appviewx

Solution Brief

AppViewX SIGN+

Make Code Signing Secure, Seamless, and Controlled for DevOps and Security Teams with AppViewX SIGN+

www.appviewx.com

We live in a world driven by digital technologies and interconnectivity, where software applications have penetrated every aspect of personal and professional existence. Millions of people worldwide download, use, and update software applications on their phones, tablets, and laptops on a daily basis. Businesses also heavily rely on various third-party software to facilitate everyday operations and transactions.

As the reliance on software continues to grow, the question of digital trust looms larger than ever before. Software users want assurance that the software they are downloading and installing is legitimate and trust-worthy. Software providers need to assure end users of the authenticity and integrity of the software they have developed. Without this line of trust, there would be no secure way of selling and consuming software on the Internet.

Trust is fundamental to any digital transaction, and with cyberattacks and data breaches increasing, trust is a strategic necessity. It is what builds user confidence, promotes brand reputation, and drives economic growth, which is why organizations must prioritize security practices and solutions that can establish and preserve digital trust. This is where code signing plays a critical role.

Code Signing Is Indispensable to Safeguarding the Software Supply Chain

Code Signing is a process of digitally signing software, firmware, or mobile applications to assure the end users of their authenticity and integrity. Developers digitally sign the code with a unique private key and a code signing certificate that serve as proof that the code originates from a verified source and has not been tampered with since it was signed.

Digitally signing code helps end users trust the software they are installing or running by verifying the digital signature. If the software or code is tampered with after it has been digitally signed, the digital signature will be invalidated and users will be warned not to trust the code and software.

The primary purpose of code signing is to establish trust between software providers and end users by providing proof of software authenticity, integrity, and non-repudiation. In addition to promoting user trust, code signing helps software providers protect intellectual property, ensure compliance, strengthen software supply chain security, and uphold their brand reputation.

Over the past several years, with the migration to the Cloud and adoption of DevOps, the need for code signing has rapidly grown, expanding to a wide range of enterprise use cases.

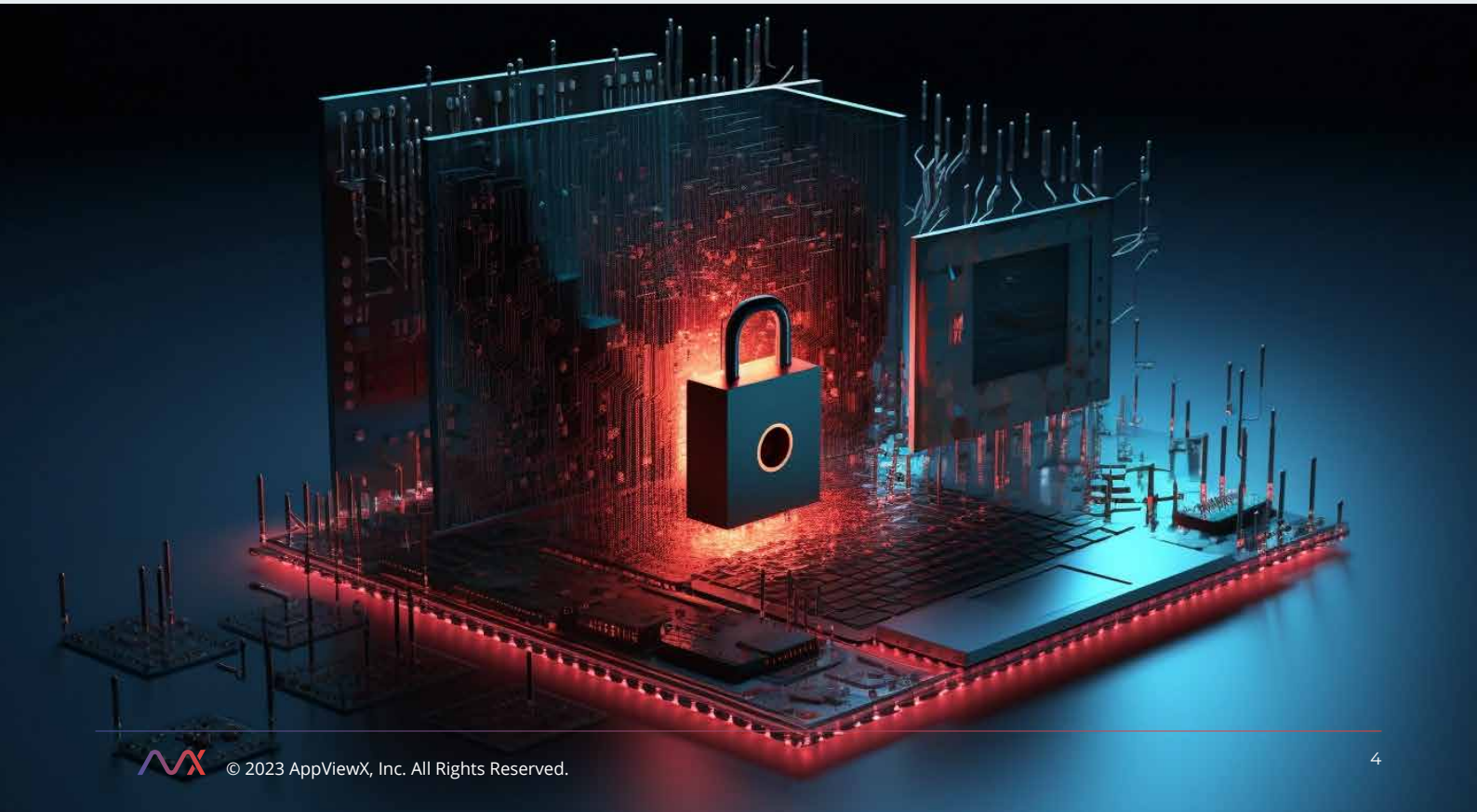
- **DevOps:** To ensure that the software artifacts, such as compiled code, containers, or PowerShell scripts, are signed before being deployed to production environments.
- **SecOps:** To ensure that all software running within the organization is properly signed and comes from trusted sources.
- **Software Distribution:** To sign desktop applications, mobile apps, or software updates to allow users to verify the authenticity and integrity of software.
- **Software Development Kits (SDKs) and Frameworks:** To sign SDKs and frameworks that are distributed to developers. By signing these components, developers can verify their authenticity and ensure they have not been tampered with.
- **Infrastructure Provisioning:** To validate and verify the integrity of configuration scripts, automation code, or infrastructure-as-code templates.
- **Virtualization and Cloud Environments:** To validate the integrity of virtual machine images, containers, and cloud-based software components.
- **Software Integration and APIs:** To sign software integration modules or API libraries and ensure that the code used to connect different systems or services is trustworthy and has not been tampered with.
- **IoT Security:** To sign firmware and ensure that IoT devices remain secure throughout their lifecycle.

Unreliable Code Signing Practices Amplify Security Risks

Although code signing is a powerful practice, it is only effective when securely implemented. Even today, many organizations struggle with code signing due to ad-hoc and fragmented processes that leave their organizations and software vulnerable to attacks.

Inadequate and insecure code signing practices have severe and far-reaching consequences. Code signing compromises can lead to malware injection, software and code tampering, and impersonation attacks that can, in turn, result in ransomware attacks, data breaches, loss of customer trust, financial damages, and compliance issues.

The SolarWinds supply chain attack clearly demonstrated the magnitude of impact code attacks can cause. A malware-infected software update masquerading as legitimate was enough to break into more than thirty-thousand highly secure organizations, causing the biggest supply chain attack known to date.



Common Obstacles That Stand in the Way of Secure and Efficient Code Signing

- **Private Key Mismanagement**

Private keys are the heart of the code signing process and must always be protected. If the private keys linked to the code signing certificates are stolen, attackers could use the compromised code signing certificate to sign malware and then distribute the software under a verified publisher name. Despite the awareness, developers have continued storing code signing keys on their local machines, or build servers, exposing organizations to private key theft and misuse.

Given how prevalent private key mismanagement is and the increase in the number of code signing attacks, the CA/Browser Forum recently passed a mandate that requires all publicly trusted code signing private keys to be generated and stored in secure hardware crypto modules such as hardware security modules (HSMs) that are at least FIPS 140-2 Level 2 or Common Criteria EAL 4+ and not on local machines.

- **Not Up to DevOps Speed**

While the new mandate around private key protection is a great step towards making code signing more secure, it can still make things difficult for DevOps in terms of speed and agility. Currently, there are two approaches organizations can opt for private key protection - hardware tokens or HSMs.

Hardware Token Challenges: By default, most Certificate Authorities (CAs) will offer hardware tokens for storing private keys, however this can make the code signing process challenging for several reasons. Multiple developers across regions usually need access to the code signing certificate for signing.

Sharing tokens between multiple developers is not practical or secure, given the risk of password leaks or compromises. At the same time, issuing multiple token-based code signing certificates to dispersed developers makes it difficult for security teams to monitor remote code signing events.

HSM Complexities: While storing private keys in crypto hardware like HSMs provides the highest level of security, connecting systems and DevOps tools to HSMs for signing purposes can be highly complex. As distributed development teams often use multiple tools for code signing, the lack of HSM integration with signing tools can complicate code signing access and slow down the signing process. Sourcing your own HSM adds to the cost and complexity, given it needs to be compliant and on-premises HSMs require maintenance.

Additionally, developers often work with different signing tools and platforms with varying code signing requirements. Developers also need to sign various file types and processes to ensure compatibility across different distribution channels and devices. Given these requirements, code signing must be easy for developers to use and ideally automated, which means it needs to be integrated with DevOps tools and processes including within the CI/CD pipeline. When it's not, developers are forced to invest more time and effort in signing code outside of their native tool sets, disrupting their workflows and cutting into their productivity.

- **Lack of Visibility or Control Over Code Signing Events**

  Modern enterprises have their development teams working in several locations across the world, including remotely. Different teams use different tools for signing, and in the past have left private keys exposed on developer workstations or build servers. Developers using different signing tools and managing private keys on their own often leads to inconsistencies and security risks in code signing. On the other hand, security teams have no visibility into code signing events. They cannot track or control who is accessing the private key, where they are stored, and what code was signed - which creates security blind spots and auditing and compliance issues.

Despite these challenges, code signing remains a critical security practice, and many of the challenges can be overcome with an efficient, integrated, and automated code signing solution.

How AppViewX SIGN+ Helps Simplify Code Signing for DevOps and Ensure Security

AppViewX SIGN+ is a fast, reliable, and secure code signing solution built to protect the integrity of code, containers, firmware, and software. With a centralized and integrated approach, AppViewX SIGN+ simplifies code signing for DevOps and streamlines security.

AppViewX SIGN+ seamlessly integrates with native signing tools, CI/CD pipeline and workflows to empower DevOps teams to securely sign code faster and more freely. Security teams can rest assured with full visibility and policy-driven control over private key storage, code signing certificate management, and access and usage.

1. Secure Code Signing

Robust HSM Protection for Private Keys

Strong Private Key Protection

With AppViewX SIGN+, private keys are generated and stored securely in FIPS 140-2 certified HSMs as mandated by the CA/Browser Forum. AppViewX SIGN+ integrates seamlessly and directly with both cloud-based and on-premises HSMs behind the scenes for secure signing. Customers have the flexibility to use a compliant HSM of their choice (on-premises or cloud based) or opt for code signing as a service. Instead of ad hoc code signing practices with developers individually managing private keys, security teams can ensure that signing keys are centrally and securely stored on compliant HSMs.

Extensive Algorithm Support

Code signing relies on various cryptographic algorithms to generate digital signatures, create hash values, and ensure the security of the signing process. AppViewX SIGN+ supports all standard asymmetric cryptographic algorithms used today, such as RSA, ECDSA, and DSA. AppViewX SIGN+ will also begin supporting Post Quantum Cryptography (PQC) algorithms once they are standardized and enabled by HSMs.

Secure Timestamping

As code signing certificates have limited validity periods (usually from one to three years), timestamping is a critical best practice to help consumers verify the legitimacy of the code signing signature even after the certificate used for signing expires or is revoked. AppViewX SIGN+ integrates with timestamping authorities to enable seamless timestamping during the code signing process and ensure the long term validity of digital signatures.

2. Seamless Code Signing

Streamlined and Effortless Signing Processes for Development Teams

AppViewX SIGN+ offers a flexible and integrated approach to code signing, making it simple and streamlined for distributed development teams. Given developers often need to sign a variety of file formats, AppViewX SIGN+ supports a wide range of file types, including Windows Applications, libraries, OCX files, Kernel drivers, Apple software, Microsoft Office VBA objects, JAR files, .air or .airi files, containers, and Android APK files.

Furthermore, AppViewX SIGN+ offers flexible and integrated methods for signing including directly from:

Native Signing Tools

AppViewX SIGN+ seamlessly integrates with popular signing tools (such as Microsoft Signtool or JarSigner) used in software development. The integration ensures seamless compatibility and allows developers to carry out signing from within their signing tools, using custom AppViewX CSP (Cryptographic Service Provider) and PKCS11 providers (that enable the signing tools to talk to and access the code signing certificate that is securely stored on the HSM). As a centralized and secure code signing service, AppViewX SIGN+ makes it easy and frictionless for developers to sign code from anywhere and from any signing tool without worrying about key management and storage.

CI/CD Pipeline

The custom CSP/PKCS11 interfaces provided by AppViewX SIGN+ also integrate seamlessly with the CI/CD pipeline— with various DevOps toolchains, CI/CD tools, and workflows, such as GitHub, GitLab, Jenkins, Bitbucket, and Azure DevOps— to automatically trigger code signing during the build process.

API Workflows

AppViewX SIGN+ provides flexible API-based signing that allows developers to interact with AppViewX to programmatically incorporate code signing capabilities into their existing workflows, applications, or automated build/deployment processes.

AppViewX SIGN+ Console

AppViewX SIGN+ also allows developers to sign their code directly through the AppViewX SIGN+ console. All it takes is a signing policy specifying the hash algorithm, timestamping authority, key(s) for signing, and metadata to be collected during the signing. Once the signing policy is successfully created, developers can upload and sign the code with a single click.

3. Controlled Code Signing

Centralized Visibility Of Code Signing Events, Policy Management, and Overall Control Over Key and Certificate Access

Visibility and Control

With AppViewX SIGN+ offering centralized code signing services, security teams can gain complete visibility into and control over code signing keys and certificates and signing events across the enterprise. Developers can efficiently sign code from anywhere, while security teams maintain overall control.

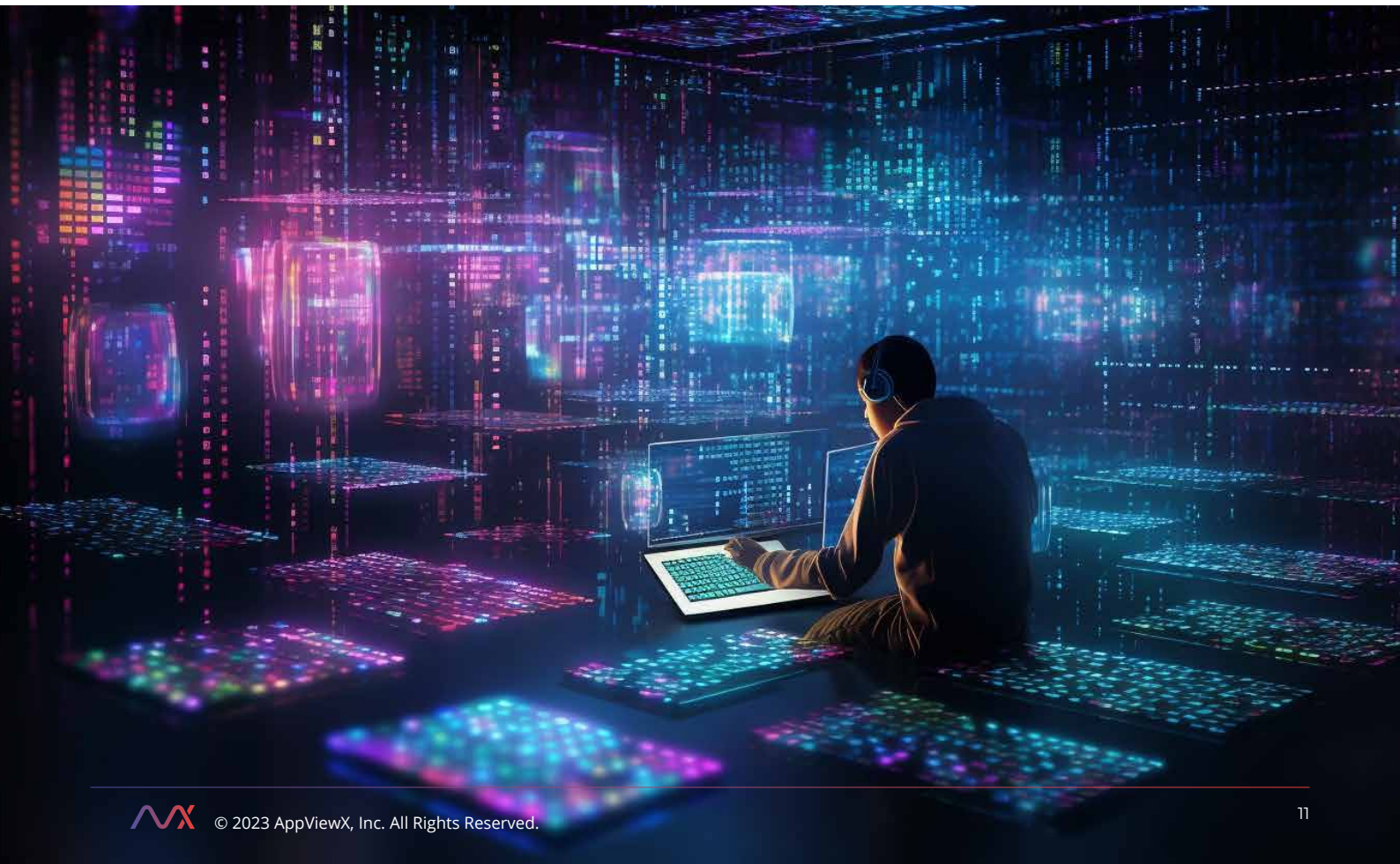
The intuitive AppViewX SIGN+ dashboard provides essential information, such as different files being signed, time of signing, timestamping usage, and the certificate and key used for signing. Security teams can also generate detailed audit logs and reports that historically track code signing events to simplify audits and ensure continuous compliance.

Policy Management

AppViewX SIGN+ helps security teams standardize the code signing process by defining and mapping signing policies, including key usage permissions, approvals, key expiry, CA type, key size, signing algorithm type, and more. It can also automate policy enforcement to ensure all files are signed based on the configured signing policy, and are compliant with the industry standards.

User Authentication and Access control

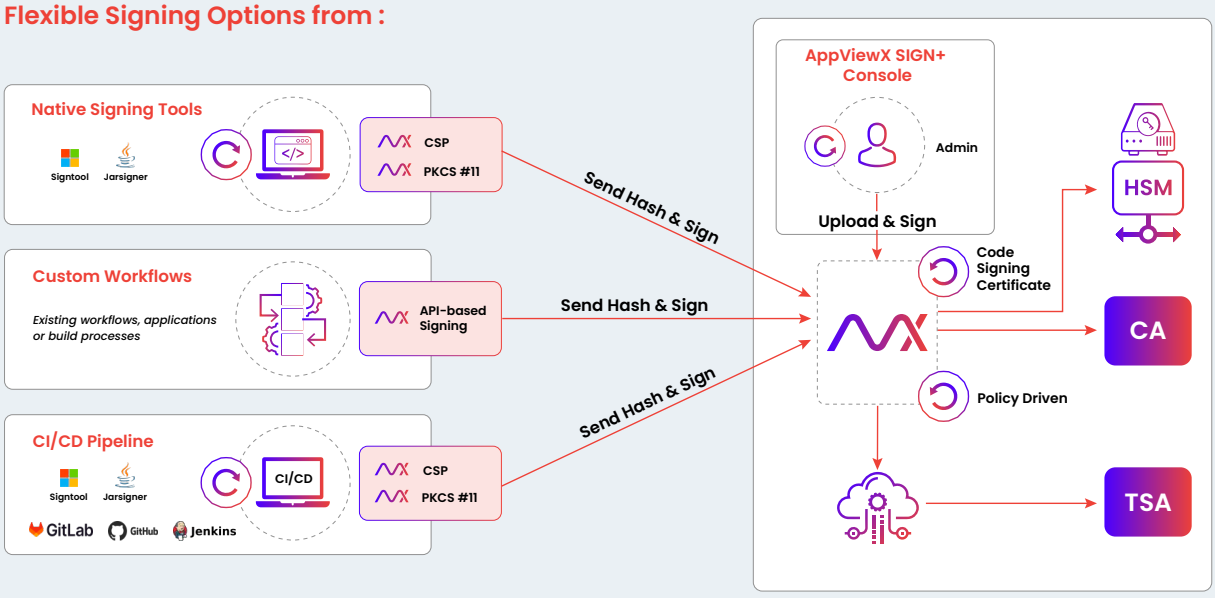
To prevent unauthorized signing, AppViewX SIGN+ enables user authentication and authorization through Role-based Access Control (RBAC), which defines who can access the certificates, what they can sign, and what signing tools they can use. Implementing RBAC ensures that only users with specific user roles assigned to them have the access to modules within AppViewX SIGN+ for signing and providing controlled access to signing keys and certificates.



Why AppViewX SIGN+

- **Simple and Seamless Code Signing** - One centralized service to support distributed development teams, process, and all enterprise code signing needs.
- **Integrated Code Signing** - Easily integrate code signing with native signing tools and DevOps processes (CI/CD pipeline) to make signing simple, fast, and secure.
- **Strong Private Key Protection** - Generate and store private keys in CA/B Forum mandated HSMs, without having to install and maintain on-premises hardware.
- **Security and Compliance** - Give security teams centralized visibility and control over policy, access and usage of code signing keys and certificates.
- **Speed and Scalability** - Enable reliable, high volume code signing at the speed of DevOps without impacting developer productivity or compromising on security.

AppViewX SIGN+ Architecture



AppViewX SIGN+ Flexible Deployment Models

AppViewX SIGN+ as part of the AppViewX Digital Trust Platform has flexible deployment options, including:

SaaS Operated by AppViewX

Available as a service, AppViewX SIGN+ is fully managed and updated by AppViewX. Customers can directly set up an account and start using it. For connecting to the non-public corporate network segments without poking a hole into the corporate firewall, AppViewX provides a Cloud Connector that needs to be installed in the private network.

On-Prem and Hosted Deployment

AppViewX SIGN+ can also be deployed within a customer-managed environment, including hypervisor-based VMs, private clouds, or public clouds using AWS, GCP, Microsoft Azure, and others. AppViewX SIGN+ can be installed on any virtual machine instance running CentOS or RHEL operating system. As AppViewX SIGN+ is a Kubernetes-based application, it can also be installed in a managed Kubernetes environment like EKS, AKS, GKE, RedHat Openshift, Rancher, and others.

Security simplified with AppViewX

AppViewX is trusted by the world's leading global organizations to ensure application availability, security and compliance with centralized visibility and control of public key infrastructure (PKI) and application delivery services across complex hybrid multi-cloud environments. The AppViewX Platform enables self-service automation and orchestration for NetOps, DevOps, SecOps and application teams to quickly and easily translate business requirements into automation workflows that improve agility, harden security, enforce compliance, eliminate errors, and reduce cost.

Make visibility the cornerstone of your protection mechanism.

https://www.appviewx.com/live-demo/

AppViewX Inc.,

City Hall, 222 Broadway
New York, NY 10038

info@appviewx.com
www.appviewx.com

+1 (206) 207-7541
+44 (0) 203-514-2226