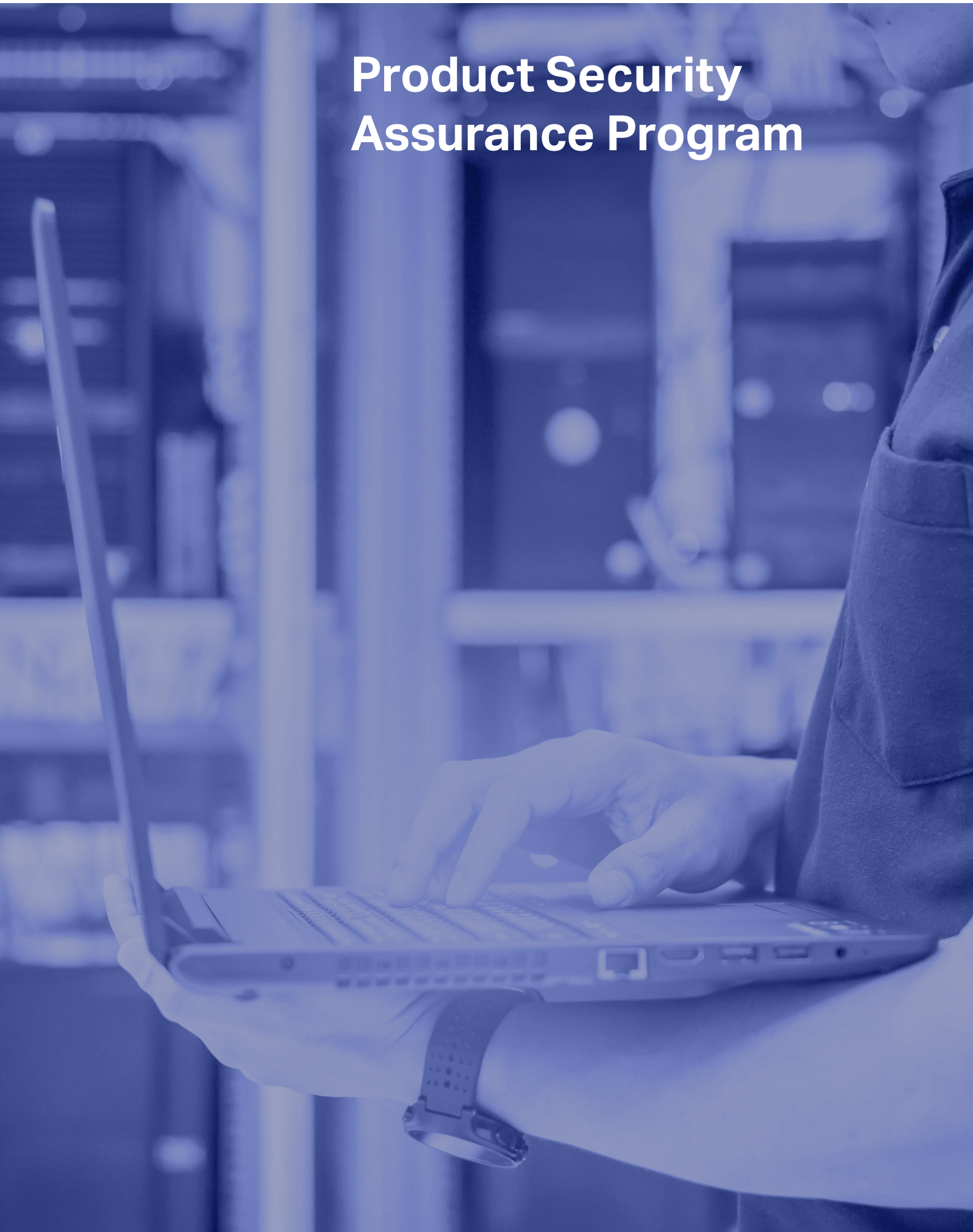
opentext™

The Information Company™

Product Security Assurance Program



opentext™

Contents

Objective

The goals of the OpenText Product Security Assurance Program (PSAP) are to help ensure that all products, solutions, and services are designed, developed, and maintained with security in mind, and to provide OpenText customers with the assurance that their important assets and information are protected at all times. This document provides a general, public overview of the key aspects and components of the PSAP program.

Scope

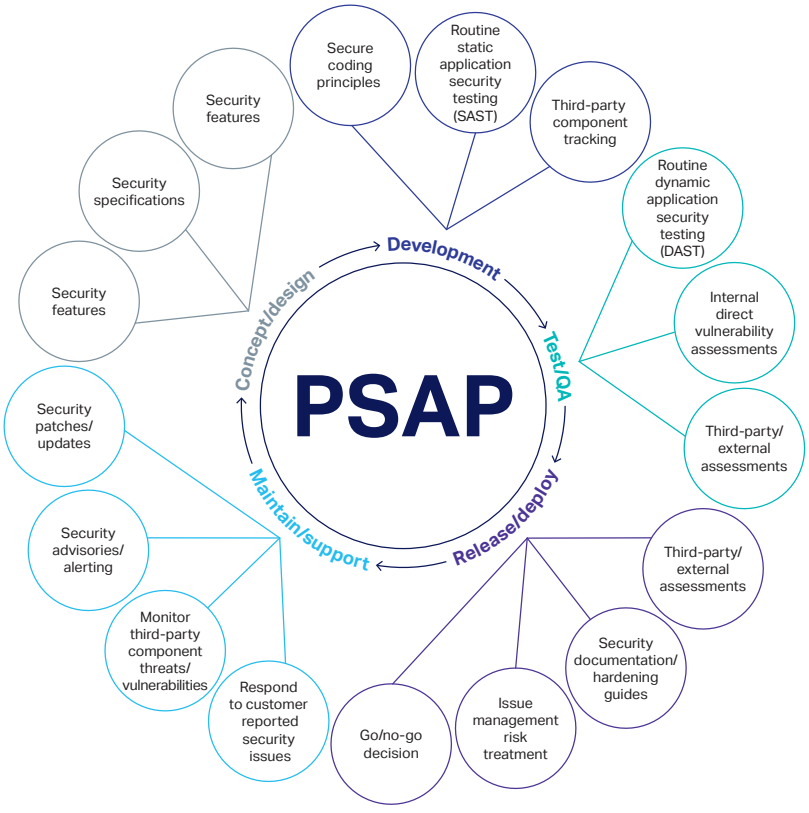
The scope of the PSAP includes all software solutions designed and developed by OpenText and its subsidiaries. All OpenText employees are responsible to uphold and participate in this program.

Sources

The source of this overview document is the PSAP Standard Operating Procedure (SOP). This SOP is highly confidential in nature, for internal OpenText consumption only. This overview document represents the aspects that are able to be shared with OpenText customers and partners.

Introduction

OpenText is committed to the confidentiality, integrity, and availability of its customer information. OpenText believes that the foundation of a highly secure system is that the security is built in to the software from the initial stages of its concept, design, development, deployment, and beyond. In this respect, the PSAP attempts to inject security principles and ideals into all stages of the product lifecycle.



Concept and design

Application security requirements, specifications, and features

With a goal to incorporate security at the earliest possible phase of the product lifecycle, OpenText captures and strives to incorporate specific application security requirements during the concept/design phases of the product lifecycle.

These requirements are normally derived from industry standard best practice guidelines such as the OWASP Development Guide and Security Cheat Sheet Series projects. Some common application security requirements injected in to OpenText products fall into the following high-level categories:

- Identity management
- Authentication
- Session management
- Authorization/access control
- Data validation and representation
- Data protection, handling, and retention
- Cryptography
- Error handling and logging
- Business logic
- Client-side security
- Configuration and deployment
- Industry specific compliance standards, regulations, and privacy

OpenText also strives to give security related specifications and features the same importance as functional and performance related specifications and features. OpenText product teams continually research new security related threats, trends, laws, and regulations to stay ahead of competitors. An example of a regulation that OpenText is preparing for is the European Parliament's General Data Protection Regulation (GDPR). To learn how OpenText products can help with this new industry regulation, visit our website.

opentext™

Development

Secure coding principles

The goal of software security is to maintain the confidentiality, integrity, and availability of information resources in order to enable successful business operations. To help accomplish this goal, OpenText follows **secure development principles** through the entire product lifecycle. These include:

Minimize attack surface area: Every feature that is added to an application adds a certain amount of risk to the overall application. The aim for secure development is to reduce the overall risk by reducing the attack surface area.

Establish secure defaults: There are many ways to deliver an "out-of-the-box" experience for users. However, by default, the experience should be secure, and it should be up to the user to reduce their security—if they have permissions.

Apply least privilege: The principle of least privilege recommends that accounts have the least amount of privilege required to perform their business processes. This encompasses user rights and resource permissions such as CPU limits, memory, network, and file system permissions.

Apply defense-in-depth: The principle of defense-in-depth suggests that where one control would be reasonable, more controls that approach risks in different fashions are better. Controls, when used in depth, can make severe vulnerabilities extraordinarily difficult to exploit and thus unlikely to occur.

Fail securely: Applications regularly fail to process transactions for many reasons. How they fail can determine if an application is secure or not.

Don't trust third-party services/data: Many applications utilize the processing capabilities of third-party partners, who may employ different security policies and postures. It is unlikely that these third parties can be influenced or controlled, whether they are end users, external APIs, or data sources. Therefore, implicit trust of externally run systems is not recommended. All external systems should be treated in a similar fashion.

Separation of duties: A key fraud control is separation of duties. For example, someone who requests a computer cannot also sign for it, nor should they directly receive the computer. This prevents the user from requesting many computers and claiming they never arrived. Certain roles have different levels of trust than normal users. In particular, administrators are different to normal users in that they should not be users of the application.

Avoid security by obscurity: Security through obscurity is a weak security control and nearly always fails when it is the only control. This is not to say that keeping secrets is a bad idea, it simply means that the security of key systems should not be solely reliant upon keeping details hidden.

Keep security simple: Attack surface area and simplicity go hand in hand. Certain software engineering fads prefer overly complex approaches to what would otherwise be relatively straightforward and simple code. Developers should avoid the use of double negatives and complex architectures when a simpler approach would be faster and simpler.

Fix security issues correctly: Once a security issue has been identified, it is important to develop a test to identify it and to understand its root cause. When design patterns are used, it is likely that the security issue is widespread amongst all code bases, so developing the right fix without introducing regressions is essential.

Training and awareness

OpenText provides both mandatory and optional application security training to its developers and testers. Some example subjects include:

- Foundations of software security
- Foundations of Java platform security
- Foundations of JavaScript and HTML5 security
- Defensive programming for JavaEE, JavaScript, and HTML5
- OWASP top ten risk prevention
- Customized training for leveraged static and dynamic application security testing tools

Secure coding standards

OpenText follows industry standard secure development guidelines, such as:

- OWASP Development Guide
- OWASP Secure Coding Practices Quick Reference Guide
- OWASP Cheat Sheet Series
- CERN Security Checklist for Developers
- Microsoft® Guidelines for Writing Secure Code
- SEI CERT Coding Standards

Static application security testing (SAST)

All product teams are mandated to routinely incorporate SAST activities into their development procedures. Both automated and manual source code review approaches are utilized to accomplish this. All results are reviewed and qualified and any true positive findings are risk treated accordingly. OpenText relies upon industry standard methods and tools to fulfill the SAST requirement of PSAP.

Third-party component tracking

From a security perspective, OpenText reviews and keeps track of all third-party software libraries and versions incorporated with or used by our products and solutions. The goal of this endeavor is to prevent vulnerable third-party software from making its way into our products and to stay informed of third-party components that are reported to be vulnerable so that they can be patched.

opentext™

Testing and quality assurance

Dynamic application security testing (DAST)

All product teams are mandated to routinely incorporate DAST activities (application layer vulnerability testing) into their quality assurance and regression testing procedures. Combinations of automated and manual approaches are used to accomplish this. All results are reviewed and qualified and true positive findings are risk treated accordingly. OpenText relies upon industry standard methods and tools to fulfill the DAST requirement of PSAP.

Direct assessments

In addition to routine SAST and DAST, direct application security vulnerability assessments and penetration tests are regularly conducted against OpenText. These assessments are normally conducted by an internal Security Engineering team, separate from the individual product teams (as a second set of unbiased eyes), but may also be conducted by third parties for key product deployments. Customers also conduct their own assessments against their own deployed environments that host OpenText products and solutions. To support these efforts, OpenText will participate in these assessment activities upon customer request.

Direct assessments normally utilize testing checklists such as the OWASP Testing Guide to test the product for resiliency against the OWASP Top 10 Risks listed as follows:

A1-Injection	Injection flows, such as SQL, OS, and LDAP injection occur when untrusted data is sent to an interpreter as part of a command or query. The attacker's hostile data can trick the interpreter into executing unintended commands or accessing data without proper authorization.
A2-Broken authentication and session management	Application functions related to authentication and session management are often not implemented correctly, allowing attackers to compromise passwords, keys, or session tokens, or to exploit other implementation flaws to assume other users' identities.
A3-Cross-site scripting (XSS)	XSS flaws occur whenever an application takes untrusted data and sends it to a web browser without proper validation or escaping. XSS allows attackers to execute scripts in the victim's browser which can hijack user sessions, deface web sites, or redirect the user to malicious sites.
A4-Insecure direct object references	A direct object reference occurs when a developer exposes a reference to an internal implementation object, such as a file, directory, or database key. Without access control check or other protection, attackers can manipulate these references to access unauthorized data.
A5-Secuirty misconfiguration	Good security requires having a secure configuration defined and deployed for the application, frameworks, application server, web server, database server, and platform. Secure settings should be defined, implemented and maintained, as defaults are often insecure. Additionally, software should be kept up to date.

A6-Sensitive data exposure	Many web applications do not properly protect sensitive data, such as credit cards, taxIDs, and authentication credentials. Attackers may steal or modify such weakly protected data to conduct credit card fraud, identity theft, or other crimes. Sensitive data deserves extra protection such as encryption at rest or in transit, as well as special precautions when exchanged with the browser.
A7-Misssing function level access control	Most web applications verify function level access rights before making that functionality visible in the UI. However, applications need to preform the same access control checks on the server when each function is accessed. If requests are not verified, attackers will be able to forge requests in order to access functionality without proper authorization.
A8-Cross-Site Request Forgery (CSRF)	A CSRF attack forces a logged-on victim's browser to send a forged HTTP request, including the victim's session cookie and any other automatically included authentication information, to a vulnerable web application. This allows the attacker to force the victim's browser to generate requests the vulnerable application thinks are legitimate requests from the victim.
A9-Using components with known vulnerabilities	Components, such as libraries, frameworks, and other software modules, almost always run with full privileges. If a vulnerable component is exploited, such as an attack can facilitate serious data loss or server takeover. Applications using components with known vulnerabilities may undermine application defenses and enable a range of possible attacks and impacts.
A10-Unvalidated redirects and forwards	Web applications frequently redirect and forward users to other pages and websites, and use untrusted data to determine the destination pages. Without proper validation, attackers can redirect victims to phishing or malware sites, or us forward to access unauthorized pages.

Security documentation

OpenText strives to provide its customers with as much supporting material relating to secure configuration and hardening as possible. An example of the type of documentation made available is the "**Best practices content server application security hardening guide**" made available within the **Champion Toolkit** for each new release, posted in OpenText My Support.

Issue management and risk treatment

All security issues and vulnerability findings that are detected from any source (e.g., SAST, DAST, direct assessments, and customer reports) are analyzed for validity, assigned a security severity rating, logged against the appropriate project in the OpenText issue tracking system, prioritized, and risk treated.

Maintain and support

Open source and third-party component tracking

Open source and third-party components that are embedded or incorporated into OpenText products and solutions are tracked and regularly updated to help ensure that vulnerable components do not negatively impact the security of OpenText products.

OpenText routinely monitors public vulnerabilities, disclosure databases, and other resources for new threats or vulnerabilities in third-party components. If a threat or vulnerability is discovered, investigations are initiated to determine if any OpenText products are affected.

If an OpenText product is affected, then an appropriate risk treatment strategy is implemented.

Customer support security alerting

If a significant threat affects the security of a product, customers are notified through the OpenText security alerting process through OpenText My Support. Patches and mitigation instructions are provided as soon as available.

Customer reported issues

OpenText takes customer reported security issues very seriously. Any security related issues brought to the attention of OpenText employees are triaged immediately and a resolution is provided as soon as reasonably possible. OpenText customers are encouraged to report security issues through their normal customer support channels.

opentext™

Partnership and responsibility

Security Advocates

Every product has an assigned individual or group of individuals acting as a Security Advocate for their product. Assigned by Product and/or Development Management, the Security Advocate role is conferred upon subject matter experts for the given product.

Having assigned Security Advocates promotes ownership and responsibility for the security of the product and its important customer related assets.

Security Advocates are not responsible for performing all security related activities. Instead, they help to ensure that adequate resources and time are allocated to security related tasks, requirements, and initiatives.

In this respect, product security assurance is promoted and championed to all product team members. Security becomes visible to everyone instead of just a single team within the organization. This ensures that everyone is responsible for and participates in the securing of OpenText products and solutions.

OpenText Security Advocates are responsible for:

- Assigning responsibility to individuals and groups for security related tasks on the product team
- Promoting the incorporation of application security requirements into the product's concept and design phases by business owners
- Ensuring secure development principles are incorporated into the product's development phases
- Ensuring static application security testing is conducted during the product's development phases
- Ensuring dynamic application security testing is conducted during the product's QA and regression testing phases
- Ensuring open source and third-party component lists are tracked and kept up to date for the product and that regular monitoring of threats or vulnerabilities in those components are risk treated
- Ensuring that reported security issues, threats, and vulnerabilities relating to the product from any source are brought to the attention of business owners and risk treated
- Ensuring all product related support documentation is up to date with its security related information
- Disseminating security related announcements to their product teams

opentext™

Security Engineering team

A Security Engineering team also exists at OpenText. This team's role is to champion security in all products. Security Engineering strives to build long-term relationships with Security Advocates and their product teams, working closely to support and assist the teams in all security related tasks and endeavors.

Security Engineering's core responsibilities include:

- Promoting security in all products and secure software development practices
- Acting as custodians for the OpenText Secure Software Development Lifecycle Program (i.e., the PSAP)
- Tracking the security maturity of all products and reporting overall risk postures to Engineering Management
- Regularly liaising with and supporting Security Advocates and their product teams
- Providing security related subject matter expertise, SAST and DAST support, and training to all product teams for SAST and DAST related tools and activities
- Performing direct application security vulnerability assessments and penetration tests as required
- Tracking all vulnerabilities, threats, and customer reported security issues holistically and ensuring they are being risk treated according to their severity ratings
- Working with OpenText Customer Support and customers to investigate and seek resolution to customer reported security issues, questions, and concerns
- Working in cooperation with the OpenText Global Information Security team on various security related initiatives
- Keeping abreast of new security related threats and trends, attack techniques, tools, and methodologies

Privacy and Security Policy

OpenText is committed to protecting the personal data of our customers. To read our policy statement outlining our principles with respect to personal data collected, processed, and used via our website, visit:
http://www.opentext.com/who-we-are/copyright-information/site-privacy.

About OpenText

OpenText enables the digital world, creating a better way for organizations to work with information, on-premises or in the cloud. For more information about OpenText (NASDAQ: OTEX, TSX: OTEX), visit opentext.com.

Connect with us:

- OpenText CEO Mark Barrenechea's blog
- Twitter | LinkedIn | Facebook

opentext.com/contact

Copyright ©2017 Open Text. OpenText is a trademark or registered trademark of Open Text. The list of trademarks is not exhaustive of other trademarks. Registered trademarks, product names, company names, brands and service names mentioned herein are property of Open Text. All rights reserved. For more information, visit: http://www.opentext.com/2/global/site-copyright.html

SKU#