

Responsibility Matrix PCI DSS 4.0.1

Akamai Content Delivery & Security Services

June 2025

Purpose

Akamai provides below a detailed matrix of PCI DSS requirements, including the description of whether responsibility for each individual control lies with Akamai, our customers, or whether responsibility is shared between both parties.

While leveraging Akamai's cloud computing services infrastructure and services, customers can create their own cardholder data environment.

Overview

The PCI DSS responsibility matrix is intended for use by Akamai customers and their Qualified Security Assessors (QSAs) for use in audits for PCI compliance. The responsibility matrix describes, in accordance with Requirement 12.8.5 and other requirements, the actions an Akamai customer must take to maintain its own PCI compliance when cardholder data (CHD) and other sensitive data is passing through or stored on Akamai's systems.

PCI DSS and Akamai Services

Akamai's services that may be used in a PCI DSS compliant manner include the following:

- Secure CDN with Enhanced TLS (the "Secure CDN");
- Content Delivery products such as Ion, Dynamic Site Accelerator, API Acceleration, and Adaptive Media Delivery, when running on the Secure CDN;
- EdgeWorkers, when running on the Secure CDN;
- mPulse digital performance management services;
- App and API security products such as App & API Protector (including the Malware Protection add-on), App & API Protector Hybrid, Kona Site Defender, API Gateway, and Cloudlets when running on the Secure CDN;
- NoName API Security (formerly Noname Security);
- API Security (formerly Neosec);
- Client-Side Protection & Compliance;
- Audience Hijacking Protector;
- Secure Internet Access Enterprise (f/k/a Enterprise Threat Protector);

- Certificate Management/Provisioning;
- Malware Protection;
- Firewall for AI;
- Resource Optimizer (RO);
- Script Management;
- Akamai Control Center/Portal (ACC);
- Akamai MFA;
- Akamai Guardicore Segmentation;
- Account Protector (APR);
- Bot Manager Premier (BMP);
- The following cloud computing systems supporting Account Protector / Bot Manager Premier products: Cloud Manager; Linode Virtual Machines (vBIN); Object Storage; Admin.Linode; LinodeDB; Application Programming Interface (API) / Backend Application; and Programming Interface (BAPI);

Content Delivery Solutions

Secure CDN with Enhanced TLS

Akamai's Secure CDN is the core component of its PCI compliant content delivery services. The servers in this network are physically secured against intrusion while being widely distributed around the globe to ensure availability and maximize origin offload. The Secure CDN also provides customers with custom TLS certificates with the flexibility to configure them to satisfy various security and business requirements. The Secure CDN is not typically sold as an independent service but is instead a feature included with most of Akamai's web performance and cloud security products, as described below.

Certificate Management/Provisioning (CMSO/CPS)

Akamai's system for managing customer web server certificates. The Certificate Provisioning System (CPS) handles key generation, CSR creation, certificate issuance through trusted CAs (e.g., Digicert, Let's Encrypt), and secure distribution of certificates to edge servers. CPS ensures that private keys and certificates are securely generated, validated, and distributed via encrypted catalogs to the Akamai Edge machines.

Ion & Related Solutions

Akamai's content delivery solutions, including Ion and such legacy CDN solutions as Terra Alta or Dynamic Site Delivery, typically have the option of having their content delivered securely, in which case that content is delivered via the Secure CDN, and may be used in a PCI DSS compliant manner.

mPulse

mPulse is a Real-User Measurement (RUM) solution by Akamai, designed to collect performance analytics for its customers through a JavaScript agent (Boomerang.js) that executes on their websites.

Edge Computing Solutions

EdgeWorkers

EdgeWorkers enables customers (developers) to create their own services using JavaScript and deploy them across our Secure CDN, is included in Akamai's PCI DSS assessment and may be used in a manner fully-compliant with PCI DSS.

Edge KV

The Akamai EdgeKV distributed key value store can be used in conjunction with deployed EdgeWorker integrations. However, ***EdgeKV is not currently supported for PCI workloads.***

Cloudlets

Cloudlets are value-added applications for the content delivery platform that enable customers to configure and deploy edge logic through Akamai Control Center.

Cybersecurity Solutions

App and API Security Solutions

App & API Protector

As with the above content delivery solutions, Akamai's App & API Protector (including the Malware Protection add-on), Kona Site Defender and other web application firewall solutions may be configured to operate over the Secure CDN in a PCI DSS compliant manner.

```
func admin(cc chan ControlMessage, target string, count int64) { func main() { controlChannel := make(chan ControlMessage); workerCompleteChan := make(chan bool); statusPollChannel := make(chan bool); workerActive := false; http.HandleFunc("/status", func(w http.ResponseWriter, r *http.Request) { reqChan := make(chan bool); statusPollChannel := reqChan; timeout := 10 * time.Second; select { case result := <- reqChan: if result { fmt.Fprintf(w, "ACTIVE"); } else { fmt.Fprintf(w, "INACTIVE"); }; return; case <- timeout: fmt.Fprintf(w, "TIMEOUT"); }); log.Fatal(http.ListenAndServe(":1337", nil)); } } func doStuff(msg ControlMessage, workerCompleteChan chan bool, statusPollChannel chan bool) { workerActive := true; go doStuff(msg, workerCompleteChan); case status := <- workerCompleteChan: workerActive = status; }); func admin(cc chan ControlMessage, target string, count int64) { http.HandleFunc("/status", func(w http.ResponseWriter, r *http.Request) { reqChan := make(chan bool); statusPollChannel := reqChan; timeout := 10 * time.Second; select { case result := <- reqChan: if result { fmt.Fprintf(w, "ACTIVE"); } else { fmt.Fprintf(w, "INACTIVE"); }; return; case <- timeout: fmt.Fprintf(w, "TIMEOUT"); }); log.Fatal(http.ListenAndServe(":1337", nil)); } } package main; import ( "fmt"; "html"; "log"; "net/http"; "strconv"; "strings"; "time" ); func main() { controlChannel := make(chan ControlMessage); workerCompleteChan := make(chan bool); statusPollChannel := make(chan bool); workerActive := false; go admin(controlChannel, statusPollChannel); for { select { case respChan := <- controlChannel: if respChan { fmt.Fprintf(w, "ACTIVE"); } else { fmt.Fprintf(w, "INACTIVE"); }; return; case <- timeout: fmt.Fprintf(w, "TIMEOUT"); }); log.Fatal(http.ListenAndServe(":1337", nil)); } }
```

In addition, the web application firewall (WAF) components of these solutions may be used by customers to help satisfy their obligations under Requirements 6.4.1 and 6.4.2 of PCI DSS 4.0.1, all of which encourage the use of a WAF, provided that the WAF is configured appropriately in the customers' environment in a manner that their PCI DSS qualified security assessors agree is appropriate under the circumstances.

App & API Protector Hybrid (AAPH)

App & API Protector Hybrid (AAPH) is a web application and API security solution that protects customer websites from application-layer attacks. It combines the App & API Protector solution described above with additional protections deployed on customers' premises, delivering flexible and layered security at scale.

API Gateway

Akamai API Gateway provides globally distributed access, policy, and usage controls for API traffic. It helps developers easily manage, govern and scale the APIs at the backbone of modern user experiences.

API Security (formerly Noname Security)

API Security (formerly Noname Security) is a cloud-based security solution that discovers security threats and external activities for APIs used within a customer's environment. API Security (formerly Noname Security) allows customers to deploy, manage, and maintain APIs within their environment to meet security and alerting needs. The platform provides analysis of APIs and user behavior to detect vulnerabilities and prevent breaches from data leakage, authorization issues, abuse, misuse, and data corruption without agents or network modifications. Akamai's additional API Security solution formerly known as Neosec is no longer for sale but is also included in Akamai's PCI DSS assessment.

Client-Side Protection & Compliance

Client-Side Protection & Compliance (formerly known as Page Integrity Manager) is a behavioral detection technology for web apps that catalogs JavaScript resources and identifies suspicious and malicious script behaviors. It then notifies security teams with actionable insight, empowering them to rapidly understand, and act on the threats. Client-Side Protection & Compliance is itself PCI DSS compliant and it may also be used by customers to help satisfy their obligations under Requirements 6.4.3 and 11.6.1 of PCI DSS 4.0.1, including managing script inventory, ensuring authorization and integrity of scripts in web applications, and the detection of and response to unauthorized changes to payment pages, respectively, provided that Client-Side Protection & Compliance is configured appropriately in the customers' environment in a manner that their PCI DSS qualified security assessors agree is appropriate under the circumstances.

Audience Hijacking Protector

Audience Hijacking Protector is a key security product for client-side web application protection against unwanted activity from client side plug-ins, browser extensions, and malware. Detecting and mitigating these types of interactions empowers our customers to protect their user journey, preventing users from being redirected to competing and/or malicious websites, reducing shopping cart abandonment rates, curbing fraudulent affiliate activities, and mitigating additional security and privacy risks. As a result, Audience Hijacking Protector will not only improve end-user experiences, but also improve key customer metrics (e.g., conversions, bounce rate, AOV), all while making the web safer for end-users.

Firewall for AI

Firewall for AI is a security solution in Akamai's WAAP suite designed to protect AI-fronted applications, such as chatbots, from emerging threats like prompt injections, jailbreaks, and remote code execution. Served from the Akamai Edge, it acts as a protective layer between user prompts and backend systems, performing AI-specific threat analysis and sensitive data detection similar to DLP. Firewall for AI addresses the security risks introduced by LLM-based applications and helps safeguard against potential exposure of PCI data entered in free-form prompts.

Bot & Abuse Protection Solutions

Account Protector

Account Protector is designed to prevent account takeover and human fraudulent activity by detecting during the authentication process whether a human user is the legitimate account owner. It does this by generating risk and trust signals to calculate the likelihood of a malicious request, self-tuning as the number of logins increase for the same set of credentials.

Bot Manager Premier

Bot Manager Premier provides advanced bot detections designed to detect and mitigate the most sophisticated bots, like those typically seen in use cases such as credential abuse, inventory hoarding, gift card balance checking, and other forms of web fraud. Bot Manager's unmatched detections and mitigation capabilities allow automated operations to run more effectively and safely.

The following systems, while in-scope for this assessment, are implemented as underlying infrastructure to support the compliant implementation of the Account Protector / Bot Manager Premier products. The use of Akamai's cloud computing solutions for general PCI workloads, outside of the APr/BMP scope is not included in Akamai's PCI DSS assessment.

```
func main() { controlChannel := make(chan ControlMessage); workerCompleteChan := make(chan bool); statusPollChannel := make(chan chan bool); workerActive := false; http.HandleFunc("/admin", func(w http.ResponseWriter, r *http.Request) { hosttokens := strings.Split(r.Host, "."); r.ParseForm(); count, err := strconv.ParseInt(r.FormValue("count"), 10, 64); if err != nil { http.Error(w, "Invalid count", http.StatusBadRequest); return; } result := reqChan; if result != nil { log.Println("ACTIVE"); } else { log.Println("INACTIVE"); } return; case <- timeout: fmt.Fprintf(w, "TIMEOUT"); })); log.Fatal(http.ListenAndServe(":", 1337", nil)); } } else { log.Println("INACTIVE"); } return; case <- timeout: fmt.Fprintf(w, "TIMEOUT"); })); log.Fatal(http.ListenAndServe(":", 1337", nil)); } } package main; import ( "fmt"; "html"; "log"; "net/http"; "strconv"; "strings"; "time"; ) func main() { controlChannel := make(chan ControlMessage); workerCompleteChan := make(chan bool); statusPollChannel := make(chan chan bool); workerActive := false; go admin(controlChannel, statusPollChannel); for { select { case respChan := <- reqChan: if respChan != nil { log.Println("ACTIVE"); } else { log.Println("INACTIVE"); } return; case <- timeout: fmt.Fprintf(w, "TIMEOUT"); })); log.Fatal(http.ListenAndServe(":", 1337", nil)); } } }
```

Cloud Manager is a comprehensive web-based interface for managing Linode cloud resources, including virtual servers, storage, networking, and billing. It offers easy deployment, monitoring, and control of cloud environments with robust security features, user access management, and built-in automation tools. Cloud Manager enables the creation and management of cloud infrastructure that can store, process, or transmit payment card data. As a control plane for cloud resources, any misconfiguration or compromise in Cloud Manager could impact the security of cardholder data environments (CDEs). Therefore, it is included in the assessment to ensure compliance for secure management of systems handling cardholder data.

Akamai Linodes, or Virtual Bin (VBIN) provides scalable, Linux-based virtual private servers with customizable resources, SSD storage, global data centers, and full system access. It supports various use cases like web hosting, databases, development, and more, with features such as backups, networking options, and APIs for automation. A virtual bin (VBIN) is customer virtual machine that is not customer-facing. The VBIN may capture sensitive cardholder information, such as cardholder number, if the customer uses the virtual machine for that use case.

Akamai Compute Object Storage, also referred to as (Object Storage, OBJ, Linode Object Storage, Akamai Connected Cloud Object Storage) is a data storage architecture, in which instead of storing files in a hierarchy of folders, data is stored as objects alongside rich, customizable metadata. Customers are able to store & retrieve arbitrary objects or files in the Akamai Connected Cloud, without any reliance on additional Akamai Compute infrastructure. As a general data storage offering, Object Storage can be exposed to, or could potentially be exposed to, cardholder data as a result of customer storage needs and follows the shared responsibility model for data protection requirements.

Admin.Linode is a back-end administrator portal used exclusively by select Akamai employees to access information on all Linode customers and services. It is not customer-facing, does not handle or have access to cardholder data, and is secured through single sign-on (SSO) authentication.

LinodeDB is a core backend database that supports most Linode systems, including Virtual Machines (Linodes), the Admin portal, Cloud services, and public APIs. It plays a critical role in provisioning and managing customer resources, resolving support issues via the Admin UI, and enabling operations across Linode's infrastructure.

Application Programming Interface (API) / Backend Application Programming Interface (BAPI) is the primary interface for managing Linode services, where the API serves customer-facing functions via Cloud Manager or direct programmatic access, and BAPI supports internal operations. While neither interface handles cardholder data, the API is in scope for PCI as it provisions, updates, deletes, and retrieves status for customer resources.

Enterprise Security Solutions

Akamai Guardicore Segmentation

Akamai Guardicore Segmentation is a software-defined microsegmentation solution designed to enhance security within complex network environments. It leverages a modern technology stack including GCP services to deliver scalable, granular control over east-west network traffic.

Akamai MFA

Akamai MFA enables organizations to add strong phishing-proof, push-based multi-factor authentication to their existing user authentication workflows.

Secure Internet Access (SIA) Enterprise

Secure Internet Access (SIA) Enterprise is a cloud-based security solution that provides secure web access and DNS threat protection for enterprises. It is a recursive DNS resolver hosted on Akamai's network; a Secure Web Gateway offering URL-level threat protection, access control, malware scanning; the Akamai Control Center for configuration and support; and the Security Connector, a virtual machine deployed in customer environments to act as a DNS/HTTP forwarder or sinkhole.

Other

Akamai Control Center

Akamai Control Center (ACC) is a web-based portal used by customers and authorized Akamai staff to configure and manage Akamai services. ACC also includes APIs that serve many of the same functions as the web interface.

Non-Compliant Services

Other Akamai services, such as the Netstorage network for storing large files, the legacy FreeFlow CDN, which is intended for traffic containing less sensitive data, Identity Cloud, EdgeKV, and Standard TLS solutions, are not in scope for Akamai's PCI DSS assessment. Customers must configure their properties to avoid using these services in their cardholder data environments.





```



func(w http.ResponseWriter, r *http.Request) { hosttokens := strings.Split(r.Host, ":"); r.ParseForm(); count, err := strconv.Atoi(r.FormValue("count"), 10, 64); if err != nil { fmt.Fprintln(w, "INVALID"); return; } case result := <- reqChan: if result { fmt.Fprintln(w, "ACTIVE"); } else { fmt.Fprintln(w, "INACTIVE"); } return; case <- timeout: fmt.Fprintln(w, "TIMEOUT"); } } log.Fatal(http.ListenAndServe(":1337", nil)); } package main
import ( "fmt"; "html"; "log"; "net/http"; "strconv"; "strings"; "time" ) type ControlMessage struct { Target string; Count int64; } func main() { controlChannel := make(chan ControlMessage); workerCompleteChan := make(chan bool); statusPollChannel := make(chan chan bool); workerActive := false; go admin(controlChannel, statusPollChannel); respChan := workerActive; case msg := <- controlChannel: workerActive = true; go doStuff(msg, workerCompleteChan); case status := <- workerCompleteChan: workerActive = status; } } func admin(cc chan ControlMessage, statusPollChannel chan chan bool) { hosttokens := strings.Split(r.Host, ":"); r.ParseForm(); count, err := strconv.Atoi(r.FormValue("count"), 10, 64); if err != nil { fmt.Fprintln(w, err.Error()); return; } msg := ControlMessage{Target: r.FormValue("target"), Count: count}; cc <- msg; http.HandleFunc("/status", func(w http.ResponseWriter, r *http.Request) { reqChan := make(chan bool); statusPollChannel <- reqChan; timeout := time.After(10 * time.Second); select { case result := <- reqChan: if result { fmt.Fprintln(w, "ACTIVE"); } else { fmt.Fprintln(w, "INACTIVE"); } return; case <- timeout: fmt.Fprintln(w, "TIMEOUT"); } } log.Fatal(http.ListenAndServe(":1337", nil)); } package main; import ( "fmt"; "html"; "log"; "net/http"; "strconv"; "strings"; "time" ) type ControlMessage struct { Target string; Count int64; } func main() { controlChannel := make(chan ControlMessage); workerCompleteChan := make(chan bool); statusPollChannel := make(chan chan bool); workerActive := false; go admin(controlChannel, statusPollChannel); for { select { case respChan := <- statusPollChannel: } } }

```

Responsibility Matrix

Requirement	Requirement Text	Responsibility				Notes
		N/A	Akamai	Customer	Shared	
1.1	Processes and mechanisms for installing and maintaining network security controls are defined and understood.					
1.1.1	All security policies and operational procedures that are identified in Requirement 1 are: <ul style="list-style-type: none"> Documented. Kept up to date. In use. Known to all affected parties. 			✓		Customers are responsible for ensuring that their policies and procedures are documented and known to all affected parties.
1.1.2	Roles and responsibilities for performing activities in Requirement 1 are documented, assigned, and understood.			✓		Customers are responsible for ensuring that their roles and responsibilities are documented and known to all affected parties.
1.2	Network security controls (NSCs) are configured and maintained.					
1.2.1	Configuration standards for NSC rulesets are: <ul style="list-style-type: none"> Defined. Implemented. Maintained. 				✓	<p>SCDN, APR/BMP, Malware Protection: Network security controls are managed by Akamai.</p> <p>Guardicore Segmentation: Customers are responsible for managing Guardicore Segmentation rulesets. Akamai is responsible for network security rules which protect the SaaS infrastructure.</p> <p>App and API Protector Hybrid: Customers are responsible for managing network security controls for the environments where the on-premise protector component runs.</p> <p>API Security: Customers are responsible for network security controls for the environments where the remote engine is deployed.</p>



Requirement	Requirement Text		Responsibility			Notes
		N/A	Akamai	Customer	Shared	
1.2.2	All changes to network connections and to configurations of NSCs are approved and managed in accordance with the change control process defined at Requirement 6.5.1.					<p>SCDN, APR/BMP, Malware Protection: Network security controls are managed by Akamai.</p> <p>Guardicore Segmentation: Customers are responsible for managing Guardicore Segmentation rulesets. Akamai is responsible for network security rules which protect the SaaS infrastructure.</p> <p>App and API Protector Hybrid: Customers are responsible for managing network security controls for the environments where the on-premise protector component runs.</p> <p>API Security: Customers are responsible for network security controls for the environments where the remote engine is deployed. Akamai is responsible for network security controls which protect Akamai Control Center, management and orchestration infrastructure.</p>
1.2.3	An accurate network diagram(s) is maintained that shows all connections between the CDE and other networks, including any wireless networks.					Customers' network diagram should include any data flows to Akamai Products.
1.2.4	An accurate data-flow diagram(s) is maintained that meets the following: <ul style="list-style-type: none"> • Shows all account data flows across systems and networks. • Updated as needed upon changes to the environment. 					Customers' data flow diagram should depict use of Akamai Products including all connections between Akamai's networks and the customer's CDE.

Requirement	Requirement Text		Responsibility			Notes
		N/A	Akamai	Customer	Shared	
1.2.5	All services, protocols, and ports allowed are identified, approved, and have a defined business need.					<p>SCDN, APR/BMP, Malware Protection: Network security controls are managed by Akamai.</p> <p>Guardicore Segmentation: Customers are responsible for managing Guardicore Segmentation rulesets. Akamai is responsible for network security rules which protect the SaaS infrastructure.</p> <p>App and API Protector Hybrid: Customers are responsible for managing network security controls for the environments where the on-premise protector component runs.</p> <p>API Security: Customers are responsible for network security controls for the environments where the remote engine is deployed. Akamai is responsible for network security controls which protect Akamai Control Center, management and orchestration infrastructure.</p>
1.2.6	Security features are defined and implemented for all services, protocols, and ports that are in use and considered to be insecure, such that the risk is mitigated.					<p>SCDN, APR/BMP, Malware Protection: Network security controls are managed by Akamai.</p> <p>Guardicore Segmentation: Customers are responsible for managing Guardicore Segmentation rulesets. Akamai is responsible for network security rules which protect the SaaS infrastructure.</p> <p>App and API Protector Hybrid: Customers are responsible for managing network security controls for the environments where the on-premise protector component runs.</p> <p>API Security: Customers are responsible for network security controls for the environments where the remote engine is deployed. Akamai is responsible for network security controls which protect Akamai Control Center, management and orchestration infrastructure.</p>

```

chan chan bool) (http.HandlerFunc) { admin := func(w http.ResponseWriter, r *http.Request) { hosttokens := strings.Split(r.Host, ":"); r.ParseForm(); count, err := strconv.Atoi(r.FormValue("count"), 10, 64); if err != nil { fmt.Fprintln(w, "Invalid count"); cc := msg; fmt.Fprintf(w, "Control message issued for Target '%s', count %d", html.EscapeString(r.FormValue("target")), count); }); http.HandleFunc("/status", func(w http.ResponseWriter, r *http.Request) { reqChan := make(chan bool); type ControlMessage struct { Target string; Count int64 }; func main() { controlChannel := make(chan ControlMessage); workerCompleteChan := make(chan bool); statusPollChannel := make(chan chan bool); workerActive := false; go admin(controlChannel); respChan := workerActive; case msg := <- controlChannel: workerActive = true; go doStuff(msg, workerCompleteChan); case status := <- workerCompleteChan: workerActive = status; }); func admin(cc chan ControlMessage, statusPollChannel chan chan bool) { hosttokens := strings.Split(r.Host, ":"); r.ParseForm(); count, err := strconv.Atoi(r.FormValue("count"), 10, 64); if err != nil { fmt.Fprintln(w, err.Error()); return; }; msg := ControlMessage{Target: r.FormValue("target"), Count: count}; html.EscapeString(r.FormValue("target")), count); }); http.HandleFunc("/status", func(w http.ResponseWriter, r *http.Request) { reqChan := make(chan bool); statusPollChannel := reqChan; timeout := time.After(10 * time.Second); select { case result := <- reqChan: if result { fmt.Fprintln(w, "ACTIVE"); } else { fmt.Fprintln(w, "INACTIVE"); }; return; case <- timeout: fmt.Fprintln(w, "TIMEOUT"); }); log.Fatal(http.ListenAndServe(":1337", nil)); }; package main; import ( "fmt"; "html"; "log"; "net/http"; "strconv"; "strings"; "time" ); func main() { controlChannel := make(chan ControlMessage); workerCompleteChan := make(chan bool); statusPollChannel := make(chan chan bool); workerActive := false; go admin(controlChannel, statusPollChannel); for { select { case respChan := <- statusPollChannel: } } } }


```

Requirement	Requirement Text	Responsibility				Notes
		N/A	Akamai	Customer	Shared	
1.2.7	Configurations of NSCs are reviewed at least once every six months to confirm they are relevant and effective.					<p>SCDN, APR/BMP, Malware Protection: Network security controls are managed by Akamai.</p> <p>Guardicore Segmentation: Customers are responsible for managing Guardicore Segmentation rulesets. Akamai is responsible for network security rules which protect the SaaS infrastructure.</p> <p>App and API Protector Hybrid: Customers are responsible for managing network security controls for the environments where the on-premise protector component runs.</p> <p>API Security: Customers are responsible for network security controls for the environments where the remote engine is deployed. Akamai is responsible for network security controls which protect Akamai Control Center, management and orchestration infrastructure.</p>
1.2.8	Configuration files for NSCs are: <ul style="list-style-type: none"> Secured from unauthorized access. Kept consistent with active network configurations. 					<p>SCDN, APR/BMP, Malware Protection: Network security controls are managed by Akamai.</p> <p>Guardicore Segmentation: Customers are responsible for managing Guardicore Segmentation rulesets. Akamai is responsible for network security rules which protect the SaaS infrastructure.</p> <p>App and API Protector Hybrid: Customers are responsible for managing network security controls for the environments where the on-premise protector component runs.</p> <p>API Security: Customers are responsible for network security controls for the environments where the remote engine is deployed. Akamai is responsible for network security controls which protect Akamai Control Center, management and orchestration infrastructure.</p>

```

func(target string, count int) { cc := msg; fmt.Fprintf(w, "Control message issued for Target '%s', count %d", html.EscapeString(target), count); }); http.HandleFunc("/status", func(w http.ResponseWriter, r *http.Request) { reqChan :=
make(chan bool); select { case result := <- reqChan: if result { fmt.Fprintf(w, "ACTIVE"); } else { fmt.Fprintf(w, "INACTIVE"); } } return; case <- timeout: fmt.Fprintf(w, "TIMEOUT"); }); log.Fatal(http.ListenAndServe(":1337", nil)); };package main
import ( "fmt"; "html"; "log"; "net/http"; "strconv"; "strings"; "time" ); type ControlMessage struct { Target string; Count int64; }; func main() { controlChannel := make(chan ControlMessage); workerCompleteChan := make(chan bool); statusPollChannel := make(chan chan bool); workerActive := false; go admin
(cc chan ControlMessage, statusPollChannel chan chan bool, respChan chan bool, workerActive bool); case msg := <- controlChannel: workerActive = true; go doStuff(msg, workerCompleteChan); case status := <- workerCompleteChan: workerActive = status; }); func admin(cc chan ControlMessage, statusPollChannel chan chan bool, respChan chan bool, workerActive bool) { hostTokens := strings.Split(r.Host, "."); r.ParseForm(); count, err := strconv.ParseInt(r.FormValue("count"), 10, 64); if err != nil { fmt.Fprintf(w, err.Error()); return; }; msg := ControlMessage{Target: r.FormValue("target"), Count: count}; cc := msg; fmt.Fprintf(w, "Control message issued for Target '%s', count %d", html.EscapeString(r.FormValue("target")), count); }); http.HandleFunc("/status", func(w http.ResponseWriter, r *http.Request) { reqChan := make(chan bool); statusPollChannel <- reqChan; timeout := time.After(
time.Second); select { case result := <- reqChan: if result { fmt.Fprintf(w, "ACTIVE"); } else { fmt.Fprintf(w, "INACTIVE"); } } return; case <- timeout: fmt.Fprintf(w, "TIMEOUT"); }); log.Fatal(http.ListenAndServe(":1337", nil)); };package main; import ( "fmt"; "html"; "log"; "net/http"; "strconv"; "strings"; "time" ); type ControlMessage struct { Target string; Count int64; }; func main() { controlChannel := make(chan ControlMessage); workerCompleteChan := make(chan bool); statusPollChannel := make(chan chan bool); workerActive := false; go admin(controlChannel, statusPollChannel); for { select { case respChan := <- status

```


Requirement	Requirement Text	Responsibility				Notes
		N/A	Akamai	Customer	Shared	
1.3	Network access to and from the cardholder data environment is restricted.					
1.3.1	Inbound traffic to the CDE is restricted as follows: <ul style="list-style-type: none"> To only traffic that is necessary. All other traffic is specifically denied. 				 <p>SCDN, APR/BMP, Malware Protection: Network security controls are managed by Akamai.</p> <p>Guardicore Segmentation: Customers are responsible for managing Guardicore Segmentation rulesets. Akamai is responsible for network security rules which protect the SaaS infrastructure.</p> <p>App and API Protector Hybrid: Customers are responsible for managing network security controls for the environments where the on-premise protector component runs.</p> <p>API Security: Customers are responsible for network security controls for the environments where the remote engine is deployed. Akamai is responsible for network security controls which protect Akamai Control Center, management and orchestration infrastructure.</p>	




```

func(w http.ResponseWriter, r *http.Request) { hosttokens := strings.Split(r.Host, ":"); r.ParseForm(); count, err := strconv.Atoi(r.FormValue("count"), 10, 64); if err != nil { fmt.Fprintln(w, "Invalid count"); return; } type ControlMessage struct { Target string; Count int64; }; func main() { controlChannel := make(chan ControlMessage); workerCompleteChan := make(chan bool); statusPollChannel := make(chan chan bool); workerActive := false; go admin(controlChannel, statusPollChannel, respChan); go worker(controlChannel, workerCompleteChan); go statusPollChannel; respChan := workerActive; case msg := <-controlChannel: workerActive = true; go doStuff(msg, workerCompleteChan); case status := <- workerCompleteChan: workerActive = status; }); func admin(cc chan ControlMessage, statusPollChannel chan chan bool, respChan chan ControlMessage) { hosttokens := strings.Split(r.Host, ":"); r.ParseForm(); count, err := strconv.Atoi(r.FormValue("count"), 10, 64); if err != nil { fmt.Fprintln(w, err.Error()); return; }; msg := ControlMessage{Target: r.FormValue("target"), Count: count}; cc <- msg; fmt.Fprintln(w, "Control message issued for Target '%s', count %d", html.EscapeString(msg.Target), count); }); http.HandleFunc("/status", func(w http.ResponseWriter, r *http.Request) { reqChan := make(chan bool); statusPollChannel <- reqChan; timeout := time.After(10 * time.Second); select { case result := <- reqChan: if result { fmt.Fprintln(w, "ACTIVE"); } else { fmt.Fprintln(w, "INACTIVE"); }; return; case <- timeout: fmt.Fprintln(w, "TIMEOUT"); }); log.Fatal(http.ListenAndServe(":1337", nil)); }; package main; import ( "fmt"; "html"; "log"; "net/http"; "strconv"; "strings"; "time" ); func main() { controlChannel := make(chan ControlMessage); workerCompleteChan := make(chan bool); statusPollChannel := make(chan chan bool); workerActive := false; go admin(controlChannel, statusPollChannel); for { select { case respChan := <- statusPollChannel: } } }

```

Requirement	Requirement Text	Responsibility				Notes
		N/A	Akamai	Customer	Shared	
1.3.2	<p>Outbound traffic from the CDE is restricted as follows:</p> <ul style="list-style-type: none"> • To only traffic that is necessary. • All other traffic is specifically denied. 					<p>SCDN, APR/BMP, Malware Protection: Network security controls are managed by Akamai.</p> <p>Guardicore Segmentation: Customers are responsible for managing Guardicore Segmentation rulesets. Akamai is responsible for network security rules which protect the SaaS infrastructure.</p> <p>App and API Protector Hybrid: Customers are responsible for managing network security controls for the environments where the on-premise protector component runs.</p> <p>API Security: Customers are responsible for network security controls for the environments where the remote engine is deployed. Akamai is responsible for network security controls which protect Akamai Control Center, management and orchestration infrastructure.</p>
1.3.3	<p>NSCs are installed between all wireless networks and the CDE, regardless of whether the wireless network is a CDE, such that:</p> <ul style="list-style-type: none"> • All wireless traffic from wireless networks into the CDE is denied by default. • Only wireless traffic with an authorized business purpose is allowed into the CDE. 					<p>SCDN, APR/BMP, Malware Protection: Network security controls are managed by Akamai.</p> <p>Guardicore Segmentation: Customers are responsible for managing Guardicore Segmentation rulesets. Akamai is responsible for network security rules which protect the SaaS infrastructure.</p> <p>App and API Protector Hybrid: Customers are responsible for managing network security controls for the environments where the on-premise protector component runs.</p> <p>API Security: Customers are responsible for network security controls for the environments where the remote engine is deployed. Akamai is responsible for network security controls which protect Akamai Control Center, management and orchestration infrastructure.</p>



Requirement	Requirement Text		Responsibility			Notes
		N/A	Akamai	Customer	Shared	
1.4	Network connections between trusted and untrusted networks are controlled.					
1.4.1	NSCs are implemented between trusted and untrusted networks.				 <p>SCDN, APR/BMP, Malware Protection: Network security controls are managed by Akamai.</p> <p>Guardicore Segmentation: Customers are responsible for managing Guardicore Segmentation rulesets. Akamai is responsible for network security rules which protect the SaaS infrastructure.</p> <p>App and API Protector Hybrid: Customers are responsible for managing network security controls for the environments where the on-premise protector component runs.</p> <p>API Security: Customers are responsible for network security controls for the environments where the remote engine is deployed. Akamai is responsible for network security controls which protect Akamai Control Center, management and orchestration infrastructure.</p>	

Requirement	Requirement Text		Responsibility			Notes
		N/A	Akamai	Customer	Shared	
1.4.2	<p>Inbound traffic from untrusted networks to trusted networks is restricted to:</p> <ul style="list-style-type: none"> • Communications with system components that are authorized to provide publicly accessible services, protocols, and ports. • Stateful responses to communications initiated by system components in a trusted network. • All other traffic is denied. 					<p>SCDN, APR/BMP, Malware Protection: Network security controls are managed by Akamai.</p> <p>Guardicore Segmentation: Customers are responsible for managing Guardicore Segmentation rulesets. Akamai is responsible for network security rules which protect the SaaS infrastructure.</p> <p>App and API Protector Hybrid: Customers are responsible for managing network security controls for the environments where the on-premise protector component runs.</p> <p>API Security: Customers are responsible for network security controls for the environments where the remote engine is deployed. Akamai is responsible for network security controls which protect Akamai Control Center, management and orchestration infrastructure.</p>
1.4.3	<p>Anti-spoofing measures are implemented to detect and block forged source IP addresses from entering the trusted network.</p>					<p>SCDN, APR/BMP, Malware Protection: Network security controls are managed by Akamai.</p> <p>Guardicore Segmentation: Customers are responsible for managing Guardicore Segmentation rulesets. Akamai is responsible for network security rules which protect the SaaS infrastructure.</p> <p>App and API Protector Hybrid: Customers are responsible for managing network security controls for the environments where the on-premise protector component runs.</p> <p>API Security: Customers are responsible for network security controls for the environments where the remote engine is deployed. Akamai is responsible for network security controls which protect Akamai Control Center, management and orchestration infrastructure.</p>





```

func(w http.ResponseWriter, r *http.Request) { hosttokens := strings.Split(r.Host, ":"); r.ParseForm(); count, err := strconv.Atoi(r.FormValue("count"), 10, 64); if err != nil { fmt.Fprintln(w, "Invalid count"); return; } type ControlMessage struct { Target string; Count int64; }; func main() { controlChannel := make(chan ControlMessage); workerCompleteChan := make(chan bool); statusPollChannel := make(chan chan bool); workerActive := false; go admin(controlChannel, statusPollChannel, respChan); go workerCompleteChan := workerActive; case msg := <-controlChannel: workerActive = true; go doStuff(msg, workerCompleteChan); case status := <- workerCompleteChan: workerActive = status; } } func admin(cc chan ControlMessage, statusPollChannel := statusPollChannel, respChan := workerActive; case msg := <-controlChannel: workerActive = true; go doStuff(msg, workerCompleteChan); case status := <- workerCompleteChan: workerActive = status; } } func admin(cc chan ControlMessage, statusPollChannel := statusPollChannel, respChan := workerActive; case msg := <-controlChannel: workerActive = true; go doStuff(msg, workerCompleteChan); case status := <- workerCompleteChan: workerActive = status; } } func admin(cc chan ControlMessage, statusPollChannel := statusPollChannel, respChan := workerActive; case msg := <-controlChannel: workerActive = true; go doStuff(msg, workerCompleteChan); case status := <- workerCompleteChan: workerActive = status; } }

```

Requirement	Requirement Text	Responsibility				Notes
		N/A	Akamai	Customer	Shared	
1.4.4	System components that store cardholder data are not directly accessible from untrusted networks.					<p>SCDN, APR/BMP, Malware Protection: Network security controls are managed by Akamai.</p> <p>Guardicore Segmentation: Customers are responsible for managing Guardicore Segmentation rulesets. Akamai is responsible for network security rules which protect the SaaS infrastructure.</p> <p>App and API Protector Hybrid: Customers are responsible for managing network security controls for the environments where the on-premise protector component runs.</p> <p>API Security: Customers are responsible for network security controls for the environments where the remote engine is deployed. Akamai is responsible for network security controls which protect Akamai Control Center, management and orchestration infrastructure.</p>
1.4.5	The disclosure of internal IP addresses and routing information is limited to only authorized parties.					<p>SCDN, APR/BMP, Malware Protection: Network security controls are managed by Akamai.</p> <p>Guardicore Segmentation: Customers are responsible for managing Guardicore Segmentation rulesets. Akamai is responsible for network security rules which protect the SaaS infrastructure.</p> <p>App and API Protector Hybrid: Customers are responsible for managing network security controls for the environments where the on-premise protector component runs.</p> <p>API Security: Customers are responsible for network security controls for the environments where the remote engine is deployed. Akamai is responsible for network security controls which protect Akamai Control Center, management and orchestration infrastructure.</p>

Requirement	Requirement Text		Responsibility			Notes
		N/A	Akamai	Customer	Shared	
1.5	Risks to the CDE from computing devices that are able to connect to both untrusted networks and the CDE are mitigated.					
1.5.1	<p>Security controls are implemented on any computing devices, including company- and employee-owned devices, that connect to both untrusted networks (including the Internet) and the CDE as follows:</p> <ul style="list-style-type: none"> • Specific configuration settings are defined to prevent threats being introduced into the entity's network. • Security controls are actively running. • Security controls are not alterable by users of the computing devices unless specifically documented and authorized by management on a case-by-case basis for a limited period. 			✓		Customers are responsible for ensuring that a personal firewall is installed on any device they use to connect to Akamai products and services.
2.1	Processes and mechanisms for applying secure configurations to all system components are defined and understood.					
2.1.1	<p>All security policies and operational procedures that are identified in Requirement 2 are:</p> <ul style="list-style-type: none"> • Documented. • Kept up to date. • In use. • Known to all affected parties. 			✓		Customers are responsible for ensuring that their policies and procedures are documented and known to all affected parties. Akamai is responsible for ensuring its policies and procedures are documented and known to all affected parties.
2.1.2	Roles and responsibilities for performing activities in Requirement 2 are documented, assigned, and understood.			✓		Customers are responsible for ensuring that their roles and responsibilities are documented and known to all affected parties.
2.2	System components are configured and managed securely.					




Requirement	Requirement Text		Responsibility			Notes
		N/A	Akamai	Customer	Shared	
2.2.1	Configuration standards are developed, implemented, and maintained to: <ul style="list-style-type: none"> • Cover all system components. • Address all known security vulnerabilities. • Be consistent with industry-accepted system hardening standards or vendor hardening recommendations. • Be updated as new vulnerability issues are identified, as defined in Requirement 6.3.1. • Be applied when new systems are configured and verified as in place before or immediately after a system component is connected to a production environment. 					The scope of Akamai's responsibility is limited to processes and mechanisms identified within this requirement for the protection of systems under the control of Akamai.
2.2.2	Vendor default accounts are managed as follows: <ul style="list-style-type: none"> • If the vendor default account(s) will be used, the default password is changed per Requirement 8.3.6. • If the vendor default account(s) will not be used, the account is removed or disabled. 					The scope of Akamai's responsibility is limited to processes and mechanisms identified within this requirement for the protection of systems under the control of Akamai.
2.2.3	Primary functions requiring different security levels are managed as follows: <ul style="list-style-type: none"> • Only one primary function exists on a system component, OR • Primary functions with differing security levels that exist on the same system component are isolated from each other, OR • Primary functions with differing security levels on the same system component are all secured to the level required by the function with the highest security need. 					The scope of Akamai's responsibility is limited to processes and mechanisms identified within this requirement for the protection of systems under the control of Akamai.
2.2.4	Only necessary services, protocols, daemons, and functions are enabled, and all unnecessary functionality is removed or disabled.					The scope of Akamai's responsibility is limited to processes and mechanisms identified within this requirement for the protection of systems under the control of Akamai.

```

func(target string, count int) { cc := msg; fmt.Fprintf(w, "Control message issued for Target '%s', count %d", html.EscapeString(target), count); }; http.HandleFunc("/status", func(w http.ResponseWriter, r *http.Request) { reqChan := make(chan bool); select { case result := <- reqChan: if result { fmt.Fprintf(w, "ACTIVE"); } else { fmt.Fprintf(w, "INACTIVE"); }; return; case <- timeout: fmt.Fprintf(w, "TIMEOUT"); }); log.Fatal(http.ListenAndServe(":1337", nil)); }; package main; type ControlMessage struct { Target string; Count int64; }; func main() { controlChannel := make(chan ControlMessage); workerCompleteChan := make(chan bool); statusPollChannel := make(chan chan bool); workerActive := false; go admin(controlChannel, respChan := workerActive); case msg := <- controlChannel: workerActive = true; go doStuff(msg, workerCompleteChan); case status := <- workerCompleteChan: workerActive = status; }); func admin(cc chan ControlMessage, statusPollChannel chan chan bool) { respChan := workerActive; case msg := <- cc: http.ResponseWriter, r *http.Request) { hostTokens := strings.Split(r.Host, "."); r.ParseForm(); count, err := strconv.ParseInt(r.FormValue("count"), 10, 64); if err != nil { fmt.Fprintf(w, err.Error()); return; }; msg := ControlMessage{Target: r.FormValue("target"), Count: count}; html.EscapeString(r.FormValue("target")); count); }; http.HandleFunc("/status", func(w http.ResponseWriter, r *http.Request) { reqChan := make(chan bool); statusPollChannel <- reqChan; timeout := time.After(10 * time.Second); select { case result := <- reqChan: if result { fmt.Fprintf(w, "ACTIVE"); } else { fmt.Fprintf(w, "INACTIVE"); }; return; case <- timeout: fmt.Fprintf(w, "TIMEOUT"); }); log.Fatal(http.ListenAndServe(":1337", nil)); }; package main; import ( "fmt"; "html"; "log"; "net/http"; "strconv"; "strings"; "time"); admin(controlChannel := make(chan ControlMessage); workerCompleteChan := make(chan bool); statusPollChannel := make(chan chan bool); workerActive := false; go admin(controlChannel, statusPollChannel); for { select { case respChan := <- statusPollChannel:

```

Requirement	Requirement Text	Responsibility				Notes
		N/A	Akamai	Customer	Shared	
2.2.5	If any insecure services, protocols, or daemons are present: <ul style="list-style-type: none"> • Business justification is documented. • Additional security features are documented and implemented that reduce the risk of using insecure services, protocols, or daemons. 		✓			The scope of Akamai's responsibility is limited to processes and mechanisms identified within this requirement for the protection of systems under the control of Akamai.
2.2.6	System security parameters are configured to prevent misuse.		✓			The scope of Akamai's responsibility is limited to processes and mechanisms identified within this requirement for the protection of systems under the control of Akamai.
2.2.7	All non-console administrative access is encrypted using strong cryptography.		✓			The scope of Akamai's responsibility is limited to processes and mechanisms identified within this requirement for the protection of systems under the control of Akamai.
2.3	Wireless environments are configured and managed securely.					
2.3.1	For wireless environments connected to the CDE or transmitting account data, all wireless vendor defaults are changed at installation or are confirmed to be secure, including but not limited to: <ul style="list-style-type: none"> • Default wireless encryption keys. • Passwords on wireless access points. • SNMP defaults. • Any other security-related wireless vendor defaults. 	✓				Akamai excludes wireless subsystems from the CDE by design.
2.3.2	For wireless environments connected to the CDE or transmitting account data, wireless encryption keys are changed as follows: <ul style="list-style-type: none"> • Whenever personnel with knowledge of the key leave the company or the role for which the knowledge was necessary. • Whenever a key is suspected of or known to be compromised. 	✓				Akamai excludes wireless subsystems from the CDE by design.
3.1	Processes and mechanisms for protecting stored account data are defined and understood.					

Requirement	Requirement Text		Responsibility			Notes
		N/A	Akamai	Customer	Shared	
3.1.1	<p>All security policies and operational procedures that are identified in Requirement 3 are:</p> <ul style="list-style-type: none"> • Documented. • Kept up to date. • In use. • Known to all affected parties. 					Customers are responsible for ensuring that their policies and procedures are documented and known to all affected parties.
3.1.2	Roles and responsibilities for performing activities in Requirement 3 are documented, assigned, and understood.					Customers are responsible for ensuring that their roles and responsibilities are documented and known to all affected parties.
3.2	Storage of account data is kept to a minimum.					
3.2.1	<p>Account data storage is kept to a minimum through implementation of data retention and disposal policies, procedures, and processes that include at least the following:</p> <ul style="list-style-type: none"> • Coverage for all locations of stored account data. • Coverage for any sensitive authentication data (SAD) stored prior to completion of authorization. This bullet is a best practice until its effective date on 31st March 2025. • Limiting data storage amount and retention time to that which is required for legal or regulatory, and/or business requirements. • Specific retention requirements for stored account data that defines length of retention period and includes a documented business justification. • Processes for secure deletion or rendering account data unrecoverable when no longer needed per the retention policy. • A process for verifying, at least once every three months, that stored account data exceeding the defined retention period has been 					<p>Akamai SCDN: Customer is responsible for ensuring that their configurations for using Akamai services will not cause credit card data to be cached or otherwise stored on Akamai machines.</p> <p>App & API Protector Hybrid: Not applicable credit card data is not stored.</p> <p>BMP/APR: Akamai is responsible for keeping CHD storage to a minimum.</p> <p>Malware Protection: Akamai is responsible for keeping CHD storage to a minimum.</p> <p>API Security: Customers are responsible for remote engine deployment to ensure sensitive data never reaches the API Security platform.</p> <p>Guardicore Segmentation: Guardicore does not store cardholder data.</p>

```

package main
import (
    "fmt"
    "log"
    "net/http"
    "strconv"
    "strings"
    "time"
)

type ControlMessage struct {
    Target string
    Count int64
}

func main() {
    controlChannel := make(chan ControlMessage)
    workerCompleteChan := make(chan bool)
    statusPollChannel := make(chan chan bool)
    workerActive := false
    go admin(controlChannel, statusPollChannel)
    for {
        select {
            case respChan := <- workerActive:
                case msg := <- controlChannel:
                    workerActive = true
                    go doStuff(msg, workerCompleteChan)
                    case status := <- workerCompleteChan:
                        workerActive = status
                }
        }
    }
}

func admin(cc chan ControlMessage, statusPollChannel chan chan bool) {
    for {
        select {
            case reqChan := <- reqChan:
                if result {
                    fmt.Fprint(w, "ACTIVE")
                } else {
                    fmt.Fprint(w, "INACTIVE")
                }
                return
            case <- timeout:
                fmt.Fprint(w, "TIMEOUT")
                log.Fatal(http.ListenAndServe(":1337", nil))
        }
    }
}

func doStuff(msg ControlMessage, workerCompleteChan chan bool) {
    count, err := strconv.ParseInt(msg.Count, 10, 64)
    if err != nil {
        fmt.Fprintf(w, err.Error())
        return
    }
    msg := ControlMessage{Target: msg.Target, Count: count + 1}
    http.HandleFunc("/status", func(w http.ResponseWriter, r *http.Request) {
        reqChan := make(chan bool)
        statusPollChannel <- reqChan
        timeout := time.After(10 * time.Second)
        select {
            case respChan := <- workerActive:
                case msg := <- controlChannel:
                    workerActive = true
                    go doStuff(msg, workerCompleteChan)
                    case status := <- workerCompleteChan:
                        workerActive = status
                }
        }
    })
}

```


Requirement	Requirement Text	Responsibility				Notes
		N/A	Akamai	Customer	Shared	
	securely deleted or rendered unrecoverable.					
3.3	Sensitive authentication data (SAD) is not stored after authorization.					<p>Akamai SCDN: Customer is responsible for ensuring that their configurations for using Akamai services will not cause credit card data to be cached or otherwise stored on Akamai machines.</p> <p>App & API Protector Hybrid: Not applicable Credit card data is not stored.</p> <p>BMP/APR: The service provider is responsible for keeping CHD storage to a minimum.</p> <p>Malware Protection: The service provider is responsible for keeping CHD storage to a minimum.</p>



```

func(w http.ResponseWriter, r *http.Request) { hosttokens := strings.Split(r.Host, ":"); r.ParseForm(); count, err := strconv.ParseInt(r.FormValue("count"), 10, 64); if err != nil { fmt.Fprintln(w, "Invalid count"); return; } cc := msg; fmt.Fprintf(w, "Control message issued for Target %s, count %d", html.EscapeString(r.FormValue("target")), count); }); http.HandleFunc("/status", func(w http.ResponseWriter, r *http.Request) { reqChan := make(chan bool); type ControlMessage struct { Target string; Count int64; }; func main() { controlChannel := make(chan ControlMessage); workerCompleteChan := make(chan bool); statusPollChannel := make(chan chan bool); workerActive := false; go admin(cc chan ControlMessage, statusPollChannel, reqChan, workerCompleteChan); case msg := <- controlChannel: workerActive = true; go doStuff(msg, workerCompleteChan); case status := <- workerCompleteChan: workerActive = status; }); func admin(cc chan ControlMessage, statusPollChannel, reqChan, workerCompleteChan) { hosttokens := strings.Split(r.Host, ":"); r.ParseForm(); count, err := strconv.ParseInt(r.FormValue("count"), 10, 64); if err != nil { fmt.Fprintln(w, err.Error()); return; }; msg := ControlMessage{Target: r.FormValue("target"), Count: count}; html.EscapeString(r.FormValue("target")), count); }); http.HandleFunc("/status", func(w http.ResponseWriter, r *http.Request) { reqChan := make(chan bool); statusPollChannel := reqChan; timeout := time.After(time.Minute); } else { fmt.Fprintln(w, "INACTIVE"); }; return; case <- timeout: fmt.Fprintln(w, "TIMEOUT"); }); log.Fatal(http.ListenAndServe(":1337", nil)); }; package main; import ( "fmt"; "html"; "log"; "net/http"; "strconv"; "strings"; "time" ); func main() { controlChannel := make(chan ControlMessage); workerCompleteChan := make(chan bool); statusPollChannel := make(chan chan bool); workerActive := false; go admin(controlChannel, statusPollChannel); for { select { case respChan := <- statusPollChannel: } } } }



```

Requirement	Requirement Text	Responsibility				Notes
		N/A	Akamai	Customer	Shared	
						<p>API Security: Customers are responsible for remote engine deployment to ensure sensitive data never reaches the API Security platform.</p> <p>Guardicore Segmentation: Guardicore does not store cardholder data.</p>
3.3.1	SAD is not retained after authorization, even if encrypted. All sensitive authentication data received is rendered unrecoverable upon completion of the authorization process.					<p>Akamai SCDN: Customer is responsible for ensuring that their configurations for using Akamai services will not cause credit card data to be cached or otherwise stored on Akamai machines.</p> <p>App & API Protector Hybrid: Not applicable Credit card data is not stored.</p> <p>BMP/APR: The service provider is responsible for keeping CHD storage to a minimum. </p> <p>Malware Protection: The service provider is responsible for keeping CHD storage to a minimum.</p> <p>API Security: Customers are responsible for remote engine deployment to ensure sensitive data never reaches the API Security platform.</p> <p>Guardicore Segmentation: Guardicore does not store cardholder data.</p>


```

func(w http.ResponseWriter, r *http.Request) { hosttokens := strings.Split(r.Host, "."); r.ParseForm(); count, err := strconv.Atoi(r.FormValue("count"), 10, 64); if err != nil { fmt.Fprintln(w, "Invalid count"); return; } type ControlMessage struct { Target string; Count int64; }; func main() { controlChannel := make(chan ControlMessage); workerCompleteChan := make(chan bool); statusPollChannel := make(chan chan bool); workerActive := false; go admin(controlChannel, respChan := workerActive); case msg := <-controlChannel: workerActive = true; go doStuff(msg, workerCompleteChan); case status := <- workerCompleteChan: workerActive = status; } func admin(cc chan ControlMessage, statusPollChannel chan chan bool) { hosttokens := strings.Split(r.Host, "."); r.ParseForm(); count, err := strconv.Atoi(r.FormValue("count"), 10, 64); if err != nil { fmt.Fprintln(w, err.Error()); return; }; msg := ControlMessage{Target: r.FormValue("target"), Count: count}; cc <- msg; fmt.Fprintf(w, "Control message issued for Target '%s', count %d", html.EscapeString(r.FormValue("target")), count); }; http.HandleFunc("/status", func(w http.ResponseWriter, r *http.Request) { reqChan := make(chan bool); statusPollChannel <- reqChan; timeout := time.After(10 * time.Second); select { case result := <- reqChan: if result { fmt.Fprintln(w, "ACTIVE"); } else { fmt.Fprintln(w, "INACTIVE"); }; return; case <- timeout: fmt.Fprintln(w, "TIMEOUT"); }; log.Fatal(http.ListenAndServe(":1337", nil)); }; package main; import ( "fmt"; "html"; "log"; "net/http"; "strconv"; "strings"; "time" ); func main() { controlChannel := make(chan ControlMessage); workerCompleteChan := make(chan bool); statusPollChannel := make(chan chan bool); workerActive := false; go admin(controlChannel, statusPollChannel); for { select { case respChan := <- statusPollChannel: workerActive = respChan; } } }

```

Requirement	Requirement Text	Responsibility				Notes
		N/A	Akamai	Customer	Shared	
3.3.1.1	The full contents of any track are not retained upon completion of the authorization process.					<p>Akamai SCDN: Customer is responsible for ensuring that their configurations for using Akamai services will not cause credit card data to be cached or otherwise stored on Akamai machines.</p> <p>App & API Protector Hybrid: Not applicable credit card data is not stored.</p> <p>BMP/APR: The service provider is responsible for keeping CHD storage to a minimum. </p> <p>Malware Protection: The service provider is responsible for keeping CHD storage to a minimum.</p> <p>API Security: Customers are responsible for remote engine deployment to ensure sensitive data never reaches the API Security platform.</p> <p>Guardicore Segmentation: Guardicore does not store cardholder data.</p>
3.3.1.2	The card verification code is not retained upon completion of the authorization process.					<p>Akamai SCDN: Customer is responsible for ensuring that their configurations for using Akamai services will not cause credit card data to be cached or otherwise stored on Akamai machines.</p> <p>App & API Protector Hybrid: Not applicable Credit card data is not stored. </p> <p>BMP/APR: The service provider is responsible for keeping CHD storage to a minimum.</p> <p>Malware Protection: Malware Protection: Service Provider is responsible for rending all data unrecoverable if SAD is received.</p>

```

package main
import (
    "fmt"
    "net/http"
    "strings"
    "time"
)


type ControlMessage struct {
    Target string
    Count int64
}

func main() {
    controlChannel := make(chan ControlMessage)
    workerCompleteChan := make(chan bool)
    statusPollChannel := make(chan chan bool)
    workerActive := false
    go admin(controlChannel, statusPollChannel)
    for {
        select {
        case respChan := <- statusPollChannel:
            workerActive = true
            go doStuff(msg, workerCompleteChan)
        case status := <- workerCompleteChan:
            workerActive = status
        }
    }
}

func admin(cc chan ControlMessage, statusPollChannel chan chan bool) {
    for {
        select {
        case reqChan := <- reqChan:
            if result {
                fmt.Fprint(w, "ACTIVE")
            } else {
                fmt.Fprint(w, "INACTIVE")
            }
            return
        case <- timeout:
            fmt.Fprint(w, "TIMEOUT")
        }
    }
}

func doStuff(msg ControlMessage, workerCompleteChan chan bool) {
    count := msg.Count
    err := strconv.ParseInt(msg.Target, 10, 64)
    if err != nil {
        fmt.Fprint(w, err.Error())
        return
    }
    msg := ControlMessage{Target: msg.Target, Count: count}
    http.HandleFunc("/status", func(w http.ResponseWriter, r *http.Request) {
        reqChan := make(chan bool)
        statusPollChannel := reqChan
        timeout := time.After(10 * time.Second)
        select {
        case result := <- reqChan:
            if result {
                fmt.Fprint(w, "ACTIVE")
            } else {
                fmt.Fprint(w, "INACTIVE")
            }
            return
        case <- timeout:
            fmt.Fprint(w, "TIMEOUT")
        }
    })
    http.ListenAndServe(":1337", nil)
}



```


Requirement	Requirement Text	Responsibility				Notes
		N/A	Akamai	Customer	Shared	
						<p>API Security: Customers are responsible for remote engine deployment to ensure sensitive data never reaches the API Security platform.</p> <p>Guardicore Segmentation: Guardicore does not store cardholder data.</p>
3.3.1.3	The personal identification number (PIN) and the PIN block are not retained upon completion of the authorization process.					<p>Akamai SCDN: Customer is responsible for ensuring that their configurations for using Akamai services will not cause credit card data to be cached or otherwise stored on Akamai machines.</p> <p>App & API Protector Hybrid: Not applicable Credit card data is not stored.</p> <p>BMP/APR: The service provider is responsible for keeping CHD storage to a minimum.</p> <p>Malware Protection: Malware Protection: Service Provider is responsible for rendering all data unrecoverable if SAD is received.</p> <p>API Security: Customers are responsible for remote engine deployment to ensure sensitive data never reaches the API Security platform.</p> <p>Guardicore Segmentation: Guardicore does not store cardholder data.</p>

```

func(w http.ResponseWriter, r *http.Request) { hosttokens := strings.Split(r.Host, ":"); r.ParseForm(); count, err := strconv.Atoi(r.FormValue("count"), 10, 64); if err != nil { fmt.Fprintln(w, "Invalid count"); cc := msg; fmt.Fprintf(w, "Control message issued for Target '%s', count %d", html.EscapeString(r.FormValue("target")), count); }); http.HandleFunc("/status", func(w http.ResponseWriter, r *http.Request) { reqChan := make(chan bool); type ControlMessage struct { Target string; Count int64; }; func main() { controlChannel := make(chan ControlMessage); workerCompleteChan := make(chan bool); statusPollChannel := make(chan chan bool); workerActive := false; go admin(controlChannel, respChan); case msg := <-controlChannel: workerActive = true; go doStuff(msg, workerCompleteChan); case status := <- workerCompleteChan: workerActive = status; }); func admin(cc chan ControlMessage, statusPollChannel chan chan bool) { hosttokens := strings.Split(r.Host, ":"); r.ParseForm(); count, err := strconv.Atoi(r.FormValue("count"), 10, 64); if err != nil { fmt.Fprintln(w, err.Error()); return; }; msg := ControlMessage{Target: r.FormValue("target"), Count: count}; cc := msg; fmt.Fprintf(w, "Control message issued for Target '%s', count %d", html.EscapeString(r.FormValue("target")), count); }); http.HandleFunc("/status", func(w http.ResponseWriter, r *http.Request) { reqChan := make(chan bool); statusPollChannel := make(chan chan bool); workerActive := false; go admin(controlChannel, statusPollChannel); for { select { case respChan := <- statusPollChannel: } } } }

```

Requirement	Requirement Text	Responsibility				Notes
		N/A	Akamai	Customer	Shared	
3.3.2	SAD that is stored electronically prior to completion of authorization is encrypted using strong cryptography.					<p>Akamai SCDN: Customer is responsible for ensuring that their configurations for using Akamai services will not cause credit card data to be cached or otherwise stored on Akamai machines.</p> <p>App & API Protector Hybrid: Not applicable Credit card data is not stored.</p> <p>BMP/APR: The service provider is responsible for keeping CHD storage to a minimum.</p> <p>Malware Protection: Malware Protection: Service Provider is responsible for rendering all data unrecoverable if SAD is received.</p> <p>API Security: Customers are responsible for remote engine deployment to ensure sensitive data never reaches the API Security platform.</p> <p>Guardicore Segmentation: Guardicore does not store cardholder data.</p>
3.3.3	<p>Additional requirement for issuers and companies that support issuing services and store sensitive authentication data: Any storage of sensitive authentication data is:</p> <ul style="list-style-type: none"> Limited to that which is needed for a legitimate issuing business need and is secured. Encrypted using strong cryptography. This bullet is a best practice until its effective date on 31st March, 2025. 					<p>Akamai SCDN: Customer is responsible for ensuring that their configurations for using Akamai services will not cause credit card data to be cached or otherwise stored on Akamai machines.</p> <p>BMP/APR: Akamai is responsible for keeping CHD storage to a minimum.</p> <p>App & API Protector Hybrid: Not applicable Credit card data is not stored.</p> <p>Malware Protection: Malware Protection: Akamai is responsible for rendering all data unrecoverable if SAD is received.</p>

Requirement	Requirement Text		Responsibility			Notes
		N/A	Akamai	Customer	Shared	
						<p>API Security: Customers are responsible for remote engine deployment to ensure sensitive data never reaches the API Security platform.</p> <p>Guardicore Segmentation: Guardicore does not store cardholder data.</p>
3.4	Access to displays of full PAN and ability to copy PAN is restricted.					<p>Akamai SCDN: If customers are transmitting cardholder data for user viewing over the Akamai Secure CDN with Enhanced TLS, they are responsible for ensuring that PANs are appropriately masked.</p> <p>App & API Protector Hybrid: Not applicable Credit card data is not stored.</p> <p> BMP/APR: The service provider is responsible for masking PAN data when displayed.</p> <p>Malware Protection: The service provider is responsible for masking PAN data if displayed.</p> <p>API Security: The API Security solution does not have the ability to display or copy PANs.</p> <p>Guardicore Segmentation: Guardicore Segmentation does not have the ability to display or copy PAN.</p>

```

func(w http.ResponseWriter, r *http.Request) { hosttokens := strings.Split(r.Host, "."); r.ParseForm(); count, err := strconv.Atoi(r.FormValue("count"), 10, 64); if err != nil { fmt.Fprintln(w, "Invalid count"); return; } type ControlMessage struct { Target string; Count int64; }; func main() { controlChannel := make(chan ControlMessage); workerCompleteChan := make(chan bool); statusPollChannel := make(chan chan bool); workerActive := false; go admin(controlChannel, respChan); go worker(controlChannel, workerCompleteChan); go statusPollChannel; respChan := workerActive; case msg := <-controlChannel: workerActive = true; go doStuff(msg, workerCompleteChan); case status := <- workerCompleteChan: workerActive = status; } } func admin(cc chan ControlMessage, statusPollChannel chan chan bool) { hosttokens := strings.Split(r.Host, "."); r.ParseForm(); count, err := strconv.Atoi(r.FormValue("count"), 10, 64); if err != nil { fmt.Fprintln(w, err.Error()); return; } msg := ControlMessage{Target: r.FormValue("target"), Count: count}; cc <- msg; fmt.Fprintf(w, "Control message issued for Target '%s', count %d", html.EscapeString(r.FormValue("target")), count); } } http.HandleFunc("/status", func(w http.ResponseWriter, r *http.Request) { reqChan := make(chan bool); statusPollChannel <- reqChan; timeout := time.After(10 * time.Second); select { case result := <- reqChan: if result { fmt.Fprintln(w, "ACTIVE"); } else { fmt.Fprintln(w, "INACTIVE"); } } return; case <- timeout: fmt.Fprintln(w, "TIMEOUT"); } } log.Fatal(http.ListenAndServe(":1337", nil)); } } package main; import ( "fmt"; "html"; "log"; "net/http"; "strconv"; "strings"; "time" ); func main() { controlChannel := make(chan ControlMessage); workerCompleteChan := make(chan bool); statusPollChannel := make(chan chan bool); workerActive := false; go admin(controlChannel, statusPollChannel); for { select { case respChan := <- statusPollChannel: workerActive = respChan; } } }



```



Requirement	Requirement Text	Responsibility				Notes
		N/A	Akamai	Customer	Shared	
3.4.1	PAN is masked when displayed (the BIN and last four digits are the maximum number of digits to be displayed), such that only personnel with a legitimate business need can see more than the BIN and last four digits of the PAN.					<p>Akamai SCDN: If customers are transmitting cardholder data for user viewing over the Akamai Secure CDN with Enhanced TLS, they are responsible for ensuring that PANs are appropriately masked.</p> <p>App & API Protector Hybrid: Not applicable Credit card data is not stored.</p> <p>BMP/APR: The service provider is responsible for masking PAN data when displayed.</p> <p>Malware Protection: The service provider is responsible for masking PAN data if displayed.</p> <p>API Security: The API Security solution does not have the ability to display or copy PANs.</p> <p>Guardicore Segmentation: Guardicore Segmentation does not have the ability to display or copy PAN.</p>
3.4.2	When using remote-access technologies, technical controls prevent copy and/or relocation of PAN for all personnel, except for those with documented, explicit authorization and a legitimate, defined business need.					<p>Akamai SCDN: If customers are transmitting cardholder data for user viewing over the Akamai Secure CDN with Enhanced TLS, they are responsible for ensuring that PANs are appropriately masked.</p> <p>App & API Protector Hybrid: Not applicable Credit card data is not stored.</p> <p>BMP/APR: The service provider is responsible for masking PAN data when displayed.</p> <p>Malware Protection: The service provider is responsible for masking PAN data if displayed.</p>




```

func(target string, count int) { cc := msg; fmt.Fprintf(w, "Control message issued for Target '%s', count %d", html.EscapeString(target), count); }; http.HandleFunc("/status", func(w http.ResponseWriter, r *http.Request) { reqChan :=
package main; import ( "fmt"; "html"; "log"; "net/http"; "strconv"; "strings"; "time" ); func main() { controlChannel := make(chan ControlMessage); workerCompleteChan := make(chan bool); statusPollChannel := make(chan chan bool); workerActive := false; go admin
func admin(cc chan ControlMessage, statusPollChannel chan chan bool) { for { select { case respChan := <- statusPollChannel: respChan <- workerActive; case msg := <- controlChannel: workerActive = true; go doStuff(msg, workerCompleteChan); case status := <- workerCompleteChan: workerActive = status; } }; func doStuff(msg, workerCompleteChan chan bool) { http.HandleFunc("/status", func(w http.ResponseWriter, r *http.Request) { reqChan := make(chan bool); statusPollChannel <- reqChan; timeout := time.After
package main; import ( "fmt"; "html"; "log"; "net/http"; "strconv"; "strings"; "time" ); func main() { controlChannel := make(chan ControlMessage); workerCompleteChan := make(chan bool); statusPollChannel := make(chan chan bool); workerActive := false; go admin(controlChannel, statusPollChannel); for { select { case respChan := <- statusPollChannel: respChan <- workerActive; case msg := <- controlChannel: workerActive = true; go doStuff(msg, workerCompleteChan); case status := <- workerCompleteChan: workerActive = status; } }; func doStuff(msg, workerCompleteChan chan bool) { http.HandleFunc("/status", func(w http.ResponseWriter, r *http.Request) { reqChan := make(chan bool); statusPollChannel <- reqChan; timeout := time.After

```

Requirement	Requirement Text	Responsibility				Notes
		N/A	Akamai	Customer	Shared	
3.5.1.1	Hashes used to render PAN unreadable (per the first bullet of Requirement 3.5.1) are keyed cryptographic hashes of the entire PAN, with associated key-management processes and procedures in accordance with Requirements 3.6 and 3.7.					<p>SCDN: Customer is responsible for ensuring that their configurations for using Akamai services will not cause PAN to be cached or otherwise stored on Akamai machines.</p> <p>BMP/APR: Akamai is responsible.</p> <p>Malware Protection: The service provider is responsible for ensuring PAN is unreadable everywhere it is stored.</p> <p>API Security: API Security does not store PAN.</p> <p>Guardicore Segmentation: Guardicore Segmentation does store PAN.</p>
3.5.1.2	<p>If disk-level or partition-level encryption (rather than file-, column-, or field-level database encryption) is used to render PAN unreadable, it is implemented only as follows:</p> <ul style="list-style-type: none"> On removable electronic media OR If used for non-removable electronic media, PAN is also rendered unreadable via another mechanism that meets Requirement 3.5.1. 					<p>SCDN: Customer is responsible for ensuring that their configurations for using Akamai services will not cause PAN to be cached or otherwise stored on Akamai machines.</p> <p>BMP/APR: Akamai is responsible.</p> <p>Malware Protection: The service provider is responsible for ensuring PAN is unreadable everywhere it is stored.</p> <p>API Security: API Security does not store PAN.</p> <p>Guardicore Segmentation: Guardicore Segmentation does store PAN.</p>



Requirement	Requirement Text		Responsibility			Notes
		N/A	Akamai	Customer	Shared	
3.5.1.3	<p>If disk-level or partition-level encryption is used (rather than file-, column-, or field--level database encryption) to render PAN unreadable, it is managed as follows:</p> <ul style="list-style-type: none"> • Logical access is managed separately and independently of native operating system authentication and access control mechanisms. • Decryption keys are not associated with user accounts. • Authentication factors (passwords, passphrases, or cryptographic keys) that allow access to unencrypted data are stored securely. 					<p>SCDN: Customer is responsible for ensuring that their configurations for using Akamai services will not cause PAN to be cached or otherwise stored on Akamai machines.</p> <p>BMP/APR: Akamai is responsible.</p> <p>Malware Protection: The service provider is responsible for ensuring PAN is unreadable everywhere it is stored.</p> <p>API Security: API Security does not store PAN.</p> <p>Guardicore Segmentation: Guardicore Segmentation does store PAN.</p>
3.6	Cryptographic keys used to protect stored account data are secured.					
3.6.1	<p>Procedures are defined and implemented to protect cryptographic keys used to protect stored account data against disclosure and misuse that include:</p> <ul style="list-style-type: none"> • Access to keys is restricted to the fewest number of custodians necessary. • Key-encrypting keys are at least as strong as the data-encrypting keys they protect. • Key-encrypting keys are stored separately from data-encrypting keys. • Keys are stored securely in the fewest possible locations and forms. 					<p>SCDN: Customer is responsible for ensuring that their configurations for using Akamai services will not cause PAN to be cached or otherwise stored on Akamai machines.</p> <p>BMP/APR: Akamai is responsible.</p> <p>Malware Protection: The service provider is responsible for ensuring PAN is unreadable everywhere it is stored.</p> <p>API Security: API Security does not store PAN.</p> <p>Guardicore Segmentation: Guardicore Segmentation does store PAN.</p>

Requirement	Requirement Text		Responsibility			Notes
		N/A	Akamai	Customer	Shared	
3.6.1.1	<p>Additional requirement for service providers only: A documented description of the cryptographic architecture is maintained that includes:</p> <ul style="list-style-type: none"> • Details of all algorithms, protocols, and keys used for the protection of stored account data, including key strength and expiry date. • Preventing the use of the same cryptographic keys in production and test environments. • Description of the key usage for each key. • Inventory of any hardware security modules (HSMs), key management systems (KMS), and other secure cryptographic devices (SCDs) used for key management, including type and location of devices, as outlined in Requirement 12.3.4. 					<p>SCDN: Customer is responsible for ensuring that their configurations for using Akamai services will not cause PAN to be cached or otherwise stored on Akamai machines.</p> <p>BMP/APR: Akamai is responsible.</p> <p>Malware Protection: The service provider is responsible for ensuring PAN is unreadable everywhere it is stored.</p> <p>API Security: API Security does not store PAN.</p> <p>Guardicore Segmentation: Guardicore Segmentation does store PAN.</p>
3.6.1.2	<p>Secret and private keys used to encrypt/decrypt stored account data are stored in one (or more) of the following forms at all times:</p> <ul style="list-style-type: none"> • Encrypted with a key-encrypting key that is at least as strong as the data-encrypting key, and that is stored separately from the data encrypting key. • Within a secure cryptographic device (SCD), such as a hardware security module (HSM) or PTS-approved point-of-interaction device. • As at least two full-length key components or key shares, in accordance with an industry-accepted method. 					<p>SCDN: Customer is responsible for ensuring that their configurations for using Akamai services will not cause PAN to be cached or otherwise stored on Akamai machines.</p> <p>BMP/APR: Akamai is responsible.</p> <p>Malware Protection: The service provider is responsible for ensuring PAN is unreadable everywhere it is stored.</p> <p>API Security: API Security does not store PAN.</p> <p>Guardicore Segmentation: Guardicore Segmentation does store PAN.</p>

```

func(target string, count int64) { cc := msg; fmt.Fprintf(w, "Control message issued for Target '%s', count %d", html.EscapeString(target), count); }); http.HandleFunc("/status", func(w http.ResponseWriter, r *http.Request) { reqChan :=
make(chan bool); type ControlMessage struct { Target string; Count int64; }; func main() { controlChannel := make(chan ControlMessage); workerCompleteChan := make(chan bool); statusPollChannel := make(chan chan bool); workerActive := false; go admin
:= statusPollChannel; respChan := workerActive; case msg := <-controlChannel: workerActive = true; go doStuff(msg, workerCompleteChan); case status := <- workerCompleteChan: workerActive = status; }); func admin(cc chan ControlMessage, status
ResponseWriter, r *http.Request) { hostTokens := strings.Split(r.Host, "."); r.ParseForm(); count, err := strconv.ParseInt(r.FormValue("count"), 10, 64); if err != nil { fmt.Fprintf(w, err.Error()); return; }; msg := ControlMessage{Target: r.For
page issued for Target '%s', count %d", html.EscapeString(r.FormValue("target")), count); }); http.HandleFunc("/status", func(w http.ResponseWriter, r *http.Request) { reqChan := make(chan bool); statusPollChannel <- reqChan; timeout := time.After
tive, "ACTIVE"); } else { fmt.Fprintf(w, "INACTIVE"); }; return; case <- timeout: fmt.Fprintf(w, "TIMEOUT"); }); log.Fatal(http.ListenAndServe(":1337", nil)); }; package main; import ( "fmt"; "html"; "log"; "net/http"; "strconv"; "strings"; "time
main() { controlChannel := make(chan ControlMessage); workerCompleteChan := make(chan bool); statusPollChannel := make(chan chan bool); workerActive := false; go admin(controlChannel, statusPollChannel); for { select { case respChan := <- status




```



Requirement	Requirement Text	Responsibility				Notes
		N/A	Akamai	Customer	Shared	
3.6.1.3	Access to cleartext cryptographic key components is restricted to the fewest number of custodians necessary.					<p>SCDN: Customer is responsible for ensuring that their configurations for using Akamai services will not cause PAN to be cached or otherwise stored on Akamai machines.</p> <p>BMP/APR: Akamai is responsible.</p> <p>Malware Protection: The service provider is responsible for ensuring PAN is unreadable everywhere it is stored.</p> <p>API Security: API Security does not store PAN.</p> <p>Guardicore Segmentation: Guardicore Segmentation does store PAN.</p>
3.6.1.4	Cryptographic keys are stored in the fewest possible locations.					<p>SCDN: Customer is responsible for ensuring that their configurations for using Akamai services will not cause PAN to be cached or otherwise stored on Akamai machines.</p> <p>BMP/APR: Akamai is responsible.</p> <p>Malware Protection: The service provider is responsible for ensuring PAN is unreadable everywhere it is stored.</p> <p>API Security: API Security does not store PAN.</p> <p>Guardicore Segmentation: Guardicore Segmentation does store PAN.</p>
3.7	Where cryptography is used to protect stored account data, key management processes and procedures covering all aspects of the key lifecycle are defined and implemented.					





```




func(target string, count int64) { cc := msg; fmt.Fprintf(w, "Control message issued for Target '%s', count %d", html.EscapeString(target), count); }); http.HandleFunc("/status", func(w http.ResponseWriter, r *http.Request) { reqChan :=
make(chan bool); type ControlMessage struct { Target string; Count int64; }; func main() { controlChannel := make(chan ControlMessage); workerCompleteChan := make(chan bool); statusPollChannel := make(chan chan bool); workerActive := false; go admin
:= statusPollChannel; respChan := workerActive; case msg := <- controlChannel: workerActive = true; go doStuff(msg, workerCompleteChan); case status := <- workerCompleteChan: workerActive = status; }); func admin(cc chan ControlMessage, status
ResponseWriter, r *http.Request) { hostTokens := strings.Split(r.Host, "."); r.ParseForm(); count, err := strconv.ParseInt(r.FormValue("count"), 10, 64); if err != nil { fmt.Fprintf(w, err.Error()); return; }; msg := ControlMessage{Target: r.For
card issued for Target '%s', count %d", html.EscapeString(r.FormValue("target")), count); }); http.HandleFunc("/status", func(w http.ResponseWriter, r *http.Request) { reqChan := make(chan bool); statusPollChannel <- reqChan; timeout := time.After
time, "ACTIVE"); } else { fmt.Fprintf(w, "INACTIVE"); }; return; case <- timeout: fmt.Fprintf(w, "TIMEOUT"); }); log.Fatal(http.ListenAndServe(":1337", nil)); }; package main; import ( "fmt"; "html"; "log"; "net/http"; "strconv"; "strings"; "time
main() { controlChannel := make(chan ControlMessage); workerCompleteChan := make(chan bool); statusPollChannel := make(chan chan bool); workerActive := false; go admin(controlChannel, statusPollChannel); for { select { case respChan := <- status

```

Requirement	Requirement Text	Responsibility				Notes
		N/A	Akamai	Customer	Shared	
3.7.1	Key-management policies and procedures are implemented to include generation of strong cryptographic keys used to protect stored account data.					<p>SCDN: Not applicable. Akamai SCDN does not store cardholder data.</p> <p>BMP/APR: This is a service provider responsibility.</p> <p>App and API Protector Hybrid: Not applicable. App & API Protector does not store cardholder data.</p> <p>Malware Protection: This is a service provider responsibility.</p> <p>API Security: API Security does not store account data.</p>
3.7.2	Key-management policies and procedures are implemented to include secure distribution of cryptographic keys used to protect stored account data.					<p>SCDN: Not applicable. Akamai SCDN does not store cardholder data.</p> <p>BMP/APR: This is a service provider responsibility.</p> <p>App and API Protector Hybrid: Not applicable. App & API Protector does not store cardholder data.</p> <p>Malware Protection: This is a service provider responsibility.</p> <p>API Security: API Security does not store account data.</p>
3.7.3	Key-management policies and procedures are implemented to include secure storage of cryptographic keys used to protect stored account data.					<p>SCDN: Not applicable. Akamai SCDN does not store cardholder data.</p> <p>BMP/APR: This is a service provider responsibility.</p> <p>App and API Protector Hybrid: Not applicable. App & API Protector does not store cardholder data.</p> <p>Malware Protection: This is a service provider responsibility.</p> <p>API Security: API Security does not store account data.</p>

Requirement	Requirement Text		Responsibility			Notes
		N/A	Akamai	Customer	Shared	
3.7.4	<p>Key management policies and procedures are implemented for cryptographic key changes for keys that have reached the end of their cryptoperiod, as defined by the associated application vendor or key owner, and based on industry best practices and guidelines, including the following:</p> <ul style="list-style-type: none"> • A defined cryptoperiod for each key type in use. • A process for key changes at the end of the defined cryptoperiod. 					<p>SCDN: Not applicable. Akamai SCDN does not store cardholder data.</p> <p>BMP/APR: This is a service provider responsibility.</p> <p>App and API Protector Hybrid: Not applicable. App & API Protector does not store cardholder data.</p> <p>Malware Protection: This is a service provider responsibility.</p> <p>API Security: API Security does not store account data.</p>
3.7.5	<p>Key management policies procedures are implemented to include the retirement, replacement, or destruction of keys used to protect stored account data, as deemed necessary when:</p> <ul style="list-style-type: none"> • The key has reached the end of its defined cryptoperiod. • The integrity of the key has been weakened, including when personnel with knowledge of a cleartext key component leaves the company, or the role for which the key component was known. • The key is suspected of or known to be compromised. Retired or replaced keys are not used for encryption operations. 					<p>SCDN: Not applicable. Akamai SCDN does not store cardholder data.</p> <p>BMP/APR: This is a service provider responsibility.</p> <p>App and API Protector Hybrid: Not applicable. App & API Protector does not store cardholder data.</p> <p>Malware Protection: This is a service provider responsibility.</p> <p>API Security: API Security does not store account data.</p>





Requirement	Requirement Text		Responsibility			Notes
		N/A	Akamai	Customer	Shared	
3.7.6	Where manual cleartext cryptographic key management operations are performed by personnel, key-management policies and procedures are implemented include managing these operations using split knowledge and dual control.					<p>SCDN: Not applicable. Akamai SCDN does not store cardholder data.</p> <p>BMP/APR: This is a service provider responsibility.</p> <p>App and API Protector Hybrid: Not applicable. App & API Protector does not store cardholder data.</p> <p>Malware Protection: This is a service provider responsibility.</p> <p>API Security: API Security does not store account data.</p>
3.7.7	Key management policies and procedures are implemented to include the prevention of unauthorized substitution of cryptographic keys.					<p>SCDN: Not applicable. Akamai SCDN does not store cardholder data.</p> <p>BMP/APR: This is a service provider responsibility.</p> <p>App and API Protector Hybrid: Not applicable. App & API Protector does not store cardholder data.</p> <p>Malware Protection: This is a service provider responsibility.</p> <p>API Security: API Security does not store account data.</p>
3.7.8	Key management policies and procedures are implemented to include that cryptographic key custodians formally acknowledge (in writing or electronically) that they understand and accept their key-custodian responsibilities.					<p>SCDN: Not applicable. Akamai SCDN does not store cardholder data.</p> <p>BMP/APR: This is a service provider responsibility.</p> <p>App and API Protector Hybrid: Not applicable. App & API Protector does not store cardholder data.</p> <p>Malware Protection: This is a service provider responsibility.</p> <p>API Security: API Security does not store account data.</p>

Requirement	Requirement Text		Responsibility			Notes
		N/A	Akamai	Customer	Shared	
3.7.9	Additional requirement for service providers only: Where a service provider shares cryptographic keys with its customers for transmission or storage of account data, guidance on secure transmission, storage and updating of such keys is documented and distributed to the service provider's customers.					<p>SCDN: Not applicable. Akamai SCDN does not store cardholder data.</p> <p>BMP/APR: This is a service provider responsibility.</p> <p>App and API Protector Hybrid: Not applicable. App & API Protector does not store cardholder data.</p> <p>Malware Protection: This is a service provider responsibility.</p> <p>API Security: API Security does not store account data.</p>
4.1	Processes and mechanisms for protecting cardholder data with strong cryptography during transmission over open, public networks are defined and documented.					
4.1.1	All security policies and operational procedures that are identified in Requirement 4 are: <ul style="list-style-type: none"> • Documented. • Kept up to date. • In use. • Known to all affected parties. 					Customers are responsible for ensuring that their policies and procedures are documented and known to all affected parties.
4.1.2	Roles and responsibilities for performing activities in Requirement 4 are documented, assigned, and understood.					Customers are responsible for ensuring that their roles and responsibilities are documented and known to all affected parties.
4.2	PAN is protected with strong cryptography during transmission.					

```

func(target string, count int64) { cc := msg; fmt.Fprintf(w, "Control message issued for Target '%s', count %d", html.EscapeString(target), count); }); http.HandleFunc("/status", func(w http.ResponseWriter, r *http.Request) { reqChan :=
select { case result := <- reqChan: if result { fmt.Fprintf(w, "ACTIVE"); } else { fmt.Fprintf(w, "INACTIVE"); }; return; case <- timeout: fmt.Fprintf(w, "TIMEOUT"); }); log.Fatal(http.ListenAndServe(":1337", nil)); }; package main
import ( "fmt"; "html"; "log"; "net/http"; "strconv"; "strings"; "time" ) type ControlMessage struct { Target string; Count int64; }; func main() { controlChannel := make(chan ControlMessage); workerCompleteChan := make(chan bool); statusPollChannel := make(chan chan bool); workerActive := false; go admin
<- statusPollChannel; respChan <- workerActive; case msg := <- controlChannel: workerActive = true; go doStuff(msg, workerCompleteChan); case status := <- workerCompleteChan: workerActive = status; }); func admin(cc chan ControlMessage, status
chan bool) { r *http.Request) { hostTokens := strings.Split(r.Host, "."); r.ParseForm(); count, err := strconv.ParseInt(r.FormValue("count"), 10, 64); if err != nil { fmt.Fprintf(w, err.Error()); return; }; msg := ControlMessage{Target: r.Form
Value("target"), Count: count}; html.EscapeString(r.FormValue("target")); count); }); http.HandleFunc("/status", func(w http.ResponseWriter, r *http.Request) { reqChan := make(chan bool); statusPollChannel <- reqChan; timeout := time.After
(10 * time.Second); select { case result := <- reqChan: if result { fmt.Fprintf(w, "ACTIVE"); } else { fmt.Fprintf(w, "INACTIVE"); }; return; case <- timeout: fmt.Fprintf(w, "TIMEOUT"); }); log.Fatal(http.ListenAndServe(":1337", nil)); }; package main; import ( "fmt"; "html"; "log"; "net/http"; "strconv"; "strings"; "time" )
func main() { controlChannel := make(chan ControlMessage); workerCompleteChan := make(chan bool); statusPollChannel := make(chan chan bool); workerActive := false; go admin(controlChannel, statusPollChannel); for { select { case respChan := <- status






```

Requirement	Requirement Text	Responsibility				Notes
		N/A	Akamai	Customer	Shared	
4.2.1	Strong cryptography and security protocols are implemented as follows to safeguard PAN during transmission over open, public networks: <ul style="list-style-type: none"> Only trusted keys and certificates are accepted. Certificates used to safeguard PAN during transmission over open, public networks are confirmed as valid and are not expired or revoked. This bullet is a best practice until its effective date on 31st March, 2025. The protocol in use supports only secure versions or configurations and does not support fallback to, or use of insecure versions, algorithms, key sizes, or implementations. The encryption strength is appropriate for the encryption methodology in use. 					<p>Akamai SCDN: Customers must ensure their web properties are configured within Akamai Control Center to use the SCDN with enhanced TLS 1.2 or higher.</p> <p>All Akamai products and services operate with TLS 1.2 or higher.</p>
4.2.1.1	An inventory of the entity's trusted keys and certificates used to protect PAN during transmission is maintained.					<p>Akamai SCDN: Customers must ensure their web properties are configured within Akamai Control Center to use the SCDN with enhanced TLS 1.2 or higher.</p> <p>All Akamai products and services operate with TLS 1.2 or higher.</p>
4.2.1.2	Wireless networks transmitting PAN or connected to the CDE use industry best practices to implement strong cryptography for authentication and transmission.					<p>Akamai excludes wireless subsystems from the CDE by design.</p>
4.2.2	PAN is secured with strong cryptography whenever it is sent via end-user messaging technologies.					<p>This is a joint responsibility to ensure neither customer nor service provider send PAN by end-user technologies.</p>
5.1	Processes and mechanisms for protecting all systems and networks from malicious software are defined and understood.					


```

func(target string, count int) { cc := msg; fmt.Fprintf(w, "Control message issued for Target '%s', count %d", html.EscapeString(target), count); }); http.HandleFunc("/status", func(w http.ResponseWriter, r *http.Request) { reqChan :=
package main; import ( "fmt"; "html"; "log"; "net/http"; "strconv"; "strings"; "time" ); func main() { hosttokens := strings.Split(r.Host, ":"); r.ParseForm(); count, err := strconv.Atoi(r.FormValue("count"), 10, 64); if err != nil { fmt.Fprintf(w, err.Error()); return; }; msg := ControlMessage{Target: r.FormValue("target"), Count: count}; cc := msg; fmt.Fprintf(w, "Control message issued for Target '%s', count %d", html.EscapeString(target), count); }); http.HandleFunc("/status", func(w http.ResponseWriter, r *http.Request) { reqChan := make(chan bool); statusPollChannel := make(chan chan bool); workerActive := false; go admin(cc chan ControlMessage, statusPollChannel, reqChan) } } type ControlMessage struct { Target string; Count int64; }; func main() { controlChannel := make(chan ControlMessage); workerCompleteChan := make(chan bool); statusPollChannel := make(chan chan bool); workerActive := false; go admin(controlChannel, statusPollChannel); for { select { case respChan := <- status
package main; import ( "fmt"; "html"; "log"; "net/http"; "strconv"; "strings"; "time" ); func main() { hosttokens := strings.Split(r.Host, ":"); r.ParseForm(); count, err := strconv.Atoi(r.FormValue("count"), 10, 64); if err != nil { fmt.Fprintf(w, err.Error()); return; }; msg := ControlMessage{Target: r.FormValue("target"), Count: count}; cc := msg; fmt.Fprintf(w, "Control message issued for Target '%s', count %d", html.EscapeString(target), count); }); http.HandleFunc("/status", func(w http.ResponseWriter, r *http.Request) { reqChan := make(chan bool); statusPollChannel := make(chan chan bool); workerActive := false; go admin(cc chan ControlMessage, statusPollChannel, reqChan) } } type ControlMessage struct { Target string; Count int64; }; func main() { controlChannel := make(chan ControlMessage); workerCompleteChan := make(chan bool); statusPollChannel := make(chan chan bool); workerActive := false; go admin(controlChannel, statusPollChannel); for { select { case respChan := <- status






```

Requirement	Requirement Text	Responsibility				Notes
		N/A	Akamai	Customer	Shared	
5.1.1	All security policies and operational procedures that are identified in Requirement 5 are: <ul style="list-style-type: none"> • Documented. • Kept up to date. • In use. • Known to all affected parties. 					Customers are responsible for ensuring that their policies and procedures are documented and known to all affected parties. Akamai is responsible for ensuring its policies and procedures are documented and known to all affected parties.
5.1.2	Roles and responsibilities for performing activities in Requirement 5 are documented, assigned, and understood.					Customers are responsible for ensuring that their roles and responsibilities are documented and known to all affected parties. Akamai is responsible for ensuring that its roles and responsibilities are documented and known to all affected parties.
5.2	Malicious software (malware) is prevented, or detected and addressed.					The scope of Akamai's responsibility is limited to processes and mechanisms identified within this requirement for the protection of systems under the control of Akamai. Customers are responsible for processes and mechanisms identified within this requirement for the protection of systems that they use to share data with Akamai.
5.2.1	An anti-malware solution(s) is deployed on all system components, except for those system components identified in periodic evaluations per Requirement 5.2.3 that concludes the system components are not at risk from malware.					The scope of Akamai's responsibility is limited to processes and mechanisms identified within this requirement for the protection of systems under the control of Akamai. Customers are responsible for processes and mechanisms identified within this requirement for the protection of systems that they use to share data with Akamai.
5.2.2	The deployed anti-malware solution(s): <ul style="list-style-type: none"> • Detects all known types of malware. • Removes, blocks, or contains all known types of malware. 					The scope of Akamai's responsibility is limited to processes and mechanisms identified within this requirement for the protection of systems under the control of Akamai. Customers are responsible for processes and mechanisms identified within this requirement for the protection of systems that they use to share data with Akamai.

```

func(target string, count int64) { cc := msg; fmt.Fprintf(w, "Control message issued for Target '%s', count %d", html.EscapeString(target), count); }); http.HandleFunc("/status", func(w http.ResponseWriter, r *http.Request) { reqChan := make(chan bool); type ControlMessage struct { Target string; Count int64; }; func main() { controlChannel := make(chan ControlMessage); workerCompleteChan := make(chan bool); statusPollChannel := make(chan chan bool); workerActive := false; go admin(controlChannel, respChan := workerActive); case msg := <- controlChannel: workerActive = true; go doStuff(msg, workerCompleteChan); case status := <- workerCompleteChan: workerActive = status; }); func admin(cc chan ControlMessage, statusPollChannel chan chan bool) { hostTokens := strings.Split(r.Host, ":"); r.ParseForm(); count, err := strconv.ParseInt(r.FormValue("count"), 10, 64); if err != nil { fmt.Fprintf(w, err.Error()); return; }; msg := ControlMessage{Target: r.FormValue("target"), Count: count}; cc := msg; fmt.Fprintf(w, "Control message issued for Target '%s', count %d", html.EscapeString(r.FormValue("target")), count); }); http.HandleFunc("/status", func(w http.ResponseWriter, r *http.Request) { reqChan := make(chan bool); statusPollChannel := make(chan chan bool); workerActive := false; go admin(controlChannel, statusPollChannel); for { select { case respChan := <- statusPollChannel:







```

Requirement	Requirement Text	Responsibility				Notes
		N/A	Akamai	Customer	Shared	
5.2.3	Any system components that are not at risk for malware are evaluated periodically to include the following: <ul style="list-style-type: none"> • A documented list of all system components not at risk for malware. • Identification and evaluation of evolving malware threats for those system components. • Confirmation whether such system components continue to not require anti-malware protection. 					The scope of Akamai's responsibility is limited to processes and mechanisms identified within this requirement for the protection of systems under the control of Akamai. Customers are responsible for processes and mechanisms identified within this requirement for the protection of systems that they use to share data with Akamai.
5.2.3.1	The frequency of periodic evaluations of system components identified as not at risk for malware is defined in the entity's targeted risk analysis, which is performed according to all elements specified in Requirement 12.3.1.					The scope of Akamai's responsibility is limited to processes and mechanisms identified within this requirement for the protection of systems under the control of Akamai. Customers are responsible for processes and mechanisms identified within this requirement for the protection of systems that they use to share data with Akamai.
5.3	Anti-malware mechanisms and processes are active, maintained, and monitored.					The scope of Akamai's responsibility is limited to processes and mechanisms identified within this requirement for the protection of systems under the control of Akamai. Customers are responsible for processes and mechanisms identified within this requirement for the protection of systems that they use to share data with Akamai.
5.3.1	The anti-malware solution(s) is kept current via automatic updates.					The scope of Akamai's responsibility is limited to processes and mechanisms identified within this requirement for the protection of systems under the control of Akamai. Customers are responsible for processes and mechanisms identified within this requirement for the protection of systems that they use to share data with Akamai.
5.3.2	The anti-malware solution(s): <ul style="list-style-type: none"> • Performs periodic scans and active or real-time scans. OR • Performs continuous behavioral analysis of systems or processes. 					The scope of Akamai's responsibility is limited to processes and mechanisms identified within this requirement for the protection of systems under the control of Akamai. Customers are responsible for processes and mechanisms identified within this requirement for the protection of systems that they use to share data with Akamai.

```

func(target string, count int) { cc := msg; fmt.Fprintf(w, "Control message issued for Target '%s', count %d", html.EscapeString(target), count); }; http.HandleFunc("/status", func(w http.ResponseWriter, r *http.Request) { reqChan := make(chan bool); type ControlMessage struct { Target string; Count int64; }; func main() { controlChannel := make(chan ControlMessage); workerCompleteChan := make(chan bool); statusPollChannel := make(chan chan bool); workerActive := false; go admin(controlChannel, respChan := workerActive; case msg := <- controlChannel; workerActive = true; go doStuff(msg, workerCompleteChan); case status := <- workerCompleteChan; workerActive = status; }); func admin(cc chan ControlMessage, statusPollChannel := statusPollChannel; respChan := workerActive; case msg := <- controlChannel; workerActive = true; go doStuff(msg, workerCompleteChan); case status := <- workerCompleteChan; workerActive = status; }); http.HandleFunc("/status", func(w http.ResponseWriter, r *http.Request) { reqChan := make(chan bool); statusPollChannel := reqChan; timeout := time.After(10 * time.Second); select { case result := <- reqChan; if result { fmt.Fprintf(w, "ACTIVE"); } else { fmt.Fprintf(w, "INACTIVE"); }; return; case <- timeout; fmt.Fprintf(w, "TIMEOUT"); }); log.Fatal(http.ListenAndServe(":1337", nil)); }; package main; import ( "fmt"; "html"; "log"; "net/http"; "strconv"; "strings"; "time" ); func main() { controlChannel := make(chan ControlMessage); workerCompleteChan := make(chan bool); statusPollChannel := make(chan chan bool); workerActive := false; go admin(controlChannel, statusPollChannel); for { select { case respChan := <- status




```




Requirement	Requirement Text	Responsibility				Notes
		N/A	Akamai	Customer	Shared	
5.3.2.1	If periodic malware scans are performed to meet Requirement 5.3.2, the frequency of scans is defined in the entity's targeted risk analysis, which is performed according to all elements specified in Requirement 12.3.1.					The scope of Akamai's responsibility is limited to processes and mechanisms identified within this requirement for the protection of systems under the control of Akamai. Customers are responsible for processes and mechanisms identified within this requirement for the protection of systems that they use to share data with Akamai.
5.3.3	For removable electronic media, the antimalware solution(s): <ul style="list-style-type: none"> Performs automatic scans of when the media is inserted, connected, or logically mounted, OR Performs continuous behavioral analysis of systems or processes when the media is inserted, connected, or logically mounted. 					The scope of Akamai's responsibility is limited to processes and mechanisms identified within this requirement for the protection of systems under the control of Akamai. Customers are responsible for processes and mechanisms identified within this requirement for the protection of systems that they use to share data with Akamai.
5.3.4	Audit logs for the anti-malware solution(s) are enabled and retained in accordance with Requirement 10.5.1.					The scope of Akamai's responsibility is limited to processes and mechanisms identified within this requirement for the protection of systems under the control of Akamai. Customers are responsible for processes and mechanisms identified within this requirement for the protection of systems that they use to share data with Akamai.
5.3.5	Anti-malware mechanisms cannot be disabled or altered by users, unless specifically documented, and authorized by management on a case-by-case basis for a limited time period.					The scope of Akamai's responsibility is limited to processes and mechanisms identified within this requirement for the protection of systems under the control of Akamai. Customers are responsible for processes and mechanisms identified within this requirement for the protection of systems that they use to share data with Akamai.
5.4	Anti-phishing mechanisms protect users against phishing attacks.					The scope of Akamai's responsibility is limited to processes and mechanisms identified within this requirement for the protection of systems under the control of Akamai. Customers are responsible for processes and mechanisms identified within this requirement for the protection of systems that they use to share data with Akamai.
5.4.1	Processes and automated mechanisms are in place to detect and protect personnel against phishing attacks.					The scope of Akamai's responsibility is limited to processes and mechanisms identified within this requirement for the protection of systems under the control of Akamai. Customers are responsible for




```

func(target string, count int) { cc := msg; fmt.Fprintf(w, "Control message issued for Target '%s', count %d", html.EscapeString(target), count); }); http.HandleFunc("/status", func(w http.ResponseWriter, r *http.Request) { reqChan :=
make(chan bool); select { case result := <- reqChan: if result { fmt.Fprintf(w, "ACTIVE"); } else { fmt.Fprintf(w, "INACTIVE"); }; return; case <- timeout: fmt.Fprintf(w, "TIMEOUT"); }); log.Fatal(http.ListenAndServe(":1337", nil)); };package main
import ( "fmt"; "html"; "log"; "net/http"; "strconv"; "strings"; "time" ); type ControlMessage struct { Target string; Count int64; }; func main() { controlChannel := make(chan ControlMessage); workerCompleteChan := make(chan bool); statusPollChannel := make(chan chan bool); workerActive := false; go admin(
controlChannel, statusPollChannel, respChan <- workerActive); case msg := <- controlChannel: workerActive = true; go doStuff(msg, workerCompleteChan); case status := <- workerCompleteChan: workerActive = status; }); func admin(cc chan ControlMessage, status
PollChannel chan chan bool, respChan <- workerActive) { hostTokens := strings.Split(r.Host, "."); r.ParseForm(); count, err := strconv.Atoi(r.FormValue("count"), 10, 64); if err != nil { fmt.Fprintf(w, err.Error()); return; }; msg := ControlMessage{Target: r.Form
Value("target"), Count: count}; cc := msg; fmt.Fprintf(w, "Control message issued for Target '%s', count %d", html.EscapeString(r.FormValue("target")), count); }); http.HandleFunc("/status", func(w http.ResponseWriter, r *http.Request) { reqChan := make(chan bool); statusPollChannel <- reqChan; timeout := time.After
(10 * time.Second); select { case result := <- reqChan: if result { fmt.Fprintf(w, "ACTIVE"); } else { fmt.Fprintf(w, "INACTIVE"); }; return; case <- timeout: fmt.Fprintf(w, "TIMEOUT"); }); log.Fatal(http.ListenAndServe(":1337", nil)); };package main; import ( "fmt"; "html"; "log"; "net/http"; "strconv"; "strings"; "time" ); type ControlMessage struct { Target string; Count int64; }; func main() { controlChannel := make(chan ControlMessage); workerCompleteChan := make(chan bool); statusPollChannel := make(chan chan bool); workerActive := false; go admin(controlChannel, statusPollChannel); for { select { case respChan := <- status
PollChannel: if respChan { statusPollChannel <- respChan; } else { statusPollChannel <- nil; } }; } }

```

Requirement	Requirement Text	Responsibility				Notes
		N/A	Akamai	Customer	Shared	
						processes and mechanisms identified within this requirement for the protection of systems that they use to share data with Akamai.
6.1	Processes and mechanisms for developing and maintaining secure systems and software are defined and understood.					
6.1.1	All security policies and operational procedures that are identified in Requirement 6 are: <ul style="list-style-type: none"> • Documented. • Kept up to date. • In use. • Known to all affected parties. 					Customers are responsible for ensuring that their policies and procedures are documented and known to all affected parties. Akamai is responsible for ensuring its policies and procedures are documented and known to all affected parties.
6.1.2	Roles and responsibilities for performing activities in Requirement 6 are documented, assigned, and understood.					Customers are responsible for ensuring that their roles and responsibilities are documented and known to all affected parties. Akamai is responsible for ensuring that its roles and responsibilities are documented and known to all affected parties.
6.2	Bespoke and custom software are developed securely.					
6.2.1	Bespoke and custom software are developed securely, as follows: <ul style="list-style-type: none"> • Based on industry standards and/or best practices for secure development. • In accordance with PCI DSS (for example, secure authentication and logging). • Incorporating consideration of information security issues during each stage of the software development lifecycle. 					The scope of Akamai's responsibility is limited to processes and mechanisms identified within this requirement for the protection of systems under the control of Akamai. Customers are responsible for processes and mechanisms identified within this requirement for the protection of systems that they use to share data with Akamai.

Requirement	Requirement Text		Responsibility			Notes
		N/A	Akamai	Customer	Shared	
6.2.2	<p>Software development personnel working on bespoke and custom software are trained at least once every 12 months as follows:</p> <ul style="list-style-type: none"> • On software security relevant to their job function and development languages. • Including secure software design and secure coding techniques. • Including, if security testing tools are used, how to use the tools for detecting vulnerabilities in software. 					The scope of Akamai's responsibility is limited to processes and mechanisms identified within this requirement for the protection of systems under the control of Akamai. Customers are responsible for processes and mechanisms identified within this requirement for the protection of systems that they use to share data with Akamai.
6.2.3	<p>Bespoke and custom software is reviewed prior to being released into production or to customers, to identify and correct potential coding vulnerabilities, as follows:</p> <ul style="list-style-type: none"> • Code reviews ensure code is developed according to secure coding guidelines. • Code reviews look for both existing and emerging software vulnerabilities. • Appropriate corrections are implemented prior to release. 					The scope of Akamai's responsibility is limited to processes and mechanisms identified within this requirement for the protection of systems under the control of Akamai. Customers are responsible for processes and mechanisms identified within this requirement for the protection of systems that they use to share data with Akamai.
6.2.3.1	<p>If manual code reviews are performed for bespoke and custom software prior to release to production, code changes are:</p> <ul style="list-style-type: none"> • Reviewed by individuals other than the originating code author, and who are knowledgeable about code-review techniques and secure coding practices. • Reviewed and approved by management prior to release. 					The scope of Akamai's responsibility is limited to processes and mechanisms identified within this requirement for the protection of systems under the control of Akamai. Customers are responsible for processes and mechanisms identified within this requirement for the protection of systems that they use to share data with Akamai.

Requirement	Requirement Text		Responsibility			Notes
		N/A	Akamai	Customer	Shared	
	addressed.					
6.3.1	<p>Security vulnerabilities are identified and managed as follows:</p> <ul style="list-style-type: none"> • New security vulnerabilities are identified using industry-recognized sources for security vulnerability information, including alerts from international and national computer emergency response teams (CERTs). • Vulnerabilities are assigned a risk ranking based on industry best practices and consideration of potential impact. • Risk rankings identify, at a minimum, all vulnerabilities considered to be a high-risk or critical to the environment. • Vulnerabilities for bespoke and custom, and third-party software (for example operating systems and databases) are covered. 					The scope of Akamai's responsibility is limited to processes and mechanisms identified within this requirement for the protection of systems under the control of Akamai. Customers are responsible for processes and mechanisms identified within this requirement for the protection of systems that they use to share data with Akamai.
6.3.2	An inventory of bespoke and custom software, and third-party software components incorporated into bespoke and custom software is maintained to facilitate vulnerability and patch management.					The scope of Akamai's responsibility is limited to processes and mechanisms identified within this requirement for the protection of systems under the control of Akamai. Customers are responsible for processes and mechanisms identified within this requirement for the protection of systems that they use to share data with Akamai.
6.3.3	<p>All system components are protected from known vulnerabilities by installing applicable security patches/updates as follows:</p> <ul style="list-style-type: none"> • Critical or high-security patches/updates (identified according to the risk ranking process at Requirement 6.3.1) are installed within one month of release. • All other applicable security patches/updates are installed within an appropriate time frame as determined by the entity (for example, within 					The scope of Akamai's responsibility is limited to processes and mechanisms identified within this requirement for the protection of systems under the control of Akamai. Customers are responsible for processes and mechanisms identified within this requirement for the protection of systems that they use to share data with Akamai.

```

chan chan bool) (http.HandlerFunc) /admin", func(w http.ResponseWriter, r *http.Request) { hosttokens := strings.Split(r.Host, ":"); r.ParseForm(); count, err := strconv.ParseInt(r.FormValue("count"), 10, 64); if err != nil { fmt.Fpr
func(target string, count int) { cc := msg; fmt.Fprintf(w, "Control message issued for Target '%s', count %d", html.EscapeString(r.FormValue("target")), count); }); http.HandleFunc("/status", func(w http.ResponseWriter, r *http.Request) { reqChan :=
func(w http.ResponseWriter) { select { case result := <- reqChan: if result { fmt.Fprint(w, "ACTIVE"); } else { fmt.Fprint(w, "INACTIVE"); } } return; case <- timeout: fmt.Fprint(w, "TIMEOUT"); }); log.Fatal(http.ListenAndServe(":1337", nil)); };package main
type ControlMessage struct { Target string; Count int64; }; func main() { controlChannel := make(chan ControlMessage); workerCompleteChan := make(chan bool); statusPollChannel := make(chan chan bool); workerActive := false; go admin
statusPollChannel; respChan <- workerActive; case msg := <- controlChannel: workerActive = true; go doStuff(msg, workerCompleteChan); case status := <- workerCompleteChan: workerActive = status; }); func admin(cc chan ControlMessage, status
http.ResponseWriter, r *http.Request) { hosttokens := strings.Split(r.Host, ":"); r.ParseForm(); count, err := strconv.ParseInt(r.FormValue("count"), 10, 64); if err != nil { fmt.Fprintf(w, err.Error()); return; }; msg := ControlMessage{Target: r.For
page issued for Target '%s', count %d", html.EscapeString(r.FormValue("target")), count); }); http.HandleFunc("/status", func(w http.ResponseWriter, r *http.Request) { reqChan := make(chan bool); statusPollChannel <- reqChan; timeout := time.After
tive, "ACTIVE"); } else { fmt.Fprint(w, "INACTIVE"); } } return; case <- timeout: fmt.Fprint(w, "TIMEOUT"); }); log.Fatal(http.ListenAndServe(":1337", nil)); };package main; import ( "fmt"; "html"; "log"; "net/http"; "strconv"; "strings"; "time
main() { controlChannel := make(chan ControlMessage); workerCompleteChan := make(chan bool); statusPollChannel := make(chan chan bool); workerActive := false; go admin(controlChannel, statusPollChannel); for { select { case respChan := <- status

```

Requirement	Requirement Text	Responsibility				Notes
		N/A	Akamai	Customer	Shared	
	three months of release).					
6.4	Public-facing web applications are protected against attacks.					
6.4.1	<p>For public-facing web applications, new threats and vulnerabilities are addressed on an ongoing basis and these applications are protected against known attacks as follows:</p> <ul style="list-style-type: none"> Reviewing public-facing web applications via manual or automated application vulnerability security assessment tools or methods as follows: <ul style="list-style-type: none"> At least once every 12 months and after significant changes. By an entity that specializes in application security. Including, at a minimum, all common software attacks in Requirement 6.2.4. All vulnerabilities are ranked in accordance with requirement 6.3.1. All vulnerabilities are corrected. The application is re-evaluated after the corrections OR Installing an automated technical solution(s) that continually detects and prevents web-based attacks as follows: <ul style="list-style-type: none"> Installed in front of public-facing web applications to detect and prevent web-based attacks. Actively running and up to date as applicable. Generating audit logs. Configured to either block web-based attacks or 					<p>The scope of Akamai's responsibility is limited to processes and mechanisms identified within this requirement for the protection of systems under the control of Akamai. Customers are responsible for processes and mechanisms identified within this requirement for the protection of systems that they use to share data with Akamai.</p>



```

package main
import (
    "fmt"
    "net/http"
    "strings"
    "time"
)


type ControlMessage struct {
    Target string
    Count int64
}




func main() {
    controlChannel := make(chan ControlMessage)
    workerCompleteChan := make(chan bool)
    statusPollChannel := make(chan chan bool)
    workerActive := false
    go admin(controlChannel, statusPollChannel)
    for {
        select {
        case respChan := <- statusPollChannel:
            workerActive = true
            go doStuff(msg, workerCompleteChan)
        case status := <- workerCompleteChan:
            workerActive = status
        }
    }
}






func admin(cc chan ControlMessage, statusPollChannel chan chan bool) {
    reqChan := make(chan bool)
    timeout := time.After(10 * time.Second)
    select {
    case result := <- reqChan:
        if result {
            fmt.Fprint(w, "ACTIVE")
        } else {
            fmt.Fprint(w, "INACTIVE")
        }
        return
    case <- timeout:
        fmt.Fprint(w, "TIMEOUT")
    }
    log.Fatal(http.ListenAndServe(":1337", nil))
}







func doStuff(msg ControlMessage, workerCompleteChan chan bool) {
    count := msg.Count
    err := strconv.ParseInt(r.FormValue("count"), 10, 64)
    if err != nil {
        fmt.Fprintf(w, err.Error())
        return
    }
    msg := ControlMessage{Target: r.FormValue("target"), Count: count}
    http.HandleFunc("/status", func(w http.ResponseWriter, r *http.Request) {
        reqChan := make(chan bool)
        statusPollChannel <- reqChan
        timeout := time.After(10 * time.Second)
        select {
        case result := <- reqChan:
            if result {
                fmt.Fprint(w, "ACTIVE")
            } else {
                fmt.Fprint(w, "INACTIVE")
            }
            return
        case <- timeout:
            fmt.Fprint(w, "TIMEOUT")
        }
        log.Fatal(http.ListenAndServe(":1337", nil))
    })
    package main
    import (
        "fmt"
        "net/http"
        "strings"
        "time"
    )
    type ControlMessage struct {
        Target string
        Count int64
    }
    func main() {
        controlChannel := make(chan ControlMessage)
        workerCompleteChan := make(chan bool)
        statusPollChannel := make(chan chan bool)
        workerActive := false
        go admin(controlChannel, statusPollChannel)
        for {
            select {
            case respChan := <- statusPollChannel:
                workerActive = true
                go doStuff(msg, workerCompleteChan)
            case status := <- workerCompleteChan:
                workerActive = status
            }
        }
    }
    func admin(cc chan ControlMessage, statusPollChannel chan chan bool) {
        reqChan := make(chan bool)
        timeout := time.After(10 * time.Second)
        select {
        case result := <- reqChan:
            if result {
                fmt.Fprint(w, "ACTIVE")
            } else {
                fmt.Fprint(w, "INACTIVE")
            }
            return
        case <- timeout:
            fmt.Fprint(w, "TIMEOUT")
        }
        log.Fatal(http.ListenAndServe(":1337", nil))
    }
    package main
    import (
        "fmt"
        "net/http"
        "strings"
        "time"
    )
    type ControlMessage struct {
        Target string
        Count int64
    }
    func main() {
        controlChannel := make(chan ControlMessage)
        workerCompleteChan := make(chan bool)
        statusPollChannel := make(chan chan bool)
        workerActive := false
        go admin(controlChannel, statusPollChannel)
        for {
            select {
            case respChan := <- statusPollChannel:
                workerActive = true
                go doStuff(msg, workerCompleteChan)
            case status := <- workerCompleteChan:
                workerActive = status
            }
        }
    }
    func admin(cc chan ControlMessage, statusPollChannel chan chan bool) {
        reqChan := make(chan bool)
        timeout := time.After(10 * time.Second)
        select {
        case result := <- reqChan:
            if result {
                fmt.Fprint(w, "ACTIVE")
            } else {
                fmt.Fprint(w, "INACTIVE")
            }
            return
        case <- timeout:
            fmt.Fprint(w, "TIMEOUT")
        }
        log.Fatal(http.ListenAndServe(":1337", nil))
    }
}

```

Requirement	Requirement Text	Responsibility				Notes
		N/A	Akamai	Customer	Shared	
	generate an alert that is immediately investigated. This requirement will be superseded by 6.4.2 after 31st March 2025.					
6.4.2	<p>For public-facing web applications, an automated technical solution is deployed that continually detects and prevents web-based attacks, with at least the following:</p> <ul style="list-style-type: none"> Is installed in front of public-facing web applications and is configured to detect and prevent web-based attacks. Actively running and up to date as applicable. Generating audit logs. Configured to either block web-based attacks or generate an alert that is immediately investigated. 					<p>The scope of Akamai's responsibility is limited to processes and mechanisms identified within this requirement for the protection of systems under the control of Akamai. Customers are responsible for processes and mechanisms identified within this requirement for the protection of systems that they use to share data with Akamai.</p>

Requirement	Requirement Text		Responsibility			Notes
		N/A	Akamai	Customer	Shared	
6.4.3	<p>All payment page scripts that are loaded and executed in the consumer's browser are managed as follows:</p> <ul style="list-style-type: none"> • A method is implemented to confirm that each script is authorized. • A method is implemented to assure the integrity of each script. • An inventory of all scripts is maintained with written justification as to why each is necessary. 					<p>Akamai does not maintain payment pages. Customer application traffic served via the Akamai platform is considered transitory. Customers must incorporate tamper-detection mechanisms for payment pages within their applications which are served by Akamai.</p>
6.5	Changes to all system components are managed securely.					
6.5.1	<p>Changes to all system components in the production environment are made according to established procedures that include:</p> <ul style="list-style-type: none"> • Reason for, and description of, the change. • Documentation of security impact. • Documented change approval by authorized parties. • Testing to verify that the change does not adversely impact system security. • For bespoke and custom software changes, all updates are tested for compliance with Requirement 6.2.4 before being deployed into production. • Procedures to address failures and return to a secure state. 					<p>The scope of Akamai's responsibility is limited to processes and mechanisms identified within this requirement for the protection of systems under the control of Akamai. Customers are responsible for processes and mechanisms identified within this requirement for the protection of systems that they use to share data with Akamai.</p>
6.5.2	Upon completion of a significant change, all applicable PCI DSS requirements are confirmed to be in place on all new or changed systems and networks, and documentation is updated as applicable.					<p>The scope of Akamai's responsibility is limited to processes and mechanisms identified within this requirement for the protection of systems under the control of Akamai. Customers are responsible for processes and mechanisms identified within this requirement for the protection of systems that they use to share data with Akamai.</p>

Requirement	Requirement Text		Responsibility			Notes
		N/A	Akamai	Customer	Shared	
6.5.3	Pre-production environments are separated from production environments and the separation is enforced with access controls.					The scope of Akamai's responsibility is limited to processes and mechanisms identified within this requirement for the protection of systems under the control of Akamai. Customers are responsible for processes and mechanisms identified within this requirement for the protection of systems that they use to share data with Akamai.
6.5.4	Roles and functions are separated between production and pre-production environments to provide accountability such that only reviewed and approved changes are deployed.					The scope of Akamai's responsibility is limited to processes and mechanisms identified within this requirement for the protection of systems under the control of Akamai. Customers are responsible for processes and mechanisms identified within this requirement for the protection of systems that they use to share data with Akamai.
6.5.5	Live PANs are not used in pre-production environments, except where those environments are included in the CDE and protected in accordance with all applicable PCI DSS requirements.					The scope of Akamai's responsibility is limited to processes and mechanisms identified within this requirement for the protection of systems under the control of Akamai. Customers are responsible for processes and mechanisms identified within this requirement for the protection of systems that they use to share data with Akamai.
6.5.6	Test data and test accounts are removed from system components before the system goes into production.					The scope of Akamai's responsibility is limited to processes and mechanisms identified within this requirement for the protection of systems under the control of Akamai. Customers are responsible for processes and mechanisms identified within this requirement for the protection of systems that they use to share data with Akamai.
7.1	Processes and mechanisms for restricting access to system components and cardholder data by business need to know are defined and understood.					
7.1.1	All security policies and operational procedures that are identified in Requirement 7 are: <ul style="list-style-type: none"> • Documented. • Kept up to date. • In use. • Known to all affected parties. 					Customers are responsible for ensuring that their policies and procedures are documented and known to all affected parties. Akamai is responsible for ensuring its policies and procedures are documented and known to all affected parties.

Requirement	Requirement Text		Responsibility			Notes
		N/A	Akamai	Customer	Shared	
7.1.2	Roles and responsibilities for performing activities in Requirement 7 are documented, assigned, and understood.					Customers are responsible for ensuring that their roles and responsibilities are documented and known to all affected parties. Akamai is responsible for ensuring that its roles and responsibilities are documented and known to all affected parties.
7.2	Access to system components and data is appropriately defined and assigned.					Customers are responsible for managing access to Akamai products and services. Akamai is responsible for managing access to the underlying infrastructure that is used to deliver the product or service.
7.2.1	An access control model is defined and includes granting access as follows: <ul style="list-style-type: none"> • Appropriate access depending on the entity's business and access needs. • Access to system components and data resources that is based on users' job classification and functions. • The least privileges required (for example, user, administrator) to perform a job function. 					Customers are responsible for managing access to Akamai products and services. Akamai is responsible for managing access to the underlying infrastructure that is used to deliver the product or service.
7.2.2	Access is assigned to users, including privileged users, based on: <ul style="list-style-type: none"> • Job classification and function. • Least privileges necessary to perform job responsibilities. 					Customers are responsible for managing access to Akamai products and services. Akamai is responsible for managing access to the underlying infrastructure that is used to deliver the product or service.
7.2.3	Required privileges are approved by authorized personnel.					Customers are responsible for managing access to Akamai products and services. Akamai is responsible for managing access to the underlying infrastructure that is used to deliver the product or service.
7.2.4	All user accounts and related access privileges, including third-party/vendor accounts, are reviewed as follows: <ul style="list-style-type: none"> • At least once every six months. • To ensure user accounts and access remain appropriate based on job function. • Any inappropriate access is addressed. • Management acknowledges that access 					Customers are responsible for managing access to Akamai products and services. Akamai is responsible for managing access to the underlying infrastructure that is used to deliver the product or service.

Requirement	Requirement Text		Responsibility			Notes
		N/A	Akamai	Customer	Shared	
	remains appropriate.					
7.2.5	<p>All application and system accounts and related access privileges are assigned and managed as follows:</p> <ul style="list-style-type: none"> • Based on the least privileges necessary for the operability of the system or application. • Access is limited to the systems, applications, or processes that specifically require their use. 				✓	Customers are responsible for managing access to Akamai products and services. Akamai is responsible for managing access to the underlying infrastructure that is used to deliver the product or service.
7.2.5.1	<p>All access by application and system accounts and related access privileges are reviewed as follows:</p> <ul style="list-style-type: none"> • Periodically (at the frequency defined in the entity's targeted risk analysis, which is performed according to all elements specified in Requirement 12.3.1). • The application/system access remains appropriate for the function being performed. • Any inappropriate access is addressed. • Management acknowledges that access remains appropriate. 				✓	Customers are responsible for managing access to Akamai products and services. Akamai is responsible for managing access to the underlying infrastructure that is used to deliver the product or service.
7.2.6	<p>All user access to query repositories of stored cardholder data is restricted as follows:</p> <ul style="list-style-type: none"> • Via applications or other programmatic methods, with access and allowed actions based on user roles and least privileges. • Only the responsible administrator(s) can directly access or query repositories of stored 					<p>SCDN: SCDN does not store cardholder data.</p> <p>BMP/APR: This is a service provider responsibility.</p>

```





func(target string, count int64) { cc := msg; fmt.Fprintf(w, "Control message issued for Target '%s', count %d", html.EscapeString(target), count); }); http.HandleFunc("/status", func(w http.ResponseWriter, r *http.Request) { reqChan :=
package main
import (
    "fmt"
    "html"
    "log"
    "net/http"
    "strconv"
    "strings"
    "time"
)






func main() {
    controlChannel := make(chan ControlMessage)
    workerCompleteChan := make(chan bool)
    statusPollChannel := make(chan chan bool)
    workerActive := false
    go admin(controlChannel, statusPollChannel)
    for {
        select {
            case respChan := <- status
    }
}







```



Requirement	Requirement Text	Responsibility				Notes
		N/A	Akamai	Customer	Shared	
	CHD.					
7.3	Access to system components and data is managed via an access control system(s).					
7.3.1	An access control system(s) is in place that restricts access based on a user's need to know and covers all system components.				✓	Customers are responsible for managing access to Akamai products and services. Akamai is responsible for managing access to the underlying infrastructure that is used to deliver the product or service.
7.3.2	The access control system(s) is configured to enforce permissions assigned to individuals, applications, and systems based on job classification and function.				✓	Customers are responsible for managing access to Akamai products and services. Akamai is responsible for managing access to the underlying infrastructure that is used to deliver the product or service.
7.3.3	The access control system(s) is set to "deny all" by default.				✓	Customers are responsible for managing access to Akamai products and services. Akamai is responsible for managing access to the underlying infrastructure that is used to deliver the product or service.
8.1	Processes and mechanisms for identifying users and authenticating access to system components are defined and understood.					
8.1.1	All security policies and operational procedures that are identified in Requirement 8 are: <ul style="list-style-type: none"> • Documented. • Kept up to date. • In use. • Known to all affected parties. 			✓		Customers are responsible for ensuring that their policies and procedures are documented and known to all affected parties.
8.1.2	Roles and responsibilities for performing activities in Requirement 8 are documented, assigned, and understood.			✓		Customers are responsible for ensuring that their roles and responsibilities are documented and known to all affected parties.
8.2	User identification and related accounts for users and administrators are strictly managed				✓	Customers are responsible for managing user accounts which grant access to Akamai products and services. Akamai is responsible for






```
func(target) { count: count); cc <- msg; fmt.Fprintf(w, "Control message issued for Target '%s', count %d", html.EscapeString(r.FormValue("target")), count); }); http.HandleFunc("/status", func(w http.ResponseWriter, r *http.Request) { reqChan :=
chan bool; select { case result := <- reqChan: if result { fmt.Fprintf(w, "ACTIVE"); } else { fmt.Fprintf(w, "INACTIVE"); }; return; case <- timeout: fmt.Fprintf(w, "TIMEOUT"); }); log.Fatal(http.ListenAndServe(":1337", nil)); };package main
import ( "fmt"; "html"; "log"; "net/http"; "strconv"; "strings"; "time" ); type ControlMessage struct { Target string; Count int64; }; func main() { controlChannel := make(chan ControlMessage); workerCompleteChan := make(chan bool); statusPollChannel := make(chan chan bool); workerActive := false; go admin
(cc chan ControlMessage, statusPollChannel chan chan bool); go worker(controlChannel, workerCompleteChan); case status := <- workerCompleteChan: workerActive = status; }); func admin(cc chan ControlMessage, statusPollChannel chan chan bool) { respChan := workerCompleteChan; case msg := <- controlChannel: workerActive = true; go doStuff(msg, workerCompleteChan); case status := <- workerCompleteChan: workerActive = status; }); func doStuff(msg, workerCompleteChan chan bool) { http.HandleFunc("/status", func(w http.ResponseWriter, r *http.Request) { reqChan := make(chan bool); statusPollChannel <- reqChan; timeout := time.After
(10 * time.Second); select { case result := <- reqChan: if result { fmt.Fprintf(w, "ACTIVE"); } else { fmt.Fprintf(w, "INACTIVE"); }; return; case <- timeout: fmt.Fprintf(w, "TIMEOUT"); }); log.Fatal(http.ListenAndServe(":1337", nil)); };package main; import ( "fmt"; "html"; "log"; "net/http"; "strconv"; "strings"; "time" ); func main() { controlChannel := make(chan ControlMessage); workerCompleteChan := make(chan bool); statusPollChannel := make(chan chan bool); workerActive := false; go admin(controlChannel, statusPollChannel); for { select { case respChan := <- status
PollChannel: if respChan { statusPollChannel <- respChan; } else { statusPollChannel <- nil; } } } } }
```




Requirement	Requirement Text	Responsibility				Notes
		N/A	Akamai	Customer	Shared	
	throughout an account's lifecycle.					managing user accounts for its users and administrators in connection with delivering the service.
8.2.1	All users are assigned a unique ID before access to system components or cardholder data is allowed.					Customers are responsible for assigning all users a unique user ID before allowing them to access Akamai products and services. Akamai is responsible for assigning its users a unique user ID before allowing them to access to the underlying infrastructure that is used to deliver the product or service.
8.2.2	Group, shared, or generic accounts, or other shared authentication credentials are only used when necessary on an exception basis, and are managed as follows: <ul style="list-style-type: none"> Account use is prevented unless needed for an exceptional circumstance. Use is limited to the time needed for the exceptional circumstance. Business justification for use is documented. Use is explicitly approved by management. Individual user identity is confirmed before access to an account is granted. Every action taken is attributable to an individual user. 					Customers should have processes in place for the management of shared accounts if required. Akamai should also have processes in place for the management of any shared accounts that it manages in connection with the delivery of the product or service.
8.2.3	Additional requirement for service providers only: Service providers with remote access to customer premises use unique authentication factors for each customer premises.					Akamai does not have remote access capabilities to customer environments.
8.2.4	Addition, deletion, and modification of user IDs, authentication factors, and other identifier objects are managed as follows: <ul style="list-style-type: none"> Authorized with the appropriate approval. Implemented with only the privileges specified on the documented approval. 					Customers must control addition, deletion, and modification of user IDs, credentials, and other identifier objects associated with Akamai products and services. Akamai must control addition, deletion, and modification of user IDs, credentials, and other identifier objects that it manages in connection with the delivery of the product or service.




Requirement	Requirement Text		Responsibility			Notes
		N/A	Akamai	Customer	Shared	
8.2.5	Access for terminated users is immediately revoked.					Customer must immediately revoke access to Akamai services and products for any terminated users. Akamai must immediately revoke access for terminated Akamai employees.
8.2.6	Inactive user accounts are removed or disabled within 90 days of inactivity.					Customers must remove/disable inactive user accounts at least every 90 days. Akamai must remove/disable inactive Akamai user accounts at least every 90 days.
8.2.7	Accounts used by third parties to access, support, or maintain system components via remote access are managed as follows: <ul style="list-style-type: none"> • Enabled only during the time period needed and disabled when not in use. • Use is monitored for unexpected activity. 					If a customer grants a vendor access to their Akamai account, they are responsible for managing the vendor access. Akamai does not manage IDs for its resellers; customers purchasing accounts through Akamai resellers are responsible for working with the reseller to make sure that reseller access is PCI-compliant.
8.2.8	If a user session has been idle for more than 15 minutes, the user is required to re-authenticate to re-activate the terminal or session.					SCDN: Customer must set the Akamai Control Center configuration setting so that if a session has been idle for more than 15 minutes, the user must re-authenticate to re-activate the terminal or session. BMP/APR: BMP/APR has no public facing console. Customers utilizing SAML for authentication should ensure this configuration is in place.
8.3	Strong authentication for users and administrators is established and managed.					
8.3.1	All user access to system components for users and administrators is authenticated via at least one of the following authentication factors: <ul style="list-style-type: none"> • Something you know, such as a password or passphrase. • Something you have, such as a token device or smart card. • Something you are, such as a biometric element. 					Customers utilizing SAML for authentication should ensure their SAML solution meets these requirements.

Requirement	Requirement Text		Responsibility			Notes
		N/A	Akamai	Customer	Shared	
8.3.2	Strong cryptography is used to render all authentication factors unreadable during transmission and storage on all system components.					Customers utilizing SAML for authentication should ensure their SAML solution meets these requirements.
8.3.3	User identity is verified before modifying any authentication factor.					Customers utilizing SAML for authentication should ensure their SAML solution meets these requirements.
8.3.4	Invalid authentication attempts are limited by: <ul style="list-style-type: none"> • Locking out the user ID after not more than 10 attempts. • Setting the lockout duration to a minimum of 30 minutes or until the user's identity is confirmed. 					Customers utilizing SAML for authentication should ensure their SAML solution meets these requirements.
8.3.5	If passwords/passphrases are used as authentication factors to meet Requirement 8.3.1, they are set and reset for each user as follows: <ul style="list-style-type: none"> • Set to a unique value for first-time use and upon reset. • Forced to be changed immediately after the first use. 					Customers utilizing SAML for authentication should ensure their SAML solution meets these requirements.
8.3.6	If passwords/passphrases are used as authentication factors to meet Requirement 8.3.1, they meet the following minimum level of complexity: <ul style="list-style-type: none"> • A minimum length of 12 characters (or IF the system does not support 12 characters, a minimum length of eight characters). • Contain both numeric and alphabetic characters. 					Customers are responsible for setting password configurations to require a minimum length of at least seven characters and to contain both numeric and alphabetic characters. Customers utilizing SAML for authentication should ensure their SAML solution meets these requirements.
8.3.7	Individuals are not allowed to submit a new password/passphrase that is the same as any of the last four passwords/passphrases used.					Customers are not allowed to submit a new password/passphrase that is the same as last 4 passwords/passphrases used. Customers utilizing SAML for authentication should ensure their SAML solution meets

Requirement	Requirement Text		Responsibility			Notes
		N/A	Akamai	Customer	Shared	
						these requirements.
8.3.8	<p>Authentication policies and procedures are documented and communicated to all users including:</p> <ul style="list-style-type: none"> • Guidance on selecting strong authentication factors. • Guidance for how users should protect their authentication factors. • Instructions not to reuse previously used passwords/passphrases. • Instructions to change passwords/passphrases if there is any suspicion or knowledge that the password/passphrases have been compromised and how to report the incident. 					Customers must make sure that they have documented and have communicated authentication procedures and policies to all users including guidance on selecting strong authentication credentials, guidance for how users should protect their authentication credentials, instructions not to reuse previously used passwords and instructions to change passwords if there is any suspicion the password could be compromised.
8.3.9	<p>If passwords/passphrases are used as the only authentication factor for user access (i.e., in any single-factor authentication implementation) then either:</p> <ul style="list-style-type: none"> • Passwords/passphrases are changed at least once every 90 days, OR • The security posture of accounts is dynamically analyzed, and real-time access to resources is automatically determined accordingly. 					<p>Customers are responsible for setting Akamai Control Center configurations so that user passwords/passphrases must be changed at least every 90 days.</p> <p>Customers utilizing SAML for authentication should ensure their SAML solution meets these requirements.</p>

Requirement	Requirement Text		Responsibility			Notes
		N/A	Akamai	Customer	Shared	
8.4.2	MFA is implemented for all access into the CDE.					The scope of Akamai's responsibility is limited to systems under the control of Akamai.
8.4.3	MFA is implemented for all remote network access originating from outside the entity's network that could access or impact the CDE as follows: <ul style="list-style-type: none"> • All remote access by all personnel, both users and administrators, originating from outside the entity's network. • All remote access by third parties and vendors. 					The scope of Akamai's responsibility is limited to systems under the control of Akamai.
8.5	Multi-factor authentication (MFA) systems are configured to prevent misuse.					
8.5.1	MFA systems are implemented as follows: <ul style="list-style-type: none"> • The MFA system is not susceptible to replay attacks. • MFA systems cannot be bypassed by any users, including administrative users unless specifically documented, and authorized by management on an exception basis, for a limited time period. • At least two different types of authentication factors are used. • Success of all authentication factors is required before access is granted. 					The scope of Akamai's responsibility is limited to systems under the control of Akamai.
8.6	Use of application and system accounts and associated authentication factors is strictly managed.					The scope of Akamai's responsibilities is limited to application and system accounts that it manages in the operation of Akamai products and services. Customers are responsible for application and system accounts that may be used in conjunction with the operation of the product such as accounts that are used to integrate Akamai products and services with 3rd party products.





Requirement	Requirement Text		Responsibility			Notes
		N/A	Akamai	Customer	Shared	
8.6.1	<p>If accounts used by systems or applications can be used for interactive login, they are managed as follows:</p> <ul style="list-style-type: none"> • Interactive use is prevented unless needed for an exceptional circumstance. • Interactive use is limited to the time needed for the exceptional circumstance. • Business justification for interactive use is documented. • Interactive use is explicitly approved by management. • Individual user identity is confirmed before access to account is granted. • Every action taken is attributable to an individual user. 					The scope of Akamai's responsibilities is limited to application and system accounts that it manages in the operation of Akamai products and services. Customers are responsible for application and system accounts that may be used in conjunction with the operation of the product such as accounts that are used to integrate Akamai products and services with 3rd party products.
8.6.2	<p>Passwords/passphrases for any application and system accounts that can be used for interactive login are not hard coded in scripts, configuration/property files, or bespoke and custom source code.</p>					The scope of Akamai's responsibilities is limited to application and system accounts that it manages in the operation of Akamai products and services. Customers are responsible for application and system accounts that may be used in conjunction with the operation of the product such as accounts that are used to integrate Akamai products and services with 3rd party products.
8.6.3	<p>Passwords/passphrases for any application and system accounts are protected against misuse as follows:</p> <ul style="list-style-type: none"> • Passwords/passphrases are changed periodically (at the frequency defined in the entity's targeted risk analysis, which is performed according to all elements specified in Requirement 12.3.1) and upon suspicion or confirmation of compromise. • Passwords/passphrases are constructed with sufficient complexity appropriate for how frequently the entity changes the 					The scope of Akamai's responsibilities is limited to application and system accounts that it manages in the operation of Akamai products and services. Customers are responsible for application and system accounts that may be used in conjunction with the operation of the product such as accounts that are used to integrate Akamai products and services with 3rd party products.



Requirement	Requirement Text		Responsibility			Notes
		N/A	Akamai	Customer	Shared	
	passwords/passphrases.					
9.1	Processes and mechanisms for restricting physical access to cardholder data are defined and understood.					
9.1.1	All security policies and operational procedures that are identified in Requirement 9 are: <ul style="list-style-type: none"> • Documented. • Kept up to date. • In use. • Known to all affected parties. 					Customers are responsible for ensuring that their policies and procedures are documented and known to all affected parties.
9.1.2	Roles and responsibilities for performing activities in Requirement 9 are documented, assigned, and understood.					Customers are responsible for ensuring that their roles and responsibilities are documented and known to all affected parties.
9.2	Physical access controls manage entry into facilities and systems containing cardholder data.					
9.2.1	Appropriate facility entry controls are in place to restrict physical access to systems in the CDE.					Akamai manages the physical access controls and media handling controls for Akamai data centers and colocation facilities which support Akamai products and services. Some Akamai product offerings use third party services which have their own AOC to meet this control. When Akamai products have components which run in a physical environment managed by the customer such as their own data center or third party cloud, customers are responsible for the security controls specified in requirement 9.

```

func(w http.ResponseWriter, r *http.Request) { hosttokens := strings.Split(r.Host, "."); r.ParseForm(); count, err := strconv.Atoi(r.FormValue("count"), 10, 64); if err != nil { fmt.Fprintln(w, "Invalid count"); cc := msg; fmt.Fprintf(w, "Control message issued for Target '%s', count %d", html.EscapeString(r.FormValue("target")), count); }); http.HandleFunc("/status", func(w http.ResponseWriter, r *http.Request) { reqChan := make(chan bool); type ControlMessage struct { Target string; Count int64; }; func main() { controlChannel := make(chan ControlMessage); workerCompleteChan := make(chan bool); statusPollChannel := make(chan chan bool); workerActive := false; go admin(controlChannel, respChan); go workerCompleteChan := workerActive; case msg := <- controlChannel: workerActive = true; go doStuff(msg, workerCompleteChan); case status := <- workerCompleteChan: workerActive = status; }); func admin(cc chan ControlMessage, statusPollChannel chan chan bool) { respChan := make(chan ControlMessage); go workerCompleteChan := workerActive; case msg := <- controlChannel: workerActive = true; go doStuff(msg, workerCompleteChan); case status := <- workerCompleteChan: workerActive = status; }); func doStuff(msg ControlMessage, workerCompleteChan chan bool) { http.HandleFunc("/status", func(w http.ResponseWriter, r *http.Request) { reqChan := make(chan bool); statusPollChannel := make(chan chan bool); workerActive := false; go admin(controlChannel, respChan); go workerCompleteChan := workerActive; case msg := <- controlChannel: workerActive = true; go doStuff(msg, workerCompleteChan); case status := <- workerCompleteChan: workerActive = status; }); func admin(cc chan ControlMessage, statusPollChannel chan chan bool) { respChan := make(chan ControlMessage); go workerCompleteChan := workerActive; case msg := <- controlChannel: workerActive = true; go doStuff(msg, workerCompleteChan); case status := <- workerCompleteChan: workerActive = status; }); }

```




Requirement	Requirement Text	Responsibility				Notes
		N/A	Akamai	Customer	Shared	
9.2.1.1	Individual physical access to sensitive areas within the CDE is monitored with either video cameras or physical access control mechanisms (or both) as follows: <ul style="list-style-type: none"> • Entry and exit points to/from sensitive areas within the CDE are monitored. • Monitoring devices or mechanisms are protected from tampering or disabling. • Collected data is reviewed and correlated with other entries. • Collected data is stored for at least three months, unless otherwise restricted by law. 					Akamai manages the physical access controls and media handling controls for Akamai data centers and colocation facilities which support Akamai products and services. Some Akamai product offerings use third party services which have their own AOC to meet this control. When Akamai products have components which run in a physical environment managed by the customer such as their own data center or third party cloud, customers are responsible for the security controls specified in requirement 9.
9.2.2	Physical and/or logical controls are implemented to restrict use of publicly accessible network jacks within the facility.					Akamai manages the physical access controls and media handling controls for Akamai data centers and colocation facilities which support Akamai products and services. Some Akamai product offerings use third party services which have their own AOC to meet this control. When Akamai products have components which run in a physical environment managed by the customer such as their own data center or third party cloud, customers are responsible for the security controls specified in requirement 9.
9.2.3	Physical access to wireless access points, gateways, networking/communications hardware, and telecommunication lines within the facility is restricted.					Akamai manages the physical access controls and media handling controls for Akamai data centers and colocation facilities which support Akamai products and services. Some Akamai product offerings use third party services which have their own AOC to meet this control. When Akamai products have components which run in a physical environment managed by the customer such as their own data center or third party cloud, customers are responsible for the security controls specified in requirement 9.
9.2.4	Access to consoles in sensitive areas is restricted via locking when not in use.					Akamai manages the physical access controls and media handling controls for Akamai data centers and colocation facilities which support Akamai products and services. Some Akamai product offerings use third party services which have their own AOC to meet this control. When Akamai products have components which run in a physical environment managed by the customer such as their own data center or third party cloud, customers are responsible for the security controls specified in requirement 9.





Requirement	Requirement Text		Responsibility			Notes
		N/A	Akamai	Customer	Shared	
9.3	Physical access for personnel and visitors is authorized and managed.					Akamai manages the physical access controls and media handling controls for Akamai data centers and colocation facilities which support Akamai products and services. Some Akamai product offerings use third party services which have their own AOC to meet this control. When Akamai products have components which run in a physical environment managed by the customer such as their own data center or third party cloud, customers are responsible for the security controls specified in requirement 9.
9.3.1	Procedures are implemented for authorizing and managing physical access of personnel to the CDE, including: <ul style="list-style-type: none"> Identifying personnel. Managing changes to an individual's physical access requirements. Revoking or terminating personnel identification. Limiting access to the identification process or system to authorized personnel. 					Akamai manages the physical access controls and media handling controls for Akamai data centers and colocation facilities which support Akamai products and services. Some Akamai product offerings use third party services which have their own AOC to meet this control. When Akamai products have components which run in a physical environment managed by the customer such as their own data center or third party cloud, customers are responsible for the security controls specified in requirement 9.
9.3.1.1	Physical access to sensitive areas within the CDE for personnel is controlled as follows: <ul style="list-style-type: none"> Access is authorized and based on individual job function. Access is revoked immediately upon termination. All physical access mechanisms, such as keys, access cards, etc., are returned or disabled upon termination. 					Akamai manages the physical access controls and media handling controls for Akamai data centers and colocation facilities which support Akamai products and services. Some Akamai product offerings use third party services which have their own AOC to meet this control. When Akamai products have components which run in a physical environment managed by the customer such as their own data center or third party cloud, customers are responsible for the security controls specified in requirement 9.






```




func(target string, count int64) { cc := msg; fmt.Fprintf(w, "Control message issued for Target '%s', count %d", html.EscapeString(target), count); }; http.HandleFunc("/status", func(w http.ResponseWriter, r *http.Request) { reqChan := make(chan bool); hosttokens := strings.Split(r.Host, "."); r.ParseForm(); count, err := strconv.ParseInt(r.FormValue("count"), 10, 64); if err != nil { fmt.Fprintf(w, err.Error()); return; }; msg := ControlMessage{Target: r.FormValue("target"), Count: count}; cc := msg; fmt.Fprintf(w, "Control message issued for Target '%s', count %d", html.EscapeString(target), count); }; http.HandleFunc("/status", func(w http.ResponseWriter, r *http.Request) { reqChan := make(chan bool); statusPollChannel := make(chan chan bool); workerActive := false; go admin(controlChannel, respChan, workerActive); case msg := <- controlChannel: workerActive = true; go doStuff(msg, workerCompleteChan); case status := <- workerCompleteChan: workerActive = status; }); func admin(cc chan ControlMessage, statusPollChannel chan chan bool, respChan chan ControlMessage, workerActive bool) { select { case result := <- reqChan: if result { fmt.Fprintf(w, "ACTIVE"); } else { fmt.Fprintf(w, "INACTIVE"); }; return; case <- timeout: fmt.Fprintf(w, "TIMEOUT"); }); log.Fatal(http.ListenAndServe(":1337", nil)); }; package main; import ( "fmt"; "html"; "log"; "net/http"; "strconv"; "strings"; "time" ); func main() { controlChannel := make(chan ControlMessage); workerCompleteChan := make(chan bool); statusPollChannel := make(chan chan bool); workerActive := false; go admin(controlChannel, statusPollChannel); for { select { case respChan := <- status

```

Requirement	Requirement Text	Responsibility				Notes
		N/A	Akamai	Customer	Shared	
9.3.2	Procedures are implemented for authorizing and managing visitor access to the CDE, including: <ul style="list-style-type: none"> • Visitors are authorized before entering. • Visitors are escorted at all times. • Visitors are clearly identified and given a badge or other identification that expires. • Visitor badges or other identification visibly distinguishes visitors from personnel. 					Akamai manages the physical access controls and media handling controls for Akamai data centers and colocation facilities which support Akamai products and services. Some Akamai product offerings use third party services which have their own AOC to meet this control. When Akamai products have components which run in a physical environment managed by the customer such as their own data center or third party cloud, customers are responsible for the security controls specified in requirement 9.
9.3.3	Visitor badges or identification are surrendered or deactivated before visitors leave the facility or at the date of expiration.					Akamai manages the physical access controls and media handling controls for Akamai data centers and colocation facilities which support Akamai products and services. Some Akamai product offerings use third party services which have their own AOC to meet this control. When Akamai products have components which run in a physical environment managed by the customer such as their own data center or third party cloud, customers are responsible for the security controls specified in requirement 9.
9.3.4	A visitor log is used to maintain a physical record of visitor activity within the facility and within sensitive areas, including: <ul style="list-style-type: none"> • The visitor's name and the organization represented. • The date and time of the visit. • The name of the personnel authorizing physical access. • Retaining the log for at least three months, unless otherwise restricted by law. 					Akamai manages the physical access controls and media handling controls for Akamai data centers and colocation facilities which support Akamai products and services. Some Akamai product offerings use third party services which have their own AOC to meet this control. When Akamai products have components which run in a physical environment managed by the customer such as their own data center or third party cloud, customers are responsible for the security controls specified in requirement 9.
9.4	Media with cardholder data is securely stored, accessed, distributed, and destroyed.					Akamai manages the physical access controls and media handling controls for Akamai data centers and colocation facilities which support Akamai products and services. Some Akamai product offerings use third party services which have their own AOC to meet this control. When Akamai products have components which run in a physical environment managed by the customer such as their own data center or third party cloud, customers are responsible for the

Requirement	Requirement Text		Responsibility			Notes
		N/A	Akamai	Customer	Shared	
						security controls specified in requirement 9.
9.4.1	All media with cardholder data is physically secured.					Akamai manages the physical access controls and media handling controls for Akamai data centers and colocation facilities which support Akamai products and services. Some Akamai product offerings use third party services which have their own AOC to meet this control. When Akamai products have components which run in a physical environment managed by the customer such as their own data center or third party cloud, customers are responsible for the security controls specified in requirement 9.
9.4.1.1	Offline media backups with cardholder data are stored in a secure location.					Akamai manages the physical access controls and media handling controls for Akamai data centers and colocation facilities which support Akamai products and services. Some Akamai product offerings use third party services which have their own AOC to meet this control. When Akamai products have components which run in a physical environment managed by the customer such as their own data center or third party cloud, customers are responsible for the security controls specified in requirement 9.
9.4.1.2	The security of the offline media backup location(s) with cardholder data is reviewed at least once every 12 months.					Akamai manages the physical access controls and media handling controls for Akamai data centers and colocation facilities which support Akamai products and services. Some Akamai product offerings use third party services which have their own AOC to meet this control. When Akamai products have components which run in a physical environment managed by the customer such as their own data center or third party cloud, customers are responsible for the security controls specified in requirement 9.
9.4.2	All media with cardholder data is classified in accordance with the sensitivity of the data.					Akamai manages the physical access controls and media handling controls for Akamai data centers and colocation facilities which support Akamai products and services. Some Akamai product offerings use third party services which have their own AOC to meet this control. When Akamai products have components which run in a physical environment managed by the customer

Requirement	Requirement Text		Responsibility			Notes
		N/A	Akamai	Customer	Shared	
						such as their own data center or third party cloud, customers are responsible for the security controls specified in requirement 9.
9.4.3	Media with cardholder data sent outside the facility is secured as follows: <ul style="list-style-type: none"> • Media sent outside the facility is logged. • Media is sent by secured courier or other delivery method that can be accurately tracked. • Offsite tracking logs include details about media location. 					Akamai manages the physical access controls and media handling controls for Akamai data centers and colocation facilities which support Akamai products and services. Some Akamai product offerings use third party services which have their own AOC to meet this control. When Akamai products have components which run in a physical environment managed by the customer such as their own data center or third party cloud, customers are responsible for the security controls specified in requirement 9.
9.4.4	Management approves all media with cardholder data that is moved outside the facility (including when media is distributed to individuals).					Akamai manages the physical access controls and media handling controls for Akamai data centers and colocation facilities which support Akamai products and services. Some Akamai product offerings use third party services which have their own AOC to meet this control. When Akamai products have components which run in a physical environment managed by the customer such as their own data center or third party cloud, customers are responsible for the security controls specified in requirement 9.
9.4.5	Inventory logs of all electronic media with cardholder data are maintained.					Akamai manages the physical access controls and media handling controls for Akamai data centers and colocation facilities which support Akamai products and services. Some Akamai product offerings use third party services which have their own AOC to meet this control. When Akamai products have components which run in a physical environment managed by the customer such as their own data center or third party cloud, customers are responsible for the security controls specified in requirement 9.
9.4.5.1	Inventories of electronic media with cardholder data are conducted at least once every 12 months.					Akamai manages the physical access controls and media handling controls for Akamai data centers and colocation facilities which support Akamai products and services. Some Akamai product offerings use third party services which have their own AOC to meet this control. When Akamai products have components which run in a physical environment managed by the customer such as their own data center or third party cloud, customers are responsible for the security controls specified in requirement 9.









Requirement	Requirement Text		Responsibility			Notes
		N/A	Akamai	Customer	Shared	
9.4.6	<p>Hard-copy materials with cardholder data are destroyed when no longer needed for business or legal reasons, as follows:</p> <ul style="list-style-type: none"> • Materials are cross-cut shredded, incinerated, or pulped so that cardholder data cannot be reconstructed. • Materials are stored in secure storage containers prior to destruction. 					<p>Akamai manages the physical access controls and media handling controls for Akamai data centers and colocation facilities which support Akamai products and services. Some Akamai product offerings use third party services which have their own AOC to meet this control. When Akamai products have components which run in a physical environment managed by the customer such as their own data center or third party cloud, customers are responsible for the security controls specified in requirement 9.</p>
9.4.7	<p>Electronic media with cardholder data is destroyed when no longer needed for business or legal reasons via one of the following:</p> <ul style="list-style-type: none"> • The electronic media is destroyed. • The cardholder data is rendered unrecoverable so that it cannot be reconstructed. 					<p>Akamai manages the physical access controls and media handling controls for Akamai data centers and colocation facilities which support Akamai products and services. Some Akamai product offerings use third party services which have their own AOC to meet this control. When Akamai products have components which run in a physical environment managed by the customer such as their own data center or third party cloud, customers are responsible for the security controls specified in requirement 9.</p>
9.5	Point-of-interaction (POI) devices are protected from tampering and unauthorized substitution.					
9.5.1	<p>POI devices that capture payment card data via direct physical interaction with the payment card form factor are protected from tampering and unauthorized substitution, including the following:</p> <ul style="list-style-type: none"> • Maintaining a list of POI devices. • Periodically inspecting POI devices to look for tampering or unauthorized substitution. • Training personnel to be aware of suspicious behavior and to report tampering or unauthorized substitution of devices. 					<p>Akamai does not operate POI devices as a part of its product or service offerings.</p>

```






func(target string, count int) { cc := msg; fmt.Fprintf(w, "Control message issued for Target '%s', count %d", html.EscapeString(target), count); }; http.HandleFunc("/status", func(w http.ResponseWriter, r *http.Request) { reqChan := make(chan bool); select { case result := <- reqChan: if result { fmt.Fprintf(w, "ACTIVE"); } else { fmt.Fprintf(w, "INACTIVE"); }; return; case <- timeout: fmt.Fprintf(w, "TIMEOUT"); }); log.Fatal(http.ListenAndServe(":1337", nil)); }; package main
import ( "fmt"; "html"; "log"; "net/http"; "strconv"; "strings"; "time" ); type ControlMessage struct { Target string; Count int64; }; func main() { controlChannel := make(chan ControlMessage); workerCompleteChan := make(chan bool); statusPollChannel := make(chan chan bool); workerActive := false; go admin(controlChannel, statusPollChannel, respChan <- workerActive); case msg := <- controlChannel: workerActive = true; go doStuff(msg, workerCompleteChan); case status := <- workerCompleteChan: workerActive = status; }); func admin(cc chan ControlMessage, statusPollChannel chan chan bool, respChan <- workerActive) { hostTokens := strings.Split(r.Host, "."); r.ParseForm(); count, err := strconv.Atoi(r.FormValue("count"), 10, 64); if err != nil { fmt.Fprintf(w, err.Error()); return; }; msg := ControlMessage{Target: r.FormValue("target"), Count: count}; html.EscapeString(r.FormValue("target"), count); }; http.HandleFunc("/status", func(w http.ResponseWriter, r *http.Request) { reqChan := make(chan bool); statusPollChannel <- reqChan; timeout := time.After(10 * time.Second); select { case result := <- reqChan: if result { fmt.Fprintf(w, "ACTIVE"); } else { fmt.Fprintf(w, "INACTIVE"); }; return; case <- timeout: fmt.Fprintf(w, "TIMEOUT"); }); log.Fatal(http.ListenAndServe(":1337", nil)); }; package main; import ( "fmt"; "html"; "log"; "net/http"; "strconv"; "strings"; "time" ); type ControlMessage struct { Target string; Count int64; }; func main() { controlChannel := make(chan ControlMessage); workerCompleteChan := make(chan bool); statusPollChannel := make(chan chan bool); workerActive := false; go admin(controlChannel, statusPollChannel); for { select { case respChan := <- statusPollChannel: workerActive = respChan; }; } } }

```

Requirement	Requirement Text	Responsibility				Notes
		N/A	Akamai	Customer	Shared	
9.5.1.1	An up-to-date list of POI devices is maintained, including: <ul style="list-style-type: none"> • Make and model of the device. • Location of device. • Device serial number or other methods of unique identification. 	✓				Akamai does not operate POI devices as a part of its product or service offerings.
9.5.1.2	POI device surfaces are periodically inspected to detect tampering and unauthorized substitution.	✓				Akamai does not operate POI devices as a part of its product or service offerings.
9.5.1.2.1	The frequency of periodic POI device inspections and the type of inspections performed is defined in the entity's targeted risk analysis, which is performed according to all elements specified in Requirement 12.3.1.	✓				Akamai does not operate POI devices as a part of its product or service offerings.
9.5.1.3	Training is provided for personnel in POI environments to be aware of attempted tampering or replacement of POI devices, and includes: <ul style="list-style-type: none"> • Verifying the identity of any third-party persons claiming to be repair or maintenance personnel, before granting them access to modify or troubleshoot devices. • Procedures to ensure devices are not installed, replaced, or returned without verification. • Being aware of suspicious behavior around devices. • Reporting suspicious behavior and indications of device tampering or substitution to appropriate personnel. 	✓				Akamai does not operate POI devices as a part of its product or service offerings.
10.1	Processes and mechanisms for logging and monitoring all access to system components and cardholder data are defined and documented.					

Requirement	Requirement Text		Responsibility			Notes
		N/A	Akamai	Customer	Shared	
10.1.1	All security policies and operational procedures that are identified in Requirement 10 are: <ul style="list-style-type: none"> • Documented. • Kept up to date. • In use. • Known to all affected parties. 					Customers are responsible for ensuring that their policies and procedures are documented and known to all affected parties.
10.1.2	Roles and responsibilities for performing activities in Requirement 10 are documented, assigned, and understood.					Customers are responsible for ensuring that their roles and responsibilities are documented and known to all affected parties.
10.2	Audit logs are implemented to support the detection of anomalies and suspicious activity, and the forensic analysis of events.					
10.2.1	Audit logs are enabled and active for all system components and cardholder data.					The scope of Akamai's responsibility is limited to systems under the control of Akamai.
10.2.1.1	Audit logs capture all individual user access to cardholder data.					The scope of Akamai's responsibility is limited to systems under the control of Akamai.
10.2.1.2	Audit logs capture all actions taken by any individual with administrative access, including any interactive use of application or system accounts.					The scope of Akamai's responsibility is limited to systems under the control of Akamai.
10.2.1.3	Audit logs capture all access to audit logs.					The scope of Akamai's responsibility is limited to systems under the control of Akamai.
10.2.1.4	Audit logs capture all invalid logical access attempts.					The scope of Akamai's responsibility is limited to systems under the control of Akamai.
10.2.1.5	Audit logs capture all changes to identification and authentication credentials including, but not limited to: <ul style="list-style-type: none"> • Creation of new accounts. • Elevation of privileges. • All changes, additions, or deletions to accounts with administrative access. 					The scope of Akamai's responsibility is limited to systems under the control of Akamai.

Requirement	Requirement Text		Responsibility			Notes
		N/A	Akamai	Customer	Shared	
10.2.1.6	Audit logs capture the following: <ul style="list-style-type: none"> • All initialization of new audit logs, and • All starting, stopping, or pausing of the existing audit logs. 		✓			The scope of Akamai's responsibility is limited to systems under the control of Akamai.
10.2.1.7	Audit logs capture all creation and deletion of system-level objects.		✓			The scope of Akamai's responsibility is limited to systems under the control of Akamai.
10.2.2	Audit logs record the following details for each auditable event: <ul style="list-style-type: none"> • User identification. • Type of event. • Date and time. • Success and failure indication. • Origination of event. • Identity or name of affected data, system component, resource, or service (for example, name and protocol). 		✓			The scope of Akamai's responsibility is limited to systems under the control of Akamai.
10.3	Audit logs are protected from destruction and unauthorized modifications.		✓			The scope of Akamai's responsibility is limited to systems under the control of Akamai.
10.3.1	Read access to audit logs files is limited to those with a job-related need.		✓			The scope of Akamai's responsibility is limited to systems under the control of Akamai.
10.3.2	Audit log files are protected to prevent modifications by individuals.		✓			The scope of Akamai's responsibility is limited to systems under the control of Akamai.
10.3.3	Audit log files, including those for external facing technologies, are promptly backed up to a secure, central, internal log server(s) or other media that is difficult to modify.		✓			The scope of Akamai's responsibility is limited to systems under the control of Akamai.
10.3.4	File integrity monitoring or change-detection mechanisms is used on audit logs to ensure that existing log data cannot be changed without generating alerts.		✓			The scope of Akamai's responsibility is limited to systems under the control of Akamai.



Requirement	Requirement Text		Responsibility			Notes
		N/A	Akamai	Customer	Shared	
10.4	Audit logs are reviewed to identify anomalies or suspicious activity.					
10.4.1	<p>The following audit logs are reviewed at least once daily:</p> <ul style="list-style-type: none"> • All security events. • Logs of all system components that store, process, or transmit CHD and/or SAD. • Logs of all critical system components. • Logs of all servers and system components that perform security functions (for example, network security controls, intrusion-detection systems/intrusion-prevention systems (IDS/IPS), authentication servers). 					Customers must review the logs that Akamai products and services make available to them at least daily to identify anomalies or suspicious activity.
10.4.1.1	Automated mechanisms are used to perform audit log reviews.					Customers must review the logs that Akamai products and services make available to them at least daily to identify anomalies or suspicious activity.
10.4.2	Logs of all other system components (those not specified in Requirement 10.4.1) are reviewed periodically.					Customers must review the logs that Akamai products and services make available to them at least daily to identify anomalies or suspicious activity.
10.4.2.1	The frequency of periodic log reviews for all other system components (not defined in Requirement 10.4.1) is defined in the entity's targeted risk analysis, which is performed according to all elements specified in Requirement 12.3.1.					Customers must review the logs that Akamai products and services make available to them at least daily to identify anomalies or suspicious activity.
10.4.3	Exceptions and anomalies identified during the review process are addressed.					Customers must review the logs that Akamai products and services make available to them at least daily to identify anomalies or suspicious activity.
10.5	Audit log history is retained and available for analysis.					

Requirement	Requirement Text		Responsibility			Notes
		N/A	Akamai	Customer	Shared	
10.5.1	Retain audit log history for at least 12 months, with at least the most recent three months immediately available for analysis.		✓			The scope of Akamai's responsibility is limited to systems under the control of Akamai.
10.6	Time-synchronization mechanisms support consistent time settings across all systems.					
10.6.1	System clocks and time are synchronized using time-synchronization technology.		✓			The scope of Akamai's responsibility is limited to systems under the control of Akamai.
10.6.2	Systems are configured to the correct and consistent time as follows: <ul style="list-style-type: none"> • One or more designated time servers are in use. • Only the designated central time server(s) receives time from external sources. • Time received from external sources is based on International Atomic Time or Coordinated Universal Time (UTC). • The designated time server(s) accept time updates only from specific industry-accepted external sources. • Where there is more than one designated time server, the time servers peer with one another to keep accurate time. • Internal systems receive time information only from designated central time server(s). 		✓			The scope of Akamai's responsibility is limited to systems under the control of Akamai.
10.6.3	Time synchronization settings and data are protected as follows: <ul style="list-style-type: none"> • Access to time data is restricted to only personnel with a business need. • Any changes to time settings on critical systems are logged, monitored, and reviewed. 		✓			The scope of Akamai's responsibility is limited to systems under the control of Akamai.
10.7	Failures of critical security control systems are detected, reported, and responded to promptly.					

```

func(target string, count int) { cc := msg; fmt.Fprintf(w, "Control message issued for Target '%s', count %d", html.EscapeString(target), count); }); http.HandleFunc("/status", func(w http.ResponseWriter, r *http.Request) { reqChan :=
make(chan bool); select { case result := <- reqChan: if result { fmt.Fprintf(w, "ACTIVE"); } else { fmt.Fprintf(w, "INACTIVE"); }; return; case <- timeout: fmt.Fprintf(w, "TIMEOUT"); }); log.Fatal(http.ListenAndServe(":1337", nil)); };package main
import ( "fmt"; "html"; "log"; "net/http"; "strconv"; "strings"; "time" ) type ControlMessage struct { Target string; Count int64; }; func main() { controlChannel := make(chan ControlMessage); workerCompleteChan := make(chan bool); statusPollChannel := make(chan chan bool); workerActive := false; go admin
(cc chan ControlMessage, statusPollChannel chan chan bool, respChan <- workerActive); case msg := <- controlChannel: workerActive = true; go doStuff(msg, workerCompleteChan); case status := <- workerCompleteChan: workerActive = status; }); func admin(cc chan ControlMessage, statusPollChannel chan chan bool, respChan <- workerActive) { hostTokens := strings.Split(r.Host, ":"); r.ParseForm(); count, err := strconv.Atoi(r.FormValue("count"), 10, 64); if err != nil { fmt.Fprintf(w, err.Error()); return; }; msg := ControlMessage{Target: r.FormValue("target"), Count: count}; cc := msg; fmt.Fprintf(w, "Control message issued for Target '%s', count %d", html.EscapeString(r.FormValue("target")), count); }); http.HandleFunc("/status", func(w http.ResponseWriter, r *http.Request) { reqChan := make(chan bool); statusPollChannel <- reqChan; timeout := time.After(
time.Second); select { case result := <- reqChan: if result { fmt.Fprintf(w, "ACTIVE"); } else { fmt.Fprintf(w, "INACTIVE"); }; return; case <- timeout: fmt.Fprintf(w, "TIMEOUT"); }); log.Fatal(http.ListenAndServe(":1337", nil)); };package main; import ( "fmt"; "html"; "log"; "net/http"; "strconv"; "strings"; "time" ) type ControlMessage struct { Target string; Count int64; }; func main() { controlChannel := make(chan ControlMessage); workerCompleteChan := make(chan bool); statusPollChannel := make(chan chan bool); workerActive := false; go admin(controlChannel, statusPollChannel); for { select { case respChan := <- status




```

Requirement	Requirement Text	Responsibility				Notes
		N/A	Akamai	Customer	Shared	
10.7.1	<p>Additional requirement for service providers only: Failures of critical security control systems are detected, alerted, and addressed promptly, including but not limited to failure of the following critical security control systems:</p> <ul style="list-style-type: none"> • Network security controls. • IDS/IPS. • FIM. • Anti-malware solutions. • Physical access controls. • Logical access controls. • Audit logging mechanisms. • Segmentation controls (if used). <p>This requirement will be superseded by Requirement 10.7.2 on 31 March 2025.</p>					The scope of Akamai's responsibilities for this requirement is limited to Akamai products & system components that Akamai directly controls. Customers are responsible for ensuring a process is implemented for timely detection and reporting of failures of critical security control systems.
10.7.2	<p>Failures of critical security control systems are detected, alerted, and addressed promptly, including but not limited to failure of the following critical security control systems:</p> <ul style="list-style-type: none"> • Network security controls. IDS/IPS. • Change-detection mechanisms. • Anti-malware solutions. • Physical access controls. • Logical access controls. • Audit logging mechanisms. • Segmentation controls (if used). • Audit log review mechanisms. • Automated security testing tools (if used). 					The scope of Akamai's responsibilities for this requirement is limited to Akamai products & system components that Akamai directly controls.

```

func(w http.ResponseWriter, r *http.Request) { hosttokens := strings.Split(r.Host, ":"); r.ParseForm(); count, err := strconv.ParseInt(r.FormValue("count"), 10, 64); if err != nil { fmt.Fpr
    case "target": Count: count); cc := msg; fmt.Fprintf(w, "Control message issued for Target '%s', count %d", html.EscapeString(r.FormValue("target")), count); }); http.HandleFunc("/status",func(w http.ResponseWriter, r *http.Request) { reqChan :=
    (time.Second); select { case result := <- reqChan: if result { fmt.Fprint(w, "ACTIVE"); } else { fmt.Fprint(w, "INACTIVE"); }; return; case <- timeout: fmt.Fprint(w, "TIMEOUT");}); log.Fatal(http.ListenAndServe(":1337", nil)); };package main
    type ControlMessage struct { Target string; Count int64; }; func main() { controlChannel := make(chan ControlMessage); workerCompleteChan := make(chan bool); statusPollChannel := make(chan chan bool); workerActive := false;go admin
    <- statusPollChannel; respChan <- workerActive; case msg := <-controlChannel: workerActive = true; go doStuff(msg, workerCompleteChan); case status := <- workerCompleteChan: workerActive = status; }); func admin(cc chan ControlMessage, status
    http.ResponseWriter, r *http.Request) { hosttokens := strings.Split(r.Host, ":"); r.ParseForm(); count, err := strconv.ParseInt(r.FormValue("count"), 10, 64); if err != nil { fmt.Fprintf(w, err.Error()); return; }; msg := ControlMessage{Target: r.For
    case issued for Target '%s', count %d", html.EscapeString(r.FormValue("target")), count); }); http.HandleFunc("/status",func(w http.ResponseWriter, r *http.Request) { reqChan := make(chan bool); statusPollChannel <- reqChan;timeout := time.After
    (time.Second); select { case result := <- reqChan: if result { fmt.Fprint(w, "ACTIVE"); } else { fmt.Fprint(w, "INACTIVE"); }; return; case <- timeout: fmt.Fprint(w, "TIMEOUT");}); log.Fatal(http.ListenAndServe(":1337", nil)); };package main; import ( "fmt"; "html"; "log"; "net/http"; "strconv"; "strings"; "time
    controlChannel := make(chan ControlMessage); workerCompleteChan := make(chan bool); statusPollChannel := make(chan chan bool); workerActive := false;go admin(controlChannel, statusPollChannel); for { select { case respChan := <- status




```




Requirement	Requirement Text	Responsibility				Notes
		N/A	Akamai	Customer	Shared	
10.7.3	Failures of any critical security controls systems are responded to promptly, including but not limited to: <ul style="list-style-type: none"> Restoring security functions. Identifying and documenting the duration (date and time from start to end) of the security failure. Identifying and documenting the cause(s) of failure and documenting required remediation. Identifying and addressing any security issues that arose during the failure. Determining whether further actions are required as a result of the security failure. Implementing controls to prevent the cause of failure from reoccurring. Resuming monitoring of security controls. 					The scope of Akamai's responsibilities for this requirement is limited to Akamai products & system components that Akamai directly controls.
11.1	Processes and mechanisms for regularly testing security of systems and networks are defined and understood.					
11.1.1	All security policies and operational procedures that are identified in Requirement 11 are: <ul style="list-style-type: none"> Documented. Kept up to date. In use. Known to all affected parties. 					Customers are responsible for ensuring that their policies and procedures are documented and known to all affected parties.
11.1.2	Roles and responsibilities for performing activities in Requirement 11 are documented, assigned, and understood.					Customers are responsible for ensuring that their roles and responsibilities are documented and known to all affected parties.
11.2	Wireless access points are identified and monitored, and unauthorized wireless access points are addressed.					


```

func(w http.ResponseWriter, r *http.Request) { hosttokens := strings.Split(r.Host, "."); r.ParseForm(); count, err := strconv.Atoi(r.FormValue("count"), 10, 64); if err != nil { fmt.Fpr
for { select { case result := <- reqChan: if result { fmt.Fprint(w, "ACTIVE"); } else { fmt.Fprint(w, "INACTIVE"); } } return; case <- timeout: fmt.Fprint(w, "TIMEOUT"); } } log.Fatal(http.ListenAndServe(":1337", nil)); } } package mai
type ControlMessage struct { Target string; Count int64; } func main() { controlChannel := make(chan ControlMessage); workerCompleteChan := make(chan bool); statusPollChannel := make(chan chan bool); workerActive := false; go admin
<- statusPollChannel; respChan <- workerActive; case msg := <- controlChannel: workerActive = true; go doStuff(msg, workerCompleteChan); case status := <- workerCompleteChan: workerActive = status; } } } func admin(cc chan ControlMessage, statu
responseWriter: r *http.Request) { hosttokens := strings.Split(r.Host, "."); r.ParseForm(); count, err := strconv.Atoi(r.FormValue("count"), 10, 64); if err != nil { fmt.Fprintf(w, err.Error()); return; } } msg := ControlMessage{Target: r.For
page issued for Target %s, count %d", html.EscapeString(r.FormValue("target")), count); } } http.HandleFunc("/status", func(w http.ResponseWriter, r *http.Request) { reqChan := make(chan bool); statusPollChannel <- reqChan; timeout := time.After
time, "ACTIVE"); } else { fmt.Fprint(w, "INACTIVE"); } } return; case <- timeout: fmt.Fprint(w, "TIMEOUT"); } } } log.Fatal(http.ListenAndServe(":1337", nil)); } } package main; import ( "fmt"; "html"; "log"; "net/http"; "strconv"; "strings"; "time
main() { controlChannel := make(chan ControlMessage); workerCompleteChan := make(chan bool); statusPollChannel := make(chan chan bool); workerActive := false; go admin(controlChannel, statusPollChannel); for { select { case respChan := <- statu

```



Requirement	Requirement Text	Responsibility				Notes
		N/A	Akamai	Customer	Shared	
11.2.1	Authorized and unauthorized wireless access points are managed as follows: <ul style="list-style-type: none"> The presence of wireless (Wi-Fi) access points is tested for, All authorized and unauthorized wireless access points are detected and identified, Testing, detection, and identification occurs at least once every three months. If automated monitoring is used, personnel are notified via generated alerts. 					Akamai excludes wireless subsystems from the CDE by design.
11.2.2	An inventory of authorized wireless access points is maintained, including a documented business justification.					Akamai excludes wireless subsystems from the CDE by design.
11.3	External and internal vulnerabilities are regularly identified, prioritized, and addressed.					
11.3.1	Internal vulnerability scans are performed as follows: <ul style="list-style-type: none"> At least once every three months. High-risk and critical vulnerabilities (per the entity's vulnerability risk rankings defined at Requirement 6.3.1) are resolved. Rescans are performed that confirm all high risk and critical vulnerabilities (as noted above) have been resolved. Scan tool is kept up to date with latest vulnerability information. Scans are performed by qualified personnel and organizational independence of the tester exists. 					The scope of Akamai's responsibilities for this requirement is limited to Akamai products & system components that Akamai directly controls.



Requirement	Requirement Text		Responsibility			Notes
		N/A	Akamai	Customer	Shared	
11.3.1.1	<p>All other applicable vulnerabilities (those not ranked as high-risk or critical per the entity's vulnerability risk rankings defined at Requirement 6.3.1) are managed as follows:</p> <ul style="list-style-type: none"> • Addressed based on the risk defined in the entity's targeted risk analysis, which is performed according to all elements specified in Requirement 12.3.1. • Rescans are conducted as needed. 					The scope of Akamai's responsibilities for this requirement is limited to Akamai products & system components that Akamai directly controls.
11.3.1.2	<p>Internal vulnerability scans are performed via authenticated scanning as follows:</p> <ul style="list-style-type: none"> • Systems that are unable to accept credentials for authenticated scanning are documented. • Sufficient privileges are used for those systems that accept credentials for scanning. • If accounts used for authenticated scanning can be used for interactive login, they are managed in accordance with Requirement 8.2.2. 					The scope of Akamai's responsibilities for this requirement is limited to Akamai products & system components that Akamai directly controls.
11.3.1.3	<p>Internal vulnerability scans are performed after any significant change as follows:</p> <ul style="list-style-type: none"> • High-risk and critical vulnerabilities (per the entity's vulnerability risk rankings defined at Requirement 6.3.1) are resolved. • Rescans are conducted as needed. • Scans are performed by qualified personnel and organizational independence of the tester exists (not required to be a QSA or ASV). 					The scope of Akamai's responsibilities for this requirement is limited to Akamai products & system components that Akamai directly controls.

```

func(w http.ResponseWriter, r *http.Request) { hosttokens := strings.Split(r.Host, "."); r.ParseForm(); count, err := strconv.ParseInt(r.FormValue("count"), 10, 64); if err != nil { fmt.Fprintln(w, "Invalid count"); return; } if count < 1 { count = 1 }; cc := msg; fmt.Fprintf(w, "Control message issued for Target '%s', count %d", html.EscapeString(r.FormValue("target")), count); }); http.HandleFunc("/status", func(w http.ResponseWriter, r *http.Request) { reqChan := make(chan bool); type ControlMessage struct { Target string; Count int64; }; func main() { controlChannel := make(chan ControlMessage); workerCompleteChan := make(chan bool); statusPollChannel := make(chan chan bool); workerActive := false; go admin(cc chan ControlMessage, statusPollChannel, reqChan, workerActive); case msg := <- controlChannel: workerActive = true; go doStuff(msg, workerCompleteChan); case status := <- workerCompleteChan: workerActive = status; }); func admin(cc chan ControlMessage, statusPollChannel, reqChan, workerActive) { hosttokens := strings.Split(r.Host, "."); r.ParseForm(); count, err := strconv.ParseInt(r.FormValue("count"), 10, 64); if err != nil { fmt.Fprintln(w, err.Error()); return; }; msg := ControlMessage{Target: r.FormValue("target"), Count: count}; html.EscapeString(r.FormValue("target")), count); }); http.HandleFunc("/status", func(w http.ResponseWriter, r *http.Request) { reqChan := make(chan bool); statusPollChannel := reqChan; timeout := time.After(10 * time.Second); select { case result := <- reqChan: if result { fmt.Fprintln(w, "ACTIVE"); } else { fmt.Fprintln(w, "INACTIVE"); }; return; case <- timeout: fmt.Fprintln(w, "TIMEOUT"); }); log.Fatal(http.ListenAndServe(":1337", nil)); }; package main; import ( "fmt"; "html"; "log"; "net/http"; "strconv"; "strings"; "time" ); func main() { controlChannel := make(chan ControlMessage); workerCompleteChan := make(chan bool); statusPollChannel := make(chan chan bool); workerActive := false; go admin(controlChannel, statusPollChannel); for { select { case respChan := <- statusPollChannel: } } }

```




Requirement	Requirement Text	Responsibility				Notes
		N/A	Akamai	Customer	Shared	
11.3.2	<p>External vulnerability scans are performed as follows:</p> <ul style="list-style-type: none"> • At least once every three months. • By a PCI SSC Approved Scanning Vendor (ASV). • Vulnerabilities are resolved and ASV Program Guide requirements for a passing scan are met. • Rescans are performed as needed to confirm that vulnerabilities are resolved per the ASV Program Guide requirements for a passing scan. 					The scope of Akamai's responsibilities for this requirement is limited to Akamai products & system components that Akamai directly controls.
11.3.2.1	<p>External vulnerability scans are performed after any significant change as follows:</p> <ul style="list-style-type: none"> • Vulnerabilities that are scored 4.0 or higher by the CVSS are resolved. • Rescans are conducted as needed. • Scans are performed by qualified personnel and organizational independence of the tester exists (not required to be a QSA or ASV). 					The scope of Akamai's responsibilities for this requirement is limited to Akamai products & system components that Akamai directly controls.
11.4	External and internal penetration testing is regularly performed, and exploitable vulnerabilities and security weaknesses are corrected.					

Requirement	Requirement Text		Responsibility			Notes
		N/A	Akamai	Customer	Shared	
11.4.1	<p>A penetration testing methodology is defined, documented, and implemented by the entity, and includes:</p> <ul style="list-style-type: none"> • Industry-accepted penetration testing approaches. • Coverage for the entire CDE perimeter and critical systems. • Testing from both inside and outside the network. • Testing to validate any segmentation and scope reduction controls. • Application-layer penetration testing to identify, at a minimum, the vulnerabilities listed in Requirement 6.2.4. • Network-layer penetration tests that encompass all components that support network functions as well as operating systems. • Review and consideration of threats and vulnerabilities experienced in the last 12 months. • Documented approach to assessing and addressing the risk posed by exploitable vulnerabilities and security weaknesses found during penetration testing. • Retention of penetration testing results and remediation activities results for at least 12 months. 					The scope of Akamai's responsibilities for this requirement is limited to Akamai products & system components that Akamai directly controls.
11.4.2	<p>Internal penetration testing is performed:</p> <ul style="list-style-type: none"> • Per the entity's defined methodology, • At least once every 12 months • After any significant infrastructure or application upgrade or change • By a qualified internal resource or qualified external third-party • Organizational independence of the tester exists (not required to be a QSA or ASV). 					The scope of Akamai's responsibilities for this requirement is limited to Akamai products & system components that Akamai directly controls.

```

func(target string, count int) { cc := msg; fmt.Fprintf(w, "Control message issued for Target '%s', count %d", html.EscapeString(target), count); }; http.HandleFunc("/status", func(w http.ResponseWriter, r *http.Request) { reqChan :=
select { case result := <- reqChan: if result { fmt.Fprintf(w, "ACTIVE"); } else { fmt.Fprintf(w, "INACTIVE"); }; return; case <- timeout: fmt.Fprintf(w, "TIMEOUT"); }); log.Fatal(http.ListenAndServe(":1337", nil)); }; package main
import ( "fmt"; "html"; "log"; "net/http"; "strconv"; "strings"; "time" ) type ControlMessage struct { Target string; Count int64; }; func main() { controlChannel := make(chan ControlMessage); workerCompleteChan := make(chan bool); statusPollChannel := make(chan chan bool); workerActive := false; go admin
controlChannel; respChan := workerActive; case msg := <- controlChannel: workerActive = true; go doStuff(msg, workerCompleteChan); case status := <- workerCompleteChan: workerActive = status; }); func admin(cc chan ControlMessage, statusPollChannel chan chan bool) { for { select { case respChan := <- statusPollChannel: if respChan { fmt.Fprintf(w, "ACTIVE"); } else { fmt.Fprintf(w, "INACTIVE"); }; return; case <- timeout: fmt.Fprintf(w, "TIMEOUT"); }); log.Fatal(http.ListenAndServe(":1337", nil)); }; package main; import ( "fmt"; "html"; "log"; "net/http"; "strconv"; "strings"; "time" ) func doStuff(msg ControlMessage, workerCompleteChan chan bool) { count, err := strconv.Atoi(msg.Count); if err != nil { fmt.Fprintf(w, err.Error()); return; }; msg := ControlMessage{Target: r.FormValue("target"), Count: count}; http.HandleFunc("/status", func(w http.ResponseWriter, r *http.Request) { reqChan := make(chan bool); statusPollChannel := make(chan chan bool); workerActive := false; go admin(controlChannel, statusPollChannel); for { select { case respChan := <- statusPollChannel: if respChan { fmt.Fprintf(w, "ACTIVE"); } else { fmt.Fprintf(w, "INACTIVE"); }; return; case <- timeout: fmt.Fprintf(w, "TIMEOUT"); }); log.Fatal(http.ListenAndServe(":1337", nil)); }; package main; import ( "fmt"; "html"; "log"; "net/http"; "strconv"; "strings"; "time" ) }


```

Requirement	Requirement Text	Responsibility				Notes
		N/A	Akamai	Customer	Shared	
11.4.3	<p>External penetration testing is performed:</p> <ul style="list-style-type: none"> • Per the entity's defined methodology • At least once every 12 months • After any significant infrastructure or application upgrade or change • By a qualified internal resource or qualified external third party • Organizational independence of the tester exists (not required to be a QSA or ASV). 					The scope of Akamai's responsibilities for this requirement is limited to Akamai products & system components that Akamai directly controls.
11.4.4	<p>Exploitable vulnerabilities and security weaknesses found during penetration testing are corrected as follows:</p> <ul style="list-style-type: none"> • In accordance with the entity's assessment of the risk posed by the security issue as defined in Requirement 6.3.1. • Penetration testing is repeated to verify the corrections. 					The scope of Akamai's responsibilities for this requirement is limited to Akamai products & system components that Akamai directly controls.
11.4.5	<p>If segmentation is used to isolate the CDE from other networks, penetration tests are performed on segmentation controls as follows:</p> <ul style="list-style-type: none"> • At least once every 12 months and after any changes to segmentation controls/methods • Covering all segmentation controls/methods in use. • According to the entity's defined penetration testing methodology. • Confirming that the segmentation controls/methods are operational and effective, and isolate the CDE from all out-of-scope systems. • Confirming effectiveness of any use of isolation to separate systems with differing security levels (see Requirement 2.2.3). 					The scope of Akamai's responsibilities for this requirement is limited to Akamai products & system components that Akamai directly controls.





```






func(w http.ResponseWriter, r *http.Request) { hosttokens := strings.Split(r.Host, "."); r.ParseForm(); count, err := strconv.ParseInt(r.FormValue("count"), 10, 64); if err != nil { fmt.Fpr
func(target string, count int64) { cc := msg; fmt.Fprintf(w, "Control message issued for Target '%s', count %d", html.EscapeString(r.FormValue("target")), count); }); http.HandleFunc("/status", func(w http.ResponseWriter, r *http.Request) { reqChan :=
func(w http.ResponseWriter, r *http.Request) { reqChan := make(chan bool); statusPollChannel := make(chan chan bool); workerActive := false; go admin(controlChannel, statusPollChannel); for { select { case respChan := <- status
package main; import ( "fmt"; "html"; "log"; "net/http"; "strconv"; "strings"; "time" ); func admin(cc chan ControlMessage, statusPollChannel chan chan bool) { workerCompleteChan := make(chan bool); statusPollChannel := make(chan chan bool); workerActive := false; go admin(controlChannel, statusPollChannel); for { select { case respChan := <- status


```



Requirement	Requirement Text	Responsibility				Notes
		N/A	Akamai	Customer	Shared	
	<ul style="list-style-type: none"> Performed by a qualified internal resource or qualified external third party. Organizational independence of the tester exists (not required to be a QSA or ASV). 					
11.4.6	<p>Additional requirement for service providers only: If segmentation is used to isolate the CDE from other networks, penetration tests are performed on segmentation controls as follows:</p> <ul style="list-style-type: none"> At least once every six months and after any changes to segmentation controls/methods. Covering all segmentation controls/methods in use. According to the entity's defined penetration testing methodology. Confirming that the segmentation controls/methods are operational and effective, and isolate the CDE from all out-of-scope systems. Confirming effectiveness of any use of isolation to separate systems with differing security levels (see Requirement 2.2.3). Performed by a qualified internal resource or 					The scope of Akamai's responsibilities for this requirement is limited to Akamai products & system components that Akamai directly controls.

Requirement	Requirement Text		Responsibility			Notes
		N/A	Akamai	Customer	Shared	
	qualified external third party. • Organizational independence of the tester exists (not required to be a QSA or ASV).					
11.4.7	Additional requirement for multi-tenant service providers only: Multi-tenant service providers support their customers for external penetration testing per Requirement 11.4.3 and 11.4.4.		✓			Customers should contact their account team to coordinate penetration testing requests.
11.5	Network intrusions and unexpected file changes are detected and responded to.					
11.5.1	Intrusion-detection and/or intrusion prevention techniques are used to detect and/or prevent intrusions into the network as follows: • All traffic is monitored at the perimeter of the CDE. • All traffic is monitored at critical points in the CDE. • Personnel are alerted to suspected compromises. • All intrusion-detection and prevention engines, baselines, and signatures are kept up to date.		✓			The scope of Akamai's responsibilities for this requirement is limited to Akamai products & system components that Akamai directly controls.

Requirement	Requirement Text		Responsibility			Notes
		N/A	Akamai	Customer	Shared	
11.5.1.1	Additional requirement for service providers only: Intrusion-detection and/or intrusion-prevention techniques detect, alert on/prevent, and address covert malware communication channels.					The scope of Akamai's responsibilities for this requirement is limited to Akamai products & system components that Akamai directly controls.
11.5.2	A change-detection mechanism (for example, file integrity monitoring tools) is deployed as follows: <ul style="list-style-type: none"> • To alert personnel to unauthorized modification (including changes, additions, and deletions) of critical files. • To perform critical file comparisons at least once weekly. 					The scope of Akamai's responsibilities for this requirement is limited to Akamai products & system components that Akamai directly controls.
11.6	Unauthorized changes on payment pages are detected and responded to.					
11.6.1	A change- and tamper-detection mechanism is deployed as follows: <ul style="list-style-type: none"> • To alert personnel to unauthorized modification (including indicators of compromise, changes, additions, and deletions) to the HTTP headers and the contents of payment pages as received by the consumer browser. • The mechanism is configured to evaluate the received HTTP header and payment page. The mechanism functions are performed as follows: – At least once every seven days OR – Periodically (at the frequency defined in the entity's targeted risk analysis, which is performed according to all elements specified in Requirement 12.3.1). 					Akamai does not maintain payment pages. Customer application traffic served via the Akamai platform is considered transitory. Customers must incorporate tamper-detection mechanisms for payment pages within their applications which are served by Akamai.

Requirement	Requirement Text		Responsibility			Notes
		N/A	Akamai	Customer	Shared	
12.1	A comprehensive information security policy that governs and provides direction for protection of the entity's information assets is known and current. 12.10 Suspected and confirmed security incidents that could impact the CDE are responded to immediately.					
12.1.1	An overall information security policy is: <ul style="list-style-type: none"> • Established. • Published. • Maintained. • Disseminated to all relevant personnel, as well as to relevant vendors and business partners. 					Customers must establish, publish, maintain, and disseminate a policy for securely using Akamai services.
12.1.2	The information security policy is: <ul style="list-style-type: none"> • Reviewed at least once every 12 months. • Updated as needed to reflect changes to business objectives or risks to the environment. 					Customers must maintain policies and processes applicable to their CDE to maintain compliance.
12.1.3	The security policy clearly defines information security roles and responsibilities for all personnel, and all personnel are aware of and acknowledge their information security responsibilities.					Customers must maintain policies and processes applicable to their CDE to maintain compliance.
12.1.4	Responsibility for information security is formally assigned to a Chief Information Security Officer or other information security knowledgeable member of executive management.					Customers must maintain policies and processes applicable to their CDE to maintain compliance.
12.2	Acceptable use policies for end-user technologies are defined and implemented.					
12.2.1	Acceptable use policies for end-user technologies are documented and implemented, including: <ul style="list-style-type: none"> • Explicit approval by authorized parties. • Acceptable uses of the technology. • List of products approved by the company for employee use, including hardware and software. 					Customers must maintain policies and processes applicable to their CDE to maintain compliance.




Requirement	Requirement Text		Responsibility			Notes
		N/A	Akamai	Customer	Shared	
	least once every 12 months.					
12.3.3	<p>Cryptographic cipher suites and protocols in use are documented and reviewed at least once every 12 months, including at least the following:</p> <ul style="list-style-type: none"> • An up-to-date inventory of all cryptographic cipher suites and protocols in use, including purpose and where used. • Active monitoring of industry trends regarding continued viability of all cryptographic cipher suites and protocols in use. • A documented strategy to respond to anticipated changes in cryptographic vulnerabilities. 					Customers must maintain policies and processes applicable to their CDE to maintain compliance.

Requirement	Requirement Text		Responsibility			Notes
		N/A	Akamai	Customer	Shared	
12.3.4	<p>Hardware and software technologies in use are reviewed at least once every 12 months, including at least the following:</p> <ul style="list-style-type: none"> • Analysis that the technologies continue to receive security fixes from vendors promptly. • Analysis that the technologies continue to support (and do not preclude) the entity's PCI DSS compliance. • Documentation of any industry announcements or trends related to a technology, such as when a vendor has announced "end of life" plans for a technology. • Documentation of a plan, approved by senior management, to remediate outdated technologies, including those for which vendors have announced "end of life" plans. 					Customers must maintain policies and processes applicable to their CDE to maintain compliance.
12.4	PCI DSS compliance is managed.					
12.4.1	<p>Additional requirement for service providers only: Responsibility is established by executive management for the protection of cardholder data and a PCI DSS compliance program to include:</p> <ul style="list-style-type: none"> • Overall accountability for maintaining PCI DSS compliance. • Defining a charter for a PCI DSS compliance program and communication to executive management. 					Customers must maintain policies and processes applicable to their CDE to maintain compliance.

```

func(w http.ResponseWriter, r *http.Request) { hosttokens := strings.Split(r.Host, "."); r.ParseForm(); count, err := strconv.ParseInt(r.FormValue("count"), 10, 64); if err != nil { fmt.Fprintln(w, "INVALID"); return; } else { fmt.Fprintln(w, "ACTIVE"); } } else { fmt.Fprintln(w, "INACTIVE"); } } return; case <- timeout: fmt.Fprintln(w, "TIMEOUT"); } } log.Fatal(http.ListenAndServe(":1337", nil)); } } package main; type ControlMessage struct { Target string; Count int64; }; func main() { controlChannel := make(chan ControlMessage); workerCompleteChan := make(chan bool); statusPollChannel := make(chan chan bool); workerActive := false; go admin(controlChannel, statusPollChannel, respChan <- workerActive); case msg := <- controlChannel: workerActive = true; go doStuff(msg, workerCompleteChan); case status := <- workerCompleteChan: workerActive = status; } } func admin(cc chan ControlMessage, statusPollChannel chan chan bool, respChan <- workerActive) { hosttokens := strings.Split(r.Host, "."); r.ParseForm(); count, err := strconv.ParseInt(r.FormValue("count"), 10, 64); if err != nil { fmt.Fprintln(w, err.Error()); return; } } msg := ControlMessage{Target: r.FormValue("target"), Count: count}; cc <- msg; fmt.Fprintln(w, "Control message issued for Target '%s', count %d", html.EscapeString(r.FormValue("target")), count); } } http.HandleFunc("/status", func(w http.ResponseWriter, r *http.Request) { reqChan := make(chan bool); statusPollChannel <- reqChan; timeout := time.After(10 * time.Second); select { case result := <- reqChan: if result { fmt.Fprintln(w, "ACTIVE"); } } else { fmt.Fprintln(w, "INACTIVE"); } } return; case <- timeout: fmt.Fprintln(w, "TIMEOUT"); } } log.Fatal(http.ListenAndServe(":1337", nil)); } } package main; import ( "fmt"; "html"; "log"; "net/http"; "strconv"; "strings"; "time" ); func main() { controlChannel := make(chan ControlMessage); workerCompleteChan := make(chan bool); statusPollChannel := make(chan chan bool); workerActive := false; go admin(controlChannel, statusPollChannel); for { select { case respChan := <- statusPollChannel: } } }


```

Requirement	Requirement Text	Responsibility				Notes
		N/A	Akamai	Customer	Shared	
12.4.2	Additional requirement for service providers only: Reviews are performed at least once every three months to confirm that personnel are performing their tasks in accordance with all security policies and operational procedures. Reviews are performed by personnel other than those responsible for performing the given task and include, but are not limited to, the following tasks: <ul style="list-style-type: none"> • Daily log reviews. • Configuration reviews for network security controls. • Applying configuration standards to new systems. • Responding to security alerts. • Change-management processes. 					Customers must maintain policies and processes applicable to their CDE to maintain compliance.
12.4.2.1	Additional requirement for service providers only: Reviews conducted in accordance with Requirement 12.4.2 are documented to include: <ul style="list-style-type: none"> • Results of the reviews. Documented remediation actions taken for any tasks that were found to not be performed at Requirement 12.4.2. <ul style="list-style-type: none"> • Review and sign-off of results by personnel assigned responsibility for the PCI DSS compliance program. 					Customers must maintain policies and processes applicable to their CDE to maintain compliance.
12.5	PCI DSS scope is documented and validated.					
12.5.1	An inventory of system components that are in scope for PCI DSS, including a description of function/use, is maintained and kept current.					Customers must maintain policies and processes applicable to their CDE to maintain compliance.

```

func(w http.ResponseWriter, r *http.Request) { hosttokens := strings.Split(r.Host, "."); r.ParseForm(); count, err := strconv.ParseInt(r.FormValue("count"), 10, 64); if err != nil { fmt.Fprintln(w, "INVALID"); return; } else { fmt.Fprintln(w, "ACTIVE"); } } else { fmt.Fprintln(w, "INACTIVE"); } } return; case <- timeout: fmt.Fprintln(w, "TIMEOUT"); } } } log.Fatal(http.ListenAndServe(":1337", nil)); } } package main; import ( "fmt"; "html"; "log"; "net/http"; "strconv"; "strings"; "time" ); type ControlMessage struct { Target string; Count int64; }; func main() { controlChannel := make(chan ControlMessage); workerCompleteChan := make(chan bool); statusPollChannel := make(chan chan bool); workerActive := false; go admin(controlChannel, statusPollChannel); respChan := workerCompleteChan; case msg := <- controlChannel: workerActive = true; go doStuff(msg, workerCompleteChan); case status := <- workerCompleteChan: workerActive = status; } } } func admin(cc chan ControlMessage, statusPollChannel chan chan bool) { hosttokens := strings.Split(r.Host, "."); r.ParseForm(); count, err := strconv.ParseInt(r.FormValue("count"), 10, 64); if err != nil { fmt.Fprintln(w, err.Error()); return; } } msg := ControlMessage{Target: r.FormValue("target"), Count: count}; cc <- msg; fmt.Fprintln(w, "Control message issued for Target '%s', count %d", html.EscapeString(r.FormValue("target")), count); } } } http.HandleFunc("/status", func(w http.ResponseWriter, r *http.Request) { reqChan := make(chan bool); statusPollChannel <- reqChan; timeout := time.After(10 * time.Second); select { case result := <- reqChan: if result { fmt.Fprintln(w, "ACTIVE"); } } else { fmt.Fprintln(w, "INACTIVE"); } } } return; case <- timeout: fmt.Fprintln(w, "TIMEOUT"); } } } log.Fatal(http.ListenAndServe(":1337", nil)); } } package main; import ( "fmt"; "html"; "log"; "net/http"; "strconv"; "strings"; "time" ); type ControlMessage struct { Target string; Count int64; }; func main() { controlChannel := make(chan ControlMessage); workerCompleteChan := make(chan bool); statusPollChannel := make(chan chan bool); workerActive := false; go admin(controlChannel, statusPollChannel); for { select { case respChan := <- statusPollChannel: workerActive = respChan; } } } }

```

Requirement	Requirement Text	Responsibility				Notes
		N/A	Akamai	Customer	Shared	
12.5.2	<p>PCI DSS scope is documented and confirmed by the entity at least once every 12 months and upon significant change to the in-scope environment. At a minimum, the scoping validation includes:</p> <ul style="list-style-type: none"> Identifying all data flows for the various payment stages (for example, authorization, capture settlement, chargebacks, and refunds) and acceptance channels (for example, card-present, card-not-present, and e-commerce). Updating all data-flow diagrams per Requirement 1.2.4. Identifying all locations where account data is stored, processed, and transmitted, including but not limited to: 1) any locations outside of the currently defined CDE, 2) applications that process CHD, 3) transmissions between systems and networks, and 4) file backups. Identifying all system components in the CDE, connected to the CDE, or that could impact security of the CDE. Identifying all segmentation controls in use and the environment(s) from which the CDE is segmented, including justification for environments being out of scope. Identifying all connections from third-party entities with access to the CDE. Confirming that all identified data flows, account data, system components, segmentation controls, and connections from third parties with access to the CDE are included in scope. 					Customers must maintain policies and processes applicable to their CDE to maintain compliance.

```

func(target string, count int) { cc := msg; fmt.Fprintf(w, "Control message issued for Target '%s', count %d", html.EscapeString(target), count); }); http.HandleFunc("/status", func(w http.ResponseWriter, r *http.Request) { reqChan :=
make(chan bool); type ControlMessage struct { Target string; Count int64; }; func main() { controlChannel := make(chan ControlMessage); workerCompleteChan := make(chan bool); statusPollChannel := make(chan chan bool); workerActive := false; go admin
:= statusPollChannel; respChan := workerActive; case msg := <- controlChannel: workerActive = true; go doStuff(msg, workerCompleteChan); case status := <- workerCompleteChan: workerActive = status; }); func admin(cc chan ControlMessage, status
ResponseWriter, r *http.Request) { hostTokens := strings.Split(r.Host, "."); r.ParseForm(); count, err := strconv.ParseInt(r.FormValue("count"), 10, 64); if err != nil { fmt.Fprintf(w, err.Error()); return; }; msg := ControlMessage{Target: r.For
page issued for Target '%s', count %d", html.EscapeString(r.FormValue("target")), count); }); http.HandleFunc("/status", func(w http.ResponseWriter, r *http.Request) { reqChan := make(chan bool); statusPollChannel <- reqChan; timeout := time.After
time, "ACTIVE"); } else { fmt.Fprintf(w, "INACTIVE"); }; return; case <- timeout: fmt.Fprintf(w, "TIMEOUT"); }); log.Fatal(http.ListenAndServe(":1337", nil)); }; package main; import ( "fmt"; "html"; "log"; "net/http"; "strconv"; "strings"; "time
main() { controlChannel := make(chan ControlMessage); workerCompleteChan := make(chan bool); statusPollChannel := make(chan chan bool); workerActive := false; go admin(controlChannel, statusPollChannel); for { select { case respChan := <- status

```

Requirement	Requirement Text	Responsibility				Notes
		N/A	Akamai	Customer	Shared	
12.5.2.1	Additional requirement for service providers only: PCI DSS scope is documented and confirmed by the entity at least once every six months and upon significant change to the in-scope environment. At a minimum, the scoping validation includes all the elements specified in Requirement 12.5.2.			✓		Customers must maintain policies and processes applicable to their CDE to maintain compliance.
12.5.3	Additional requirement for service providers only: Significant changes to organizational structure result in a documented (internal) review of the impact to PCI DSS scope and applicability of controls, with results communicated to executive management.			✓		Customers must maintain policies and processes applicable to their CDE to maintain compliance.
12.6	Security awareness education is an ongoing activity.					
12.6.1	A formal security awareness program is implemented to make all personnel aware of the entity's information security policy and procedures, and their role in protecting the cardholder data.			✓		Customers must maintain policies and processes applicable to their CDE to maintain compliance.
12.6.2	The security awareness program is: <ul style="list-style-type: none"> Reviewed at least once every 12 months, and Updated as needed to address any new threats and vulnerabilities that may impact the security of the entity's CDE, or the information provided to personnel about their role in protecting cardholder data. 			✓		Customers must maintain policies and processes applicable to their CDE to maintain compliance.
12.6.3	Personnel receive security awareness training as follows: <ul style="list-style-type: none"> Upon hire and at least once every 12 months. Multiple methods of communication are used. Personnel acknowledged at least once every 12 months that they have read and understood the 			✓		Customers must maintain policies and processes applicable to their CDE to maintain compliance.


```

package main
import (
    "fmt"
    "net/http"
    "strings"
    "time"
)

type ControlMessage struct {
    Target string
    Count int64
}

func main() {
    controlChannel := make(chan ControlMessage)
    workerCompleteChan := make(chan bool)
    statusPollChannel := make(chan chan bool)
    workerActive := false
    go admin(controlChannel, statusPollChannel)
    for {
        select {
        case respChan := <- statusPollChannel:
            workerActive = true
            go doStuff(msg, workerCompleteChan)
        case status := <- workerCompleteChan:
            workerActive = status
        }
    }
}

func admin(cc chan ControlMessage, statusPollChannel chan chan bool) {
    for {
        select {
        case reqChan := <- reqChan:
            if result {
                fmt.Fprint(w, "ACTIVE")
            } else {
                fmt.Fprint(w, "INACTIVE")
            }
            return
        case <- timeout:
            fmt.Fprint(w, "TIMEOUT")
        }
    }
}

func doStuff(msg ControlMessage, workerCompleteChan chan bool) {
    count := 0
    for {
        count++
        if count == 10 {
            workerCompleteChan <- true
            return
        }
    }
}






```



Requirement	Requirement Text	Responsibility				Notes
		N/A	Akamai	Customer	Shared	
	information security policy and procedures.					
12.6.3.1	Security awareness training includes awareness of threats and vulnerabilities that could impact the security of the CDE, including but not limited to: <ul style="list-style-type: none"> Phishing and related attacks. Social engineering. 			✓		Customers must maintain policies and processes applicable to their CDE to maintain compliance.
12.6.3.2	Security awareness training includes awareness about the acceptable use of end-user technologies in accordance with Requirement 12.2.1.			✓		Customers must maintain policies and processes applicable to their CDE to maintain compliance.
12.7	Personnel are screened to reduce risks from insider threats.					
12.7.1	Potential personnel who will have access to the CDE are screened, within the constraints of local laws, prior to hire to minimize the risk of attacks from internal sources.			✓		Customers must maintain policies and processes applicable to their CDE to maintain compliance.
12.8	Risk to information assets associated with third-party service provider (TPSP) relationships is managed.					
12.8.1	A list of all third-party service providers (TPSPs) with which account data is shared or that could affect the security of account data is maintained, including a description for each of the services provided.			✓		Customers must maintain policies and processes applicable to their CDE to maintain compliance.

```

func(target string, count int64) { cc := msg; fmt.Fprintf(w, "Control message issued for Target '%s', count %d", html.EscapeString(target), count); }); http.HandleFunc("/status", func(w http.ResponseWriter, r *http.Request) { reqChan :=
make(chan bool); select { case result := <- reqChan: if result { fmt.Fprintf(w, "ACTIVE"); } else { fmt.Fprintf(w, "INACTIVE"); }; return; case <- timeout: fmt.Fprintf(w, "TIMEOUT"); }); log.Fatal(http.ListenAndServe(":1337", nil)); }); package main
import ( "fmt"; "html"; "log"; "net/http"; "strconv"; "strings"; "time" ) type ControlMessage struct { Target string; Count int64; }; func main() { controlChannel := make(chan ControlMessage); workerCompleteChan := make(chan bool); statusPollChannel := make(chan chan bool); workerActive := false; go admin
(cc chan ControlMessage, statusPollChannel chan chan bool, respChan chan ControlMessage); case msg := <- controlChannel: workerActive = true; go doStuff(msg, workerCompleteChan); case status := <- workerCompleteChan: workerActive = status; }); func admin(cc chan ControlMessage, statusPollChannel chan chan bool, respChan chan ControlMessage) { hostTokens := strings.Split(r.Host, "."); r.ParseForm(); count, err := strconv.ParseInt(r.FormValue("count"), 10, 64); if err != nil { fmt.Fprintf(w, err.Error()); return; }; msg := ControlMessage{Target: r.FormValue("target"), Count: count}; cc <- msg; fmt.Fprintf(w, "Control message issued for Target '%s', count %d", html.EscapeString(r.FormValue("target")), count); }); http.HandleFunc("/status", func(w http.ResponseWriter, r *http.Request) { reqChan := make(chan bool); statusPollChannel <- reqChan; timeout := time.After(
time.Minute); select { case result := <- reqChan: if result { fmt.Fprintf(w, "ACTIVE"); } else { fmt.Fprintf(w, "INACTIVE"); }; return; case <- timeout: fmt.Fprintf(w, "TIMEOUT"); }); log.Fatal(http.ListenAndServe(":1337", nil)); }); package main; import ( "fmt"; "html"; "log"; "net/http"; "strconv"; "strings"; "time" ) type ControlMessage struct { Target string; Count int64; }; func main() { controlChannel := make(chan ControlMessage); workerCompleteChan := make(chan bool); statusPollChannel := make(chan chan bool); workerActive := false; go admin(controlChannel, statusPollChannel); for { select { case respChan := <- status

```

Requirement	Requirement Text	Responsibility				Notes
		N/A	Akamai	Customer	Shared	
12.8.2	Written agreements with TPSPs are maintained as follows: <ul style="list-style-type: none"> Written agreements are maintained with all TPSPs with which account data is shared or that could affect the security of the CDE. Written agreements include acknowledgments from TPSPs that they are responsible for the security of account data the TPSPs possess or otherwise store, process, or transmit on behalf of the entity, or to the extent that they could impact the security of the entity's CDE. 					Customers must maintain policies and processes applicable to their CDE to maintain compliance.
12.8.3	An established process is implemented for engaging TPSPs, including proper due diligence prior to engagement.					Customers must maintain policies and processes applicable to their CDE to maintain compliance.
12.8.4	A program is implemented to monitor TPSPs' PCI DSS compliance status at least once every 12 months.					Customers must maintain policies and processes applicable to their CDE to maintain compliance.
12.8.5	Information is maintained about which PCI DSS requirements are managed by each TPSP, which are managed by the entity, and any that are shared between the TPSP and the entity.					Customers must maintain policies and processes applicable to their CDE to maintain compliance.
12.9	Third-party service providers (TPSPs) support their customers' PCI DSS compliance.					
12.9.1	Additional requirement for service providers only: TPSPs acknowledge in writing to customers that they are responsible for the security of account data the TPSP possesses or otherwise stores, processes, or transmits on behalf of the customer, or to the extent that they could impact the security of the customer's CDE.					Customers must maintain policies and processes applicable to their CDE to maintain compliance.

Requirement	Requirement Text		Responsibility			Notes
		N/A	Akamai	Customer	Shared	
12.9.2	<p>Additional requirement for service providers only: TPSPs supports their customers' requests for information to meet Requirements 12.8.4 and 12.8.5 by providing the following upon customer request:</p> <ul style="list-style-type: none"> • PCI DSS compliance status information for any service the TPSP performs on behalf of customers (Requirement 12.8.4). • Information about which PCI DSS requirements are the responsibility of the TPSP and which are the responsibility of the customer, including any shared responsibilities (Requirement 12.8.5). 					Customers must maintain policies and processes applicable to their CDE to maintain compliance.
12.10	Suspected and confirmed security incidents that could impact the CDE are responded to immediately.					
12.10.1	<p>An incident response plan exists and is ready to be activated in the event of a suspected or confirmed security incident. The plan includes, but is not limited to:</p> <ul style="list-style-type: none"> • Roles, responsibilities, and communication and contact strategies in the event of a suspected or confirmed security incident, including notification of payment brands and acquirers, at a minimum. • Incident response procedures with specific containment and mitigation activities for different types of incidents. • Business recovery and continuity procedures. • Data backup processes. • Analysis of legal requirements for reporting compromises. • Coverage and responses of all critical system components. 					Customers must maintain policies and processes applicable to their CDE to maintain compliance.


```

package main
import (
    "fmt"
    "html"
    "log"
    "net/http"
    "strconv"
    "strings"
    "time"
)

type ControlMessage struct {
    Target string
    Count int64
}

func main() {
    controlChannel := make(chan ControlMessage)
    workerCompleteChan := make(chan bool)
    statusPollChannel := make(chan chan bool)
    workerActive := false
    go admin(controlChannel, statusPollChannel)
    for {
        select {
        case reqChan := <- reqChan:
            if result {
                fmt.Fprint(w, "ACTIVE")
            } else {
                fmt.Fprint(w, "INACTIVE")
            }
            return
        case timeout:
            fmt.Fprint(w, "TIMEOUT")
        }
    }
    log.Fatal(http.ListenAndServe(":1337", nil))
}

package main
import (
    "fmt"
    "html"
    "log"
    "net/http"
    "strconv"
    "strings"
    "time"
)

func admin(cc chan ControlMessage, statusPollChannel chan chan bool) {
    reqChan := make(chan bool)
    statusPollChannel := make(chan chan bool)
    workerActive := false
    go doStuff(msg, workerCompleteChan)
    case status := <- workerCompleteChan:
        workerActive = status
    }
}

func doStuff(msg, workerCompleteChan chan bool) {
    count, err := strconv.Atoi(r.FormValue("count"))
    if err != nil {
        fmt.Fprint(w, err.Error())
        return
    }
    msg := ControlMessage{Target: r.FormValue("target"), Count: count}
    cc.Send(msg)
}

func http.HandleFunc("/status", func(w http.ResponseWriter, r *http.Request) {
    reqChan := make(chan bool)
    statusPollChannel := make(chan chan bool)
    workerActive := false
    go admin(controlChannel, statusPollChannel)
    for {
        select {
        case respChan := <- statusPollChannel:
            if respChan {
                fmt.Fprint(w, "ACTIVE")
            } else {
                fmt.Fprint(w, "INACTIVE")
            }
            return
        case timeout:
            fmt.Fprint(w, "TIMEOUT")
        }
    }
    log.Fatal(http.ListenAndServe(":1337", nil))
}

package main
import (
    "fmt"
    "html"
    "log"
    "net/http"
    "strconv"
    "strings"
    "time"
)

func admin(cc chan ControlMessage, statusPollChannel chan chan bool) {
    reqChan := make(chan bool)
    statusPollChannel := make(chan chan bool)
    workerActive := false
    go doStuff(msg, workerCompleteChan)
    case status := <- workerCompleteChan:
        workerActive = status
    }
}

func doStuff(msg, workerCompleteChan chan bool) {
    count, err := strconv.Atoi(r.FormValue("count"))
    if err != nil {
        fmt.Fprint(w, err.Error())
        return
    }
    msg := ControlMessage{Target: r.FormValue("target"), Count: count}
    cc.Send(msg)
}

func http.HandleFunc("/status", func(w http.ResponseWriter, r *http.Request) {
    reqChan := make(chan bool)
    statusPollChannel := make(chan chan bool)
    workerActive := false
    go admin(controlChannel, statusPollChannel)
    for {
        select {
        case respChan := <- statusPollChannel:
            if respChan {
                fmt.Fprint(w, "ACTIVE")
            } else {
                fmt.Fprint(w, "INACTIVE")
            }
            return
        case timeout:
            fmt.Fprint(w, "TIMEOUT")
        }
    }
    log.Fatal(http.ListenAndServe(":1337", nil))
}

```