April 2023

PREVENTING DATA LEAKS FROM APPLICATION SCREEN-SHARING

Dave Zacks

Tim Szigeti

Ted Hulick

Carlos M. Pignataro

Stefano Giorcelli

PREVENTING DATA LEAKS FROM APPLICATION SCREEN-SHARING

AUTHORS:

Dave Zacks
Tim Szigeti
Ted Hulick
Carlos M. Pignataro
Stefano Giorcelli

ABSTRACT

According to a recent and comprehensive analysis of information security breaches, 23% of attacks are attributable to internal instances. Presented herein are techniques for protecting businesses against the sharing of confidential information within applications with unauthorized meeting participants. In particular, techniques presented herein restrict screen sharing of confidential information by preventing confidential content from being displayed on an unauthorized user's endpoint device during a collaboration session.

DETAILED DESCRIPTION

According to a recent and comprehensive analysis of information security breaches, the percentage of attacks attributable to internal incidences is 23%. This percentage has more than doubled since 2019. When third-party attacks/incidents (attributable for 25% of breaches) are added to internal incidents, nearly half of all information security breaches originate from authorized users (23% from internal users and 25% from third-parties). A commonly used internal data breach vector is the taking of screenshots of confidential data that is being shared via a collaboration application.

Rather than relying on keyword and/or steganographic insertions into document-centric content, techniques described herein make use of application security agents, such as a Multi-Tenant Agent (MTA). An MTA is typically utilized to provide Application Performance Monitoring (APM), but more recently has been repurposed to provide Runtime Application Self-Protection (RASP), Software Composition Analysis (SCA), and Interactive Application Security Testing (IAST) as part of a secure application product.

According to techniques provided herein, an application security agent may be enabled to run on confidential application(s) within an organization, providing these applications with both RASP and User Entity and Behavior Analytics (UEBA). The application security agents are centrally controlled and are aware of who a user is, what the user's role is, the Internet Protocol (IP) address of the endpoint that the user is using to access the application, etc. The application security agent is also aware of which transactions or requests for data are considered "confidential" or "sensitive" by the organization.

Techniques described herein provide for integration between the controller of the application security agents and the collaboration application. Figure 1, below, illustrates an integrated architecture of application security agents and a collaboration application.

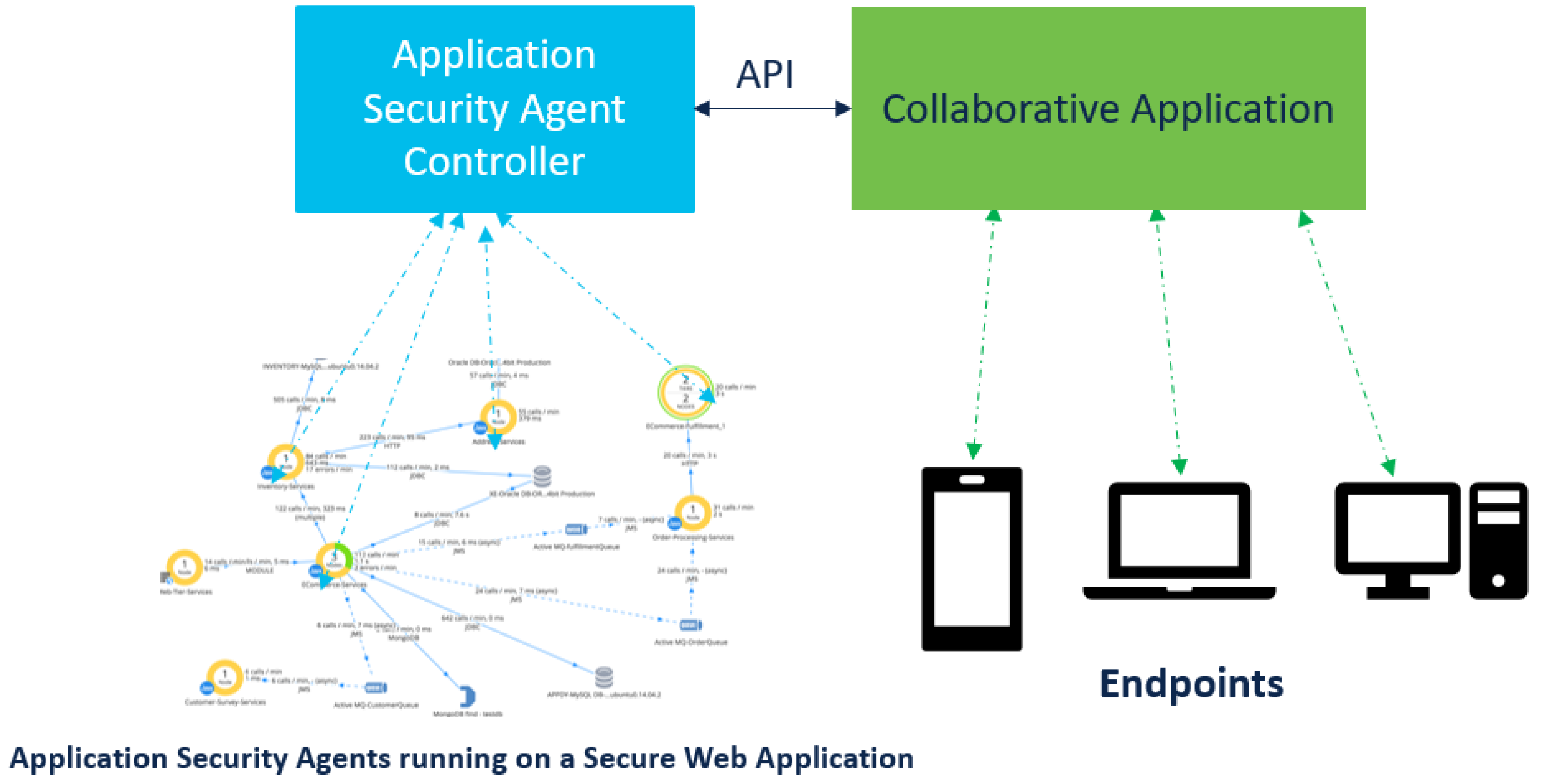
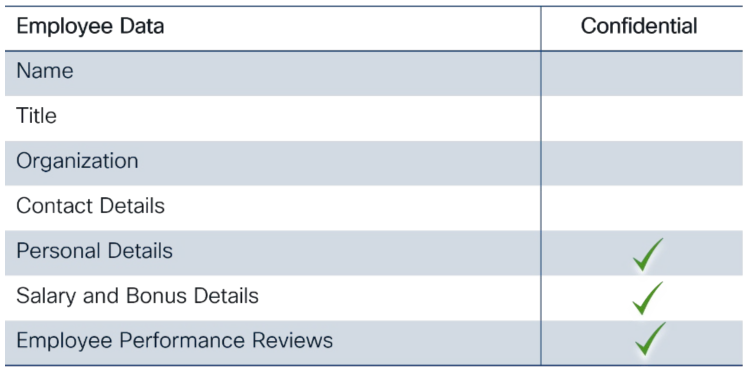

Application Security Agent Controller
API
Collaborative Application
Endpoints
Application Security Agents running on a Secure Web Application

Figure 1: Diagram of Example Integration

Similar integrations already exist for collaborative applications, allowing the collaborative applications to interface with policy servers to make different collaboration decisions. With the architecture illustrated in Figure 1 in place, techniques described herein are outlined in a number of steps. The first two steps are (Day 1) administrative tasks and the remaining five steps are (Day n) operational steps.

In administrative step 1, the application administrator expresses business intent policies into the application security agent controller indicating which transactions are to be considered "sensitive" or which requests for data are to be considered "confidential." Figure 2, below, illustrates an example of designating sensitive/confidential content that is accessible within an application.



Employee Data	Confidential
Name	
Title	
Organization	
Contact Details	
Personal Details	✓
Salary and Bonus Details	✓
Employee Performance Reviews	✓

Figure 2: Example of Designation of Confidential Content

As illustrated in Figure 2, the business intent may be a matrix of type of data and context of that data (e.g., project). For example, a timeline may be sensitive only in the context of Project X, but not in the context of Project Y. In addition, confidentiality is not necessarily a binary function (i.e., a YES or NO designation). Instead, gradient levels of confidentiality may be designated (such as Director-Level, VP-Level, SVP-level, etc.) or by role (project leaders, program managers, auditors, etc.). However, in the steps outlined below, for the sake of simplicity, content is simply labeled "confidential" (or not confidential).

According to techniques described herein, in administrative step 2, an administrator of the collaboration application designates the actions to be taken if confidential application content is attempted to be shared with a non-authorized viewer. The actions may include (but are not limited to) sending an alert (e.g. to the presenter, the application administrator, or other designated individuals), triggering a request for step-up authentication, blocking video content shared by the unauthorized participant, blocking audio content (while displaying an appropriate message to this effect) during the period that the confidential content is being shared, setting similar audio/video restrictions on the recording of the presentation for the given unauthorized user(s), and providing a security incident report to Security Information and Event Management (SIEM).

Figure 3, below, illustrates administrative steps 1 and 2. More specifically, Figure 3 illustrates defining administrative policies to prevent data leakage from the screen-sharing of confidential data.

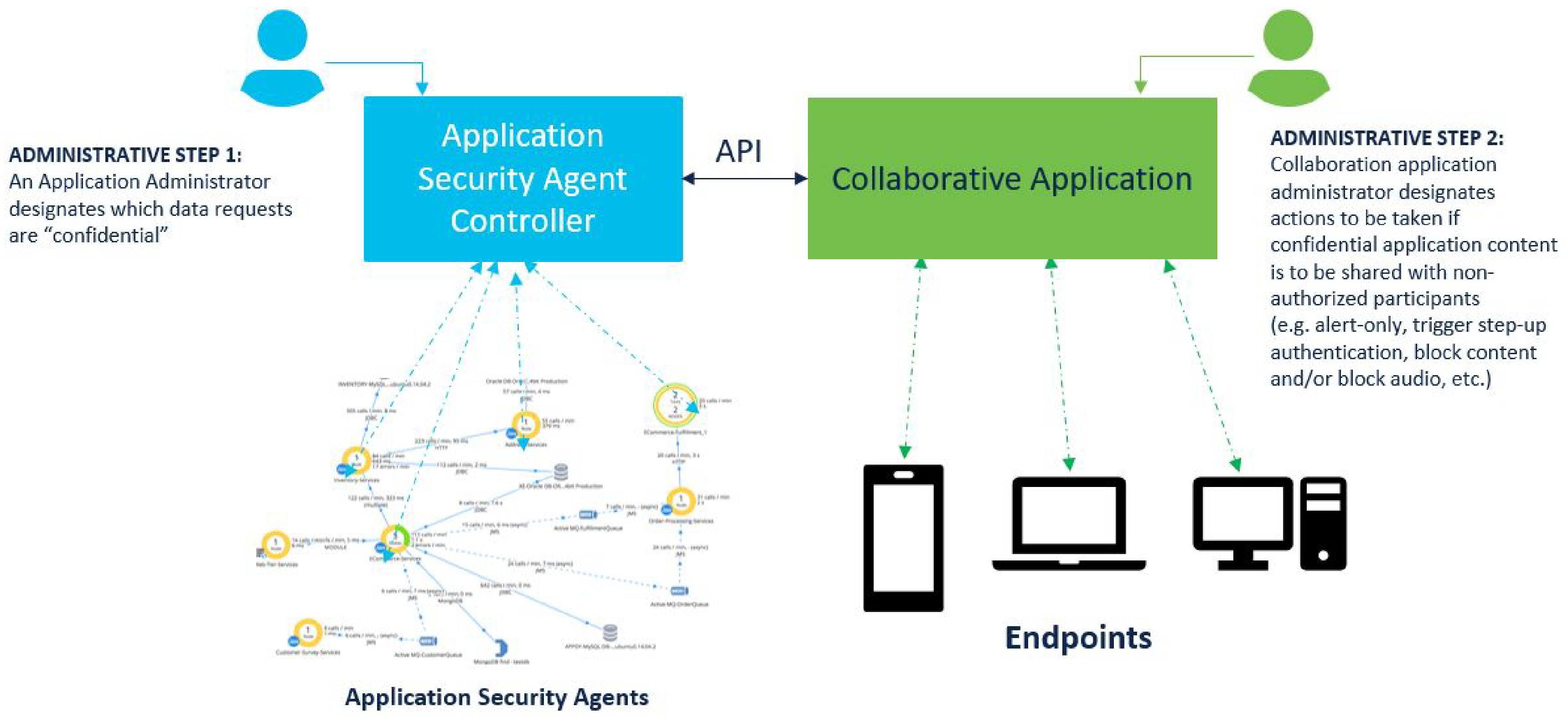

Application Security Agent Controller
API
Collaborative Application
ADMINISTRATIVE STEP 1:
An Application Administrator designates which data requests are “confidential”
ADMINISTRATIVE STEP 2:
Collaboration application administrator designates actions to be taken if confidential application content is to be shared with non-authorized participants (e.g. alert-only, trigger step-up authentication, block content and/or block audio, etc.)
Application Security Agents
Endpoints

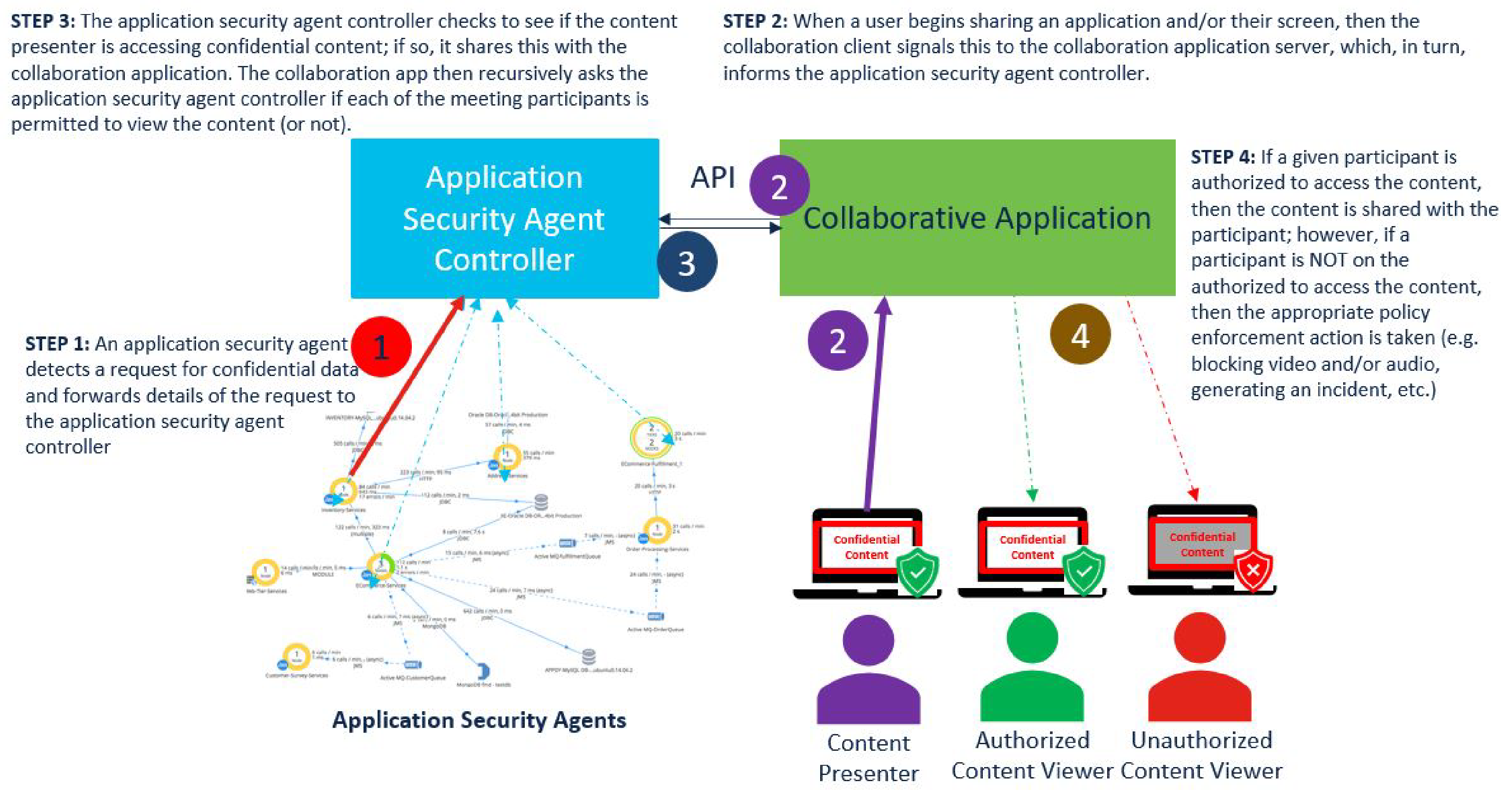
Figure 3: Diagram Defining Administrative Policies to Prevent Data Leakage

According to techniques described herein, there are four operational steps for preventing data leakage. In operational step 1, an application security agent detects a request for data and forwards details of the request to the application security agent controller. In operational step 2, when a user begins sharing an application and/or the user's screen, the collaboration client signals the sharing to the collaboration application server, which, in turn, informs the application security agent controller of the sharing.

In operational step 3, the application security agent controller checks to see if the content presenter is accessing confidential content. If the presenter is accessing confidential content, the application security agent controller shares this result with the collaboration application. The collaboration application then recursively asks the application security agent controller whether each of the participants is permitted to view the content. The application security agent controller returns the authorization result for each participant.

In operational step 4, if a given participant is authorized to view the content, then the content is shared with the participant. However, if a viewer is not authorized to access the content, then the appropriate policy enforcement action is taken (e.g. alerts, step-up authentication, blocking video and/or audio, generating an incident report, etc.).

Figure 4, below, illustrates operational steps 1-4 to prevent data leakage from application screen-sharing.



STEP 3: The application security agent controller checks to see if the content presenter is accessing confidential content; if so, it shares this with the collaboration application. The collaboration app then recursively asks the application security agent controller if each of the meeting participants is permitted to view the content (or not).

STEP 2: When a user begins sharing an application and/or their screen, then the collaboration client signals this to the collaboration application server, which, in turn, informs the application security agent controller.

STEP 4: If a given participant is authorized to access the content, then the content is shared with the participant; however, if a participant is NOT on the authorized to access the content, then the appropriate policy enforcement action is taken (e.g. blocking video and/or audio, generating an incident, etc.)

STEP 1: An application security agent detects a request for confidential data and forwards details of the request to the application security agent controller

Application Security Agent Controller
API
2
3
Collaborative Application
1
2
4
Confidential Content
Confidential Content
Confidential Content
Application Security Agents
Content Presenter
Authorized Content Viewer
Unauthorized Content Viewer

Figure 4: Diagram Illustrating Prevention of Data Leakage from Screen Sharing

Once the presenter is no longer sharing their screen or is no longer sharing confidential content via their screen, the system performs the same steps to inform the collaboration system that the screen sharing or screen sharing of confidential information is no longer taking place. At this point, any content-sharing policy that has been applied may be relaxed.

In summary, according to techniques described herein, a collaboration application is automatically informed when a user is sharing confidential data from within an application, without requiring the insertion of any keywords, watermarks or steganography into/over the confidential content combined with image-recognition decisions on the collaboration application. The collaboration application is dynamically informed of which meeting participants are (and are not) authorized to view the confidential information that a presenter has elected to share. Confidential content (e.g., audio and/or video) is dynamically restricted from being displayed on a given unauthorized user's endpoint device via a collaborative application, which prevents unauthorized users from being able to replicate the content (e.g., by taking a screenshot and/or printing, saving, copying, etc.) and thus potentially leaking sensitive data. In addition, the collaboration application is dynamically informed when confidential information is no longer being shared from within an application so that content-sharing policy restrictions can be relaxed.

Techniques described herein protect businesses against sharing confidential information within applications with unauthorized meeting participants. Techniques described herein do not rely on traditional document-centric methods, such as image recognition of keywords and/or steganography, which are rarely (if ever) available in such scenarios.