad way:



Look at this example. You can't tell which node is communicating to which other node.

Better way:

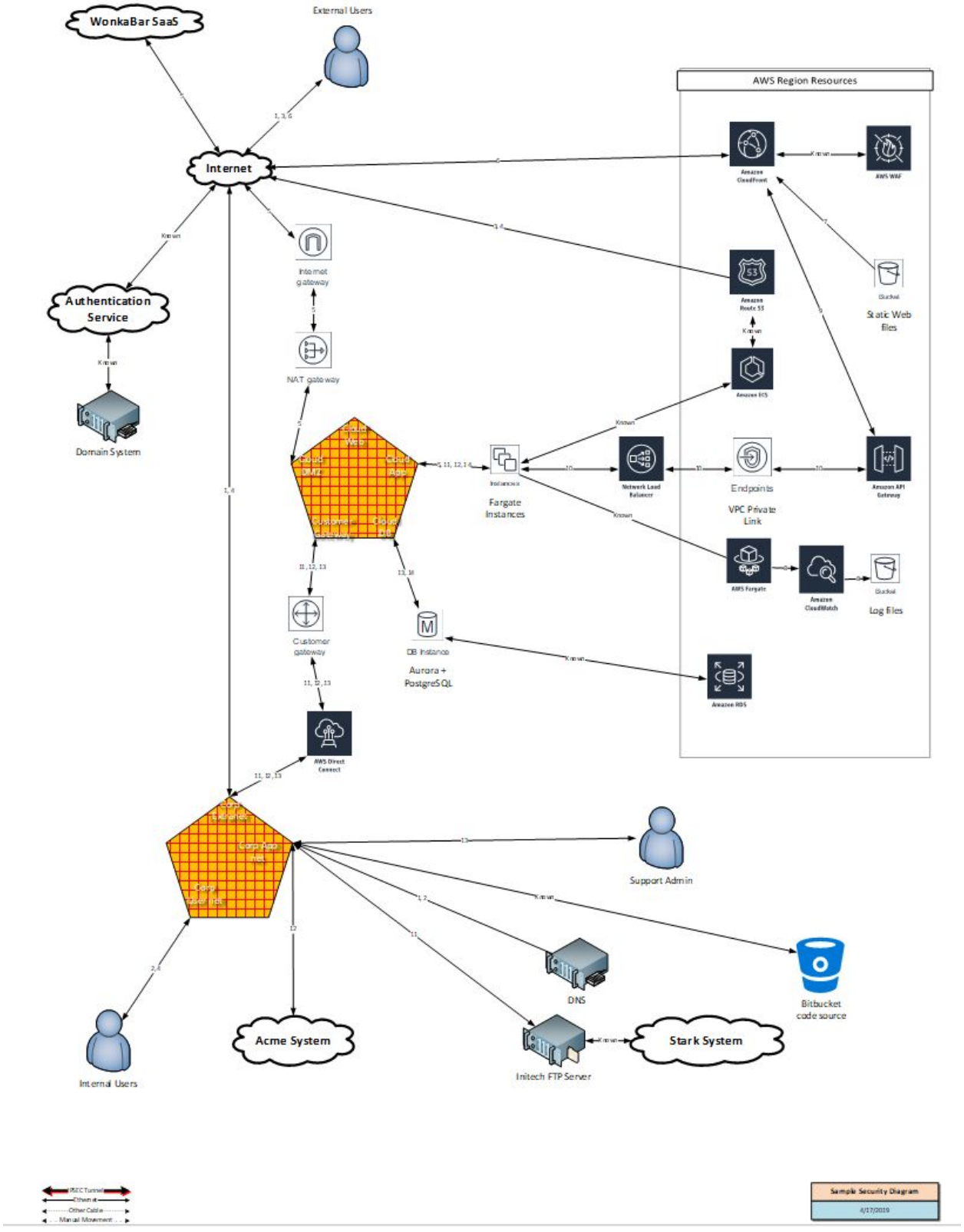


Now, you can tell exactly which servers are talking to each other and with our labeling of IPSec channels you can have that added context without creating new labels.

Challenge Diagram Checklist:

- ❏ All nodes using the same design palette.
- ❏ All nodes the same size.
- ❏ All photos eliminated from the diagram.
- ❏ All lines terminated to an object.
- ❏ All line ends visible, so the use can determine if the line contains an arrowhead or not.
- ❏ All line types properly indicate the mode of transfer.
- ❏ All lines marked with at least one data flow number.
- ❏ All link labels are used between two node entities only.
- ❏ The diagram is titled on the page.
- ❏ The diagram is dated on the page.
- ❏ All relevant networks separated by the firewall are labeled on the firewall.
- ❏ All lines terminating at a firewall terminate to a labeled point on the firewall.
- ❏ All entities are labeled uniquely.
- ❏ Other systems that have an integration with this system are represented by a cloud and labeled.

Example Diagram:



Example Tables (not a complete list, but some of the flows to explain the content needed):

1 – External Users –> DNS

Field	Value				
Description (Be very specific.)	External users query on-premise DNS servers for the IP corresponding the domain name searched for.				
Which Node Initiates the communication?	External users		Protocol and Port (HTTPs: 443)		DNS: 53
Transport Encryption Used (E.g. TLS 1.2)	None		Payload Encryption Used (E.g. AES 256)		None
Data Fields (Sample fields of highest classification. E.g. SSN, Post Sales VIN, etc.)	Domain name record		Data Classification (E.g. Protect, Confidential, etc.)		Public
Timing (E.g. Nightly, On-demand, etc.)	On-demand	Trans./day	50,000-100,000	Est. avg file size	<1kb
Authentication Technique (E.g. Domain\UserAccount, Domain\ServiceAccount, OAUTH2 Implicit Grant, etc.)	No authentication – Anonymous				
Rotation period for credentials (E.g. 60days, 1hour, etc.)	N/A no credentials used.				

5 – WonkaBar SaaS –> Fargate Instances

Field	Value				
Description (Be very specific.)	Fargate Instances pushes data via webservice call to WonkaBar SaaS.				
Which Node Initiates the communication?	Fargate Instances		Protocol and Port (HTTPs: 443)		HTTPS: 443
Transport Encryption Used (E.g. TLS 1.2)	TLS 1.1		Payload Encryption Used (E.g. AES 256)		None
Data Fields (Sample fields of highest classification. E.g. SSN, Post Sales VIN, etc.)	Lists of widgets, shipping dates, CustomerID, etc.		Data Classification (E.g. Protect, Confidential, etc.)		Protected
Timing (E.g. Nightly, On-demand, etc.)	On-demand	Trans./day	10,000-30,000	Est. avg file size	1kb
Authentication Technique (E.g. Domain\UserAccount, Domain\ServiceAccount, OAUTH2 Implicit Grant, etc.)	SFTP account				
Rotation period for credentials (E.g. 60days, 1hour, etc.)	60days				

14 – Fargate Instances <–> Aurora + PostgreSQL

Field	Value				
Description (Be very specific.)	The application performs CRUD functions against the application database.				
Which Node Initiates the communication?	Fargate Instances		Protocol and Port (HTTPs: 443)		TCP: 5432
Transport Encryption Used (E.g. TLS 1.2)	TLS 1.2		Payload Encryption Used (E.g. AES 256)		None
Data Fields (Sample fields of highest classification. E.g. SSN, Post Sales VIN, etc.)	Lists of widgets, shipping dates, CustomerID, etc.		Data Classification (E.g. Protect, Confidential, etc.)		Protected
Timing (E.g. Nightly, On-demand, etc.)	On-demand	Trans./day	10,000-30,000	Est. avg file size	1kb
Authentication Technique (E.g. Domain\UserAccount, Domain\ServiceAccount, OAUTH2 Implicit Grant, etc.)			Domain\ServiceAccount		
Rotation period for credentials (E.g. 60days, 1hour, etc.)			60days		

Data Flow Block Diagram Table Template:

1 – host 1 –> host 2

Field	Value				
Description (Be very specific.)					
Which Node Initiates the communication?			Protocol and Port (HTTPs: 443)		
Transport Encryption Used (E.g. TLS 1.2)			Payload Encryption Used (E.g. AES 256)		
Data Fields (Sample fields of highest classification. E.g. SSN, Post Sales VIN, etc.)			Data Classification (E.g. Protect, Confidential, etc.)		
Timing (E.g. Nightly, On-demand, etc.)		Trans./day		Est. avg file size	
Authentication Technique (E.g. Domain\UserAccount, Domain\ServiceAccount, OAUTH2 Implicit Grant, etc.)					
Rotation period for credentials (E.g. 60days, 1hour, etc.)					