

Securing Collaborative Applications

Manoj R. Sastry, Michael J. Covington, Deepak J. Manohar
Corporate Technology Group, Intel Corporation

Abstract—Mobile technologies are experiencing rapid growth and adoption in both enterprise and consumer markets, as users demand unfettered access to resources, services, and data at any time and from any location. Increasingly capable mobile platforms and rich, new applications are enabling users to communicate and collaborate anytime and anywhere. However, concerns about the security and reliability of collaborative mobile applications on open platforms could impede the evolution and widespread adoption of these usage models. In this paper, we explore threats to open platforms in a mobile environment and propose an architecture to address security and reliability concerns for collaborative applications.

I. INTRODUCTION

Mobile devices, networks and applications are experiencing rapid growth, making it increasingly possible for consumers to stay connected to important information and resources at all times. The mobile applications that provide users with rich communication tools enable them to collaborate even if they are remote. Seamless Collaboration is an important mobile usage model that addresses the need for rich, real-time data, voice and video collaboration capabilities on mobile enterprise platforms. As more companies are distributed geographically, it becomes a challenge for employees to interact effectively. In such cases, collaborative applications can help reduce the need for face-to-face meetings and improve overall productivity. An example Seamless Collaboration application is one in which an employee's notebook computer serves as her Voice over Internet Protocol (VoIP) based phone, allowing the employee to collaborate with her team regardless of her location (in her office, at the airport, or at a business conference). Before these collaborative applications can be widely deployed in the enterprise, security measures must be in place to protect the confidential information that is exchanged.

In order to support an "anywhere, anytime" access paradigm, open mobile platforms continue to increase in complexity by incorporating new integrated hardware components and data-rich applications. However, this complexity does not evolve without some increased security and reliability concerns; these concerns may slow the evolution and widespread adoption of seamless collaboration usage models. Although the continuous integration of new components and capabilities on an open mobile platform is beneficial to the user, it also opens up new avenues for malicious attacks on the platform. Most security concerns are currently addressed through network security or mobile service providers that offer end-to-end encryption for the (voice) application. However, these solutions are often limited and application-specific; they still leave the platform open to vulnerabilities as new technologies and rich, emerging applications are added to the platform.

The mobile environment introduces new usage models accompanied by new threats. Our approach began with a thorough review of an application such as Seamless Collaboration. We performed a threat analysis and used that understanding to derive security requirements. The result of this research is an architecture that provides security and reliability for the Seamless Collaboration class of applications.

The remainder of this paper proceeds as follows: Section II presents the Seamless Collaboration usage model that motivated our research. Section III identifies the key findings of the threat analysis we performed on the Seamless Collaboration class of applications when running on an open mobile platform. In Section IV, we present our Seamless Collaboration Security Architecture; the proof-of-concept vehicle that we have implemented is described in Section V. Finally, related work is presented in Section VI and we conclude in Section VII.

II. MOTIVATION

Seamless Collaboration is an emerging usage model for the enterprise that addresses the need for rich collaborative tools on the mobile platform. Consider the scenario depicted in Figure 1. Alice wants to collaborate with remote colleagues while she is on a business trip. Alice is attending a conference that offers free wireless Internet access to attendees, so she connects to her company network through the public hotspot. To accomplish this, Alice turns on the wireless capability in her notebook and connects to the conference's wireless access point. Using the local Internet connection, she is able to connect to her corporate network and initiates a VoIP-based soft phone application on her notebook; this allows her to make and receive real-time voice calls from the same extension as her desk phone in the office. All calls made to her office's desk telephone are now routed to her notebook. Alice's collaboration tools allow her to have a single integrated messaging solution, with voice-mail, e-mail and instant messaging all in one place. In addition, she is able to exchange messages and interactive content (voice, video, electronic whiteboard, etc.) with her colleagues, even though she may be several hundred miles away.

This usage scenario highlights the benefits of collaborative applications. An open mobile platform provides a powerful computing environment and opportunities for rich integration with common applications that provide a familiar user interface to the user. As shown in the figure, users taking advantage of Seamless Collaboration on an open mobile platform have the ability to centralize all of their communication tools on a single platform. In addition, the integration of those tools allows for rich, new applications to be developed that help the

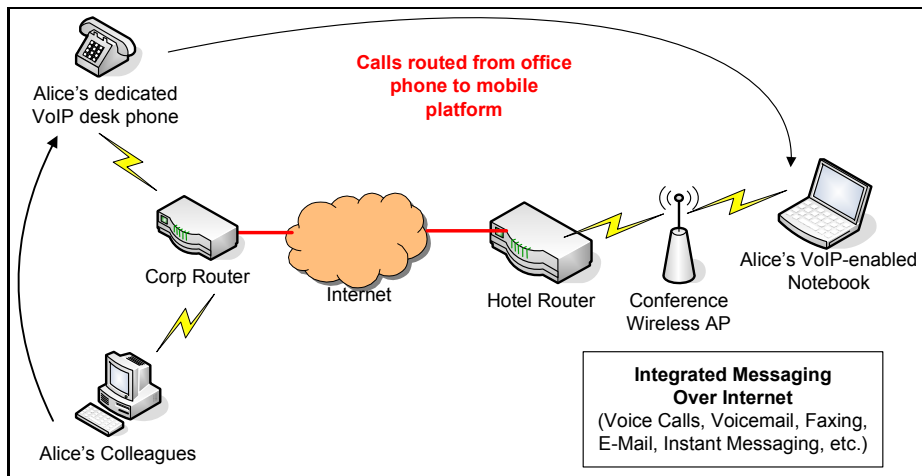


Fig. 1. Seamless Collaboration

user locate relevant information from a number of sources in a timely fashion.

On the other hand, there are also some disadvantages because the collaborative applications have to coexist with commodity software on an open mobile platform. Vulnerabilities exist in the commodity operating system and applications which can be exploited by viruses. These vulnerabilities impact both security and reliability. As an example, a virus that infects a collaborative application can not only steal confidential information, but also degrade voice quality by consuming large amount of CPU and memory resources.

Our work has focused on examining the unique threats that exist with collaborative applications on a mobile platform. Next, we describe our threat analysis approach and identify categories of threats that impact real-time VoIP applications.

III. THREAT ANALYSIS

We identified and analyzed threats to Seamless Collaboration applications on an open mobile platform. This analysis helped us derive the related security and reliability requirements. The threat modeling process used for this exercise started with identification of a mobile application or usage scenario in which the threats would be prevalent. For our research, VoIP telephony served as the primary Seamless Collaboration usage model around which we studied platform capabilities, assets, and existing vulnerabilities. The VoIP application supports many rich capabilities, including integrated voice, video and data collaboration. In addition, many other applications can be integrated with a VoIP telephony foundation, such as voicemail, e-mail, instant messaging, and interactive content (voice, video, electronic whiteboard, faxing.)

One of the most critical aspects of threat modeling involves the documentation and analysis of "assets" in the platform. Assets refer to components that must be protected. Some of the critical assets that are relevant to Seamless Collaboration on a mobile platform include the following:

- Voice data
- Session Initiation Protocol (SIP) authentication data
- Messaging data (e.g., voicemail, e-mail, and IM)

- Collaboration data shared by users as they interact
- Platform resources (e.g., CPU cycles, RAM, and storage)
- Identification data (e.g., passwords and account details)
- Valuable user data (e.g., addressbook, financial data)
- Valuable and confidential enterprise data
- IT-related data (e.g., network authentication credentials)

In addition to fully understanding the platform assets, our threat model investigated a variety of "entry points" through which an attacker could gain access to the platform. A high-level architectural representation of the mobile platform is shown in Figure 2. It shows two primary means for information to enter the mobile platform: user-oriented input/output and networking components.

In our threat analysis, threats are described according to the asset they seek to compromise, the "point of attack" or vulnerability they exploit, and their consequences (e.g., Denial of Service). From an economical perspective, the primary concern over threats in any situation focuses on the consequences and the damage potential of those threats. Figure 3 presents a collection of threat categories that were identified through our research. These threats are prevalent because the Seamless Collaboration application is susceptible to the vulnerabilities in commodity software that impact both reliability and security. In terms of reliability, we identified both availability and In-box QoS as qualities of the real-time VoIP application that could be impacted by a vulnerability. Due to the sensitive nature of data involved in these applications, we focused on threats that resulted in a loss of confidentiality, compromised integrity and unauthorized disclosure of sensitive information.

Given the nature of emerging mobile applications such as Seamless Collaboration, users are increasingly using mobile platforms to access these sensitive services through untrusted networks. In addition, more capabilities are being integrated into the notebook platform, which introduces new avenues for adversaries to attack the system. Through our research, we have examined threats to the Seamless Collaboration usage model that are unique to mobility. Further, we have examined how security threats such as viruses can impact the security and reliability of real-time collaborative applications.

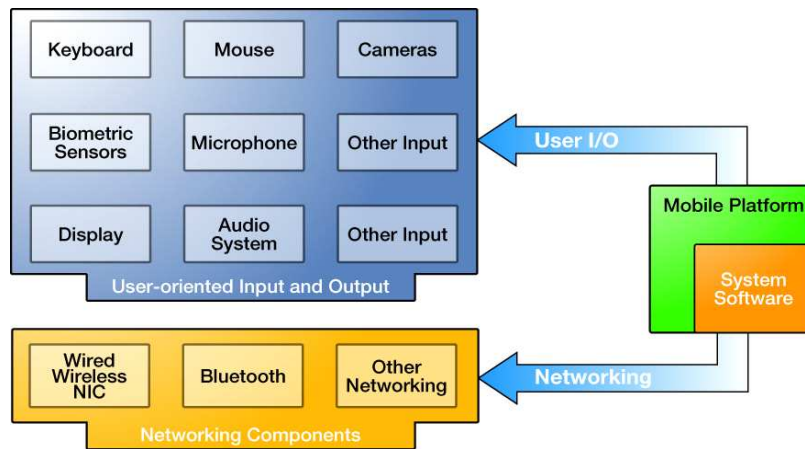


Fig. 2. Points of Attack

VoIP Applications are impacted by co-existing with Commodity Software		
SYSTEM IMPACT	CATEGORY	EXAMPLE
Reliability	Availability	VoIP call interrupted by O/S crash
	In-Box QoS	Resource intensive application degrades voice quality
Security	Confidentiality	Vulnerability allows adversary to eavesdrop on VoIP call
	Integrity	Vulnerability allows adversary to tamper with security configuration information (Secret key compromised)
	Unauthorized Access	Vulnerability leads to unauthorized access to secrets on the platform (voice, video, data)

Fig. 3. Impact of Threats to Seamless Collaboration

IV. SEAMLESS COLLABORATION SECURITY ARCHITECTURE

To address the threats discussed in the previous section, we developed the following approach:

First, we analyzed the components that made up the Seamless Collaboration application. We identified two categories of application components, namely critical and non-critical from a security and performance perspective. Figure 4 lists the critical and non-critical components of a collaborative application. The critical components include the security, communication and performance related components. The non-critical components include the user interface and user policy components. We split the monolithic Seamless Collaboration application into critical and non-critical components and we separated them from each other.

Second, we designed a low-latency communication channel for the critical and non-critical components of the Seamless Collaboration application to communicate with each other.

Finally, we defined a set of security services to secure the communication channel between the critical and non-critical components.

Based on the approach outlined in the previous paragraphs,

we developed an architecture to provide security and reliability to a Seamless Collaboration application. In the following, we describe the details of this architecture.

Figure 5 depicts the security architecture we developed for Seamless Collaboration. The platform hardware includes capabilities that support virtualization and security (e.g. a TPM [1]). The Service Domain is a special privileged domain that provides support for device virtualization and presents virtual device models to the guest domains. The Commodity domain, Seamless Collaboration Application and Seamless Collaboration Engine are guest domains. All commodity software including the operating system and the applications reside in the Commodity Domain.

We split the monolithic Seamless Collaboration application into critical and non-critical components, and protected the critical components of the application such as the VoIP SIP communication stack within the Seamless Collaboration Engine domain. We moved the non-critical components of the application such as the GUI to the Seamless Collaboration Application domain. Access to the Seamless Collaboration Engine domain was restricted to the administrator. One example of an administrator in the enterprise environment is the

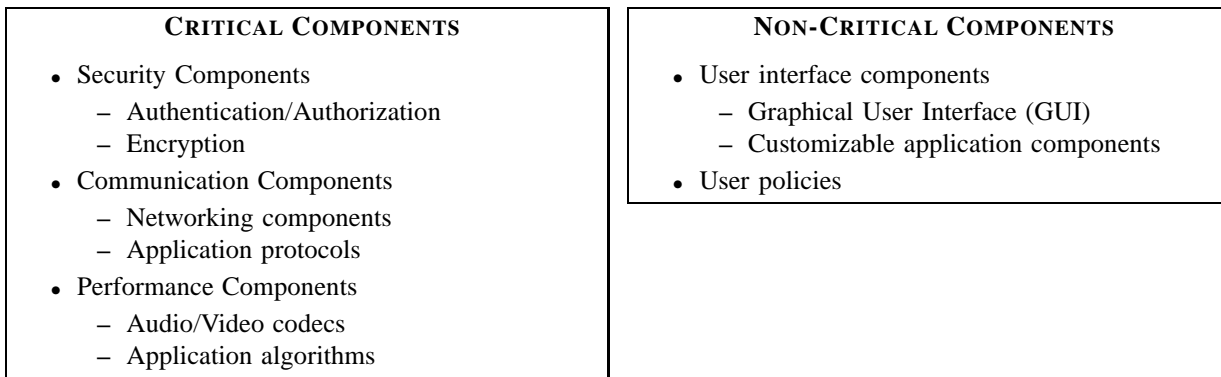


Fig. 4. Critical, Non-Critical Components of Collaborative Applications

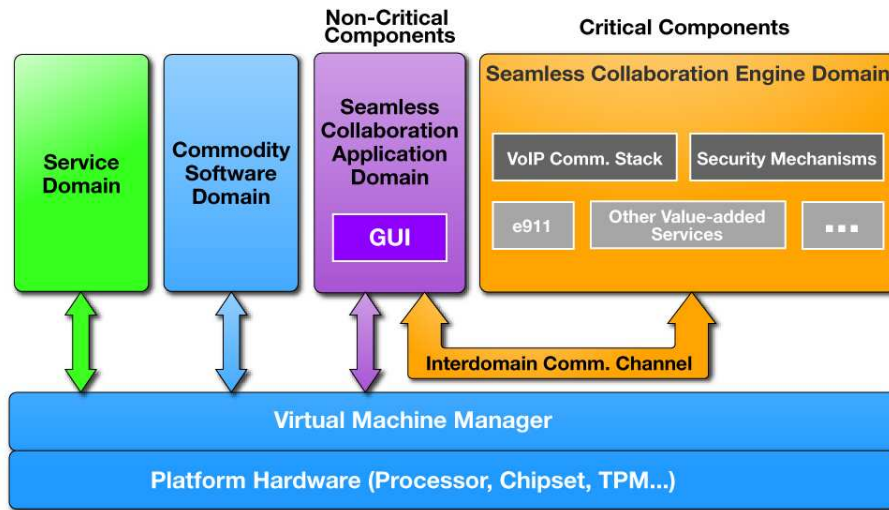


Fig. 5. Seamless Collaboration Security Architecture

Information Technology (IT) department. The user has access only to the Seamless Collaboration Application domain.

Next, we designed a low-latency communication channel for the critical components in the Seamless Collaboration Engine domain to communicate with the non-critical components in the Seamless Collaboration Application domain. The interdomain communication channel depicted in Figure 6 includes two primary components that enable communication between two domains. The first component is in the user-space and is called User-level Translation Layer (UTL). The second component resides in the kernel and is called Kernel-level Translation Layer (KTL).

The UTL is a user-level component and is responsible for transferring control from user-level to the kernel-level and responding to calls triggered by the kernel-level code. For instance, in the case of a Xen-based domain running the Linux operating system, the UTL includes code for the invocation of system calls and signal handler code to handle signals raised by the kernel.

The KTL is a kernel-level component and is responsible for sending notifications to the other domain (using event channels in Xen) and responding to notifications sent by the other domain (using VIRQs in Xen), transferring control from kernel-level code to the user-level code and data transfer from kernel-level to user-level and vice versa. For example, in the

case of a Xen domain running the Linux operating system, the KTL includes code for implementation of system calls to raise event channel-based notifications, code for handling VIRQs to respond to event channel notifications sent by the other domain and code to raise signals from the kernel-space to the user-space.

Finally, as outlined in our approach, we defined a set of security services to secure the communication channel between the critical and non-critical components of the Seamless Collaboration application:

- **Integrity Checking:** This service utilizes the capability of a TPM to ensure that the integrity of the Application Domain is intact. This service defends against a compromised Seamless Collaboration application. For instance, a Seamless Collaboration Application domain could have been attacked by a virus that alters messages sent by that domain to the Seamless Collaboration Engine Domain. Integrity Check enables the Seamless Collaboration Engine domain to detect if the messages were received by a compromised domain.
- **Parameter Checking:** This service checks if the input values used by the Application Domain are within the specified range. This helps provide immunity against buffer overflow problems caused by out-of-range input

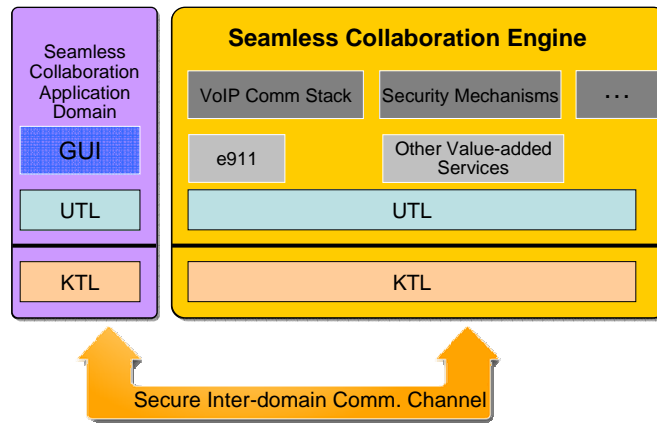


Fig. 6. Inter-Domain Communication Channel

values.

- Encryption: The traffic between the Seamless Collaboration Engine domain and the Seamless Collaboration Application domain is protected by encrypting it using the encryption algorithms specified by the Encryption service.

Having described the details of an architecture that supports Secure and Reliable Seamless Collaboration, we now identify the benefits provided by this architecture:

- Immunity against attacks that compromise the critical components of the collaborative applications is provided. For instance, in the case of Seamless Collaboration, attacks such as Call eavesdropping, Caller-ID spoofing are prevented.
- Reliability issues caused by commodity software are addressed by this architecture. For instance, even if the commodity software crashes, the collaborative application is unaffected. Further, malicious software exploiting vulnerabilities in commodity software cannot impact the latency requirements of the collaborative applications.
- Given that the Seamless Collaboration Engine domain is provided and controlled by a service provider, the service provider now has the opportunity to provide Secure Value-added Services at the endpoint. One example of a secure value-added service is E911, where current location information is reported when a VoIP call is initiated. This information is more useful than reporting the static home address information (collected during the registration process), especially when the call is placed by a mobile user.
- The user can access and modify the contents of the Seamless Collaboration Application domain. This provides the user with the flexibility to customize this domain with the most appropriate collaborative application front-end. In fact, the user could potentially have multiple such application domains talking to the same Seamless Collaboration Engine domain.
- An authorized entity, such as the Information Technology department, controls updates to the critical components of the Seamless Collaboration application. Additionally, our architecture allows the authorized entity to enforce

policies such as verifying integrity of the Engine domain at the end-point before processing a VoIP call.

V. PROOF-OF-CONCEPT

We have developed a proof-of-concept vehicle using the open-source VoIP software-based phone application called Linphone [2] on a hardware platform with support for virtualization and the Xen [3] hypervisor. In this section, we will present an overview of the Linphone architecture.

A. Linphone Architecture

Using the categorization of components described in Figure 4 as a guideline, we identified the critical and non-critical components in the Linphone architecture. Figure 7 depicts the non-critical and critical components of the Linphone application. Since our architecture requires that the critical components and the non-critical components be run in two separate domains, we needed to make the application aware of this split. We decided to perform this translation at the Core API layer. This layer is very suitable for splitting and running in two separate domains because it is the interface through which the UI and GUI components interact with the rest of the Linphone components. Therefore, we have two Core API layers one each in the non-critical and critical components.

The Core API Layer of Linphone uses the API shown in Figure 8 to interact with the KTL mechanism responsible for transferring data to the critical Linphone components. Changes to the Core API Layer include the introduction of signal handlers to handle signals from the KTL, invocations of KTL using system calls and security services such as integrity checking. These changes to the Core API comprise the UTL.

The API for the two new system calls that we introduced are described below.

```
int sys_send(char func_opcode,
             char req_resp,
             int s_no,
             char *params/result)
```

The `sys_send` function, depicted in Figure 8, is used for sending notifications from one domain to another. For the inter-domain communication, we had to provide a technique

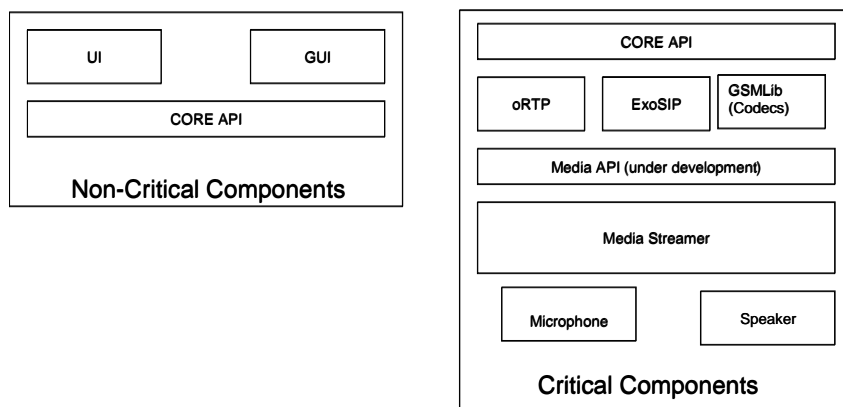


Fig. 7. Non-Critical and Critical Components of Linphone

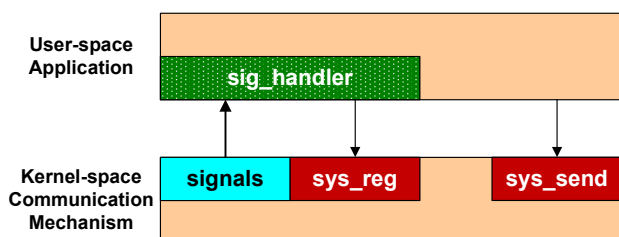


Fig. 8. Linphone Core API

for the two communicating process each in a separate domain to invoke functions and receive responses for them. Hence the first parameter `func_opcode` in the `sys_send` function uniquely identifies the function. The second parameter `req_resp` denotes whether the function is being invoked or the response for the function is being passed. The third parameter is used to correlate between requests and responses. The final parameter is used to pass the function parameters or the result of the function invocation. The function is non-blocking and returns either success or reports an error. The errors could signify that there are too many outstanding requests, the remote domain is not operational, etc.

```
int sys_reg(int pid)
```

The `sys_reg` function is used to register the application’s signal handler with the in-kernel mechanism. The `sys_reg` function expects the application’s process id as a parameter. The in-kernel mechanism needs this information before any signal can be delivered. As a result, user-space applications will use `sys_send` to send any notifications to the other remote domain and the signaling mechanism will be used to receive any notifications from the other domain.

B. Linphone CoreAPI Layer Execution Sequence

Figure 9 shows data communication between the Seamless Collaboration Application Domain (domain A) and the Seamless Collaboration Engine Domain (domain B). It shows the sequence of steps involved during the registration of Domain A with the KTL and the transfer of data from Domain A to Domain B. The registration phase indicated by the shaded arrows begins by the Linphone CoreAPI layer registering the

application buffer and the signal handler with the KTL layer and requesting a certain number of pages to be shared between domain A and domain B. The non-shaded arrows indicate the sequence of steps that transpire when Domain A transfers some data to Domain B. When the CoreAPI layer in Domain A receives a certain function call it passes this request onto the KTL using the `sys_send` system call described above. This system call packs this request into a well-formatted structure and requests the KTL to copy certain number of bytes to be transferred to Domain B. The KTL upon checking the policy copies this data onto the shared pages and sends a notification to Domain B. The event channel mechanism in Xen (hypervisor) raises this notification as an IRQ in Domain B. The IRQ handler in the KTL in Domain B serves this IRQ by checking the policy and deciding if the data needs to be copied from the shared pages onto the local pages and determining if the application needs to be signaled.

In this section, we have described a proof-of-concept based on open-source Linphone. This proof-of-concept is as an example implementation of our architecture that provides security and reliability for collaborative applications. Next, we will discuss related work in this area.

VI. RELATED WORK

Current solutions for addressing VoIP security are not adequate. In this paper, we have introduced an approach to secure a real-time collaborative VoIP application such as Seamless Collaboration. In particular, we have illustrated how a monolithic application can be partitioned into critical and non-critical components where the partitioning is enforced by hardware-based virtualization.

An important defensive programming technique is encapsulating the security-sensitive components within small, simple components. These components can then be more easily verified. Kilpatrick [4] has presented Privman, a library that makes privilege separation easy, therefore allowing security sensitive components of an application to be separated into a different process from the bug-prone components. Mistakes in the majority of the application will not and cannot result in total compromise of the entire application. Hence, attackers would be unable to compromise the system.

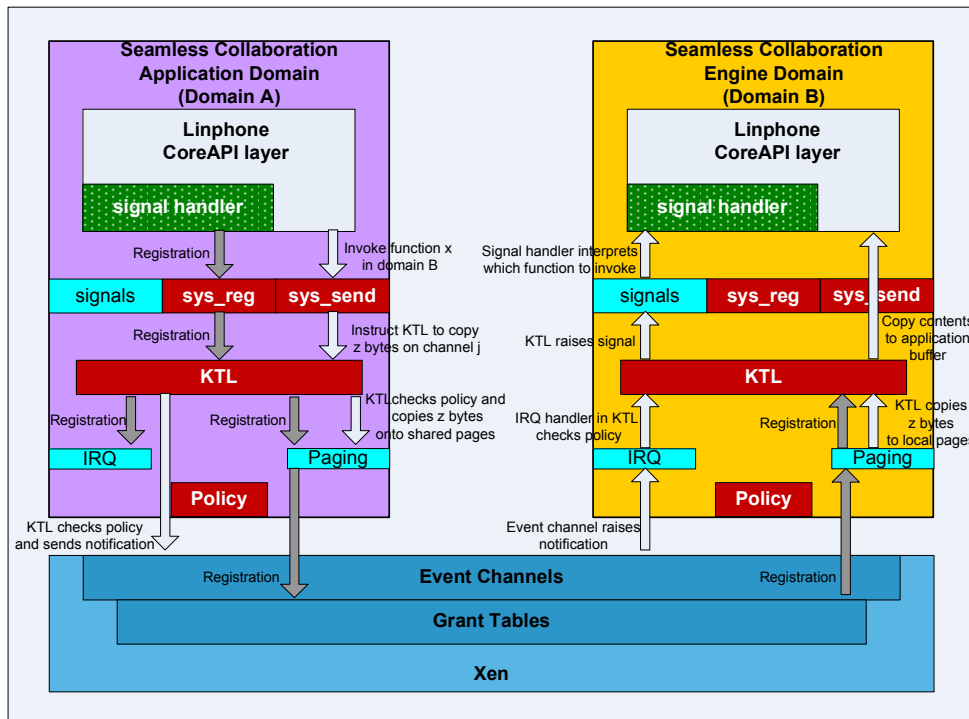


Fig. 9. Sequence of Data communication

Using a similar technique, we have protected critical security and performance-sensitive components of a real-time collaborative application using virtualization enforced by hardware. Virtualization has been applied to operating systems both commercially and in research for nearly thirty years, including work from IBM¹ [5], VMware¹ [6], Connectix¹ [7], and the Xen open source community [3], [8]. Additionally, we utilize the security capabilities provided by TPM hardware to provide integrity for the critical components.

Previous research has indicated that isolating applications from one another provides increased security on the platform. We take a novel approach by further partitioning a monolithic collaborative application to provide increased security assurances. At the same time, our architecture helps the user retain flexibility in being able to configure the application's user interface and user-level policy.

VII. CONCLUSION

We see an emergence of mobile applications that are enabling the user to engage in rich, real-time communication. VoIP-based Seamless Collaboration is one such application that enables business users to collaborate effectively even when the users are on the move or distributed geographically.

In general, open mobile platforms are becoming increasingly complex as new technologies to provide connectivity and rich user experience are being integrated into the platform. As platform complexity increases, the attack surface for compromising the platform increases. Collaborative applications such as Seamless Collaboration involve confidential information that can be valuable to an adversary. Further, the mobile platform is often used in untrusted physical and network

domains that makes it susceptible to attacks. All these factors combine to present some unique threats to mobile platforms.

Enabling such rich mobile collaborative applications on an open mobile platform has its advantages in terms of familiar user environment and the potential for integration with other applications. However, these collaborative applications are subject to security and reliability issues since they have to co-exist with commodity software that has vulnerabilities.

We have described our approach to address security and reliability problems in real-time VoIP applications while retaining user flexibility to configure an application's user interface and user-level policy. Additionally, this architecture provides an opportunity for an administrator to offer secure services at the end-point. We verified this architecture by developing a proof-of-concept using the open-source VoIP soft-phone Linphone and have presented the details of our prototype.

REFERENCES

- [1] Trusted Computing Group. TCG specifications. TCG Website, 2005. Available from: <http://www.trustedcomputinggroup.org/home/>.
- [2] Linphone Open Source Community. Linphone soft-phone application software. Linphone Website, 2006. Available from: <http://www.linphone.org/>.
- [3] Xen Open Source Community. Xen virtual machine monitor. Xen Website, 2006. Available from: <http://xen.sourceforge.net/>.
- [4] Douglas Kilpatrick. Privman: A library for partitioning applications. In *USENIX 2003 Annual Technical Conference, FREENIX Track*, pages 273–284. USENIX, June 2003.
- [5] P.H. Gum. System/370 extended architecture: Facilities for virtual machines. *IBM Journal of Research and Development*, 27(6):530–544, November 1983.
- [6] VMware. VMware product documentation. VMware Website, 2006. Available from: <http://www.vmware.com/>.
- [7] Connectix. Product overview: Connectix virtual server, 2003. Available from: <http://www.connectix.com/products/vs.html>.

- [8] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebar, Ian Pratt, and Andrew Warfield. Xen and the art of virtualization. In *Proceedings of ACM SIGOPS Symposium on Operating Systems Principles (SOSP)*, October 2003.

NOTES

¹Intel is a registered trademark of Intel Corporation in the United States and other jurisdictions. Other names and brands may be claimed as the property of others.