Project Title: Querying Invisible Objects: A Programming Framework for Data-Intensive, Privacy-Preserving Applications

Presentation Date: [Date]  Presentation Type: New (New, Status, Final)  Project Budget: $40K

Center: Security & Software Engineering Research Center  Center Director: Wayne Zage

Site (University): Virginia Tech  Co-Director: 

Project PI: Dr. Eli Tilevich  Email: tilevich@cs.vt.edu  Phone: (540) 231-3475

Project description:
We propose a novel programming framework, Object Expiration (or ObEx for short), for working with sensitive data that needs to be transferred across distributed managed execution environments, such as the JVM or Android. While end-users require that their sensitive data be kept private, mobile and IoT application developers want to use the data to provide better user experience and intelligent services. The proposed programming framework will effectively reconcile these two seemingly conflicting perspectives. Specifically, the framework will keep the values of sensitive data invisible, while providing clients with configurable interfaces to query for various statistical data properties. Further privacy protection will be provided via differential privacy, applied to the sensitive data exposed via the statistical query interfaces. A lightweight distributed runtime system will reliably enforce the sensitive data’s lifecycle and how its properties can be accessed. This novel programming mechanism will support the engineering of distributed applications that keep sensitive data private, while leveraging the data’s properties to provide intelligent mobile services.

Experimental plan:
I. Design and Implementation:
(1) Programming Abstractions and Software Framework
We plan to concretely realize the proposed concepts behind ObEx as a software framework with an intuitive API, implemented in a managed execution environment, such as the JVM accessed via a Java or Scala API.

(2) Distributed Runtime
Since ObEx objects are intended for distributed execution environments, so as to benefit extant and emerging mobile and IoT applications, we plan to support ObEx via a portable, lightweight, distributed runtime system, which will be installed on every device hosting ObEx objects. The runtime will comprise two main components: the local operations that securely store sensitive data and reliably destroy it as dictated by a given expiration policy; and the middleware system that will encrypt/decrypt/serialize ObEx objects whenever they need to be transferred across the network between different devices. The distributed runtime will also ensure the correctness of following the expiration policy in place, in the presence of transferring ObEx objects between devices.

II. Evaluation:
(1) Privacy Preservation
To evaluate how ObEx improves user privacy, we plan to experiment with known attack models to determine how resilient ObEx objects are against them. Specifically, we seek to ascertain two key properties: (a) ObEx objects are resilient against attacks, trying to maliciously gain access to the sensitive data, and (b) Using ObEx objects does not render the system vulnerable to new exploits. To that end, we will examine how vulnerable ObEx objects are to attacks before their sensitive data is cleared. In addition, we plan to first investigate existing attack models, such as Man-in-the-middle (MITM) attack and the malicious modification of bytecode, and then we plan to study how the ObEx objects could help protect against these attacks.

(2) Performance
We plan to compare the performance of ObEx objects with that of regular Java objects, including the speed of instantiation, serialization, and network transmission. Meanwhile, we will investigate the memory usage of the ObEx objects and compare it with regular Java objects. For example, to measure the actual memory consumed, we can continuously instantiate an ObEx object to the point of exhausting the system memory.

(3) Programming Expressiveness
To assess the expressiveness of the ObEx programming model, our evaluation will focus on answering the following questions:
   a) How intuitive is the provided API for the programmer to express realistic lifecycle policies?
   b) How can one determine the number and type of queries against an ObEx object clients can make to preserve safety and privacy?
   c) How well does the API design accommodate the volatilities associated with distributed execution?

Related work elsewhere and/or in the center:
The proposed ObEx objects are conceptually related to prior research in areas including (1) authentication and access control, (2) self-destructing data, (3) language runtime protection mechanisms, and (4) differential privacy, which we describe in turn next.
(1) Authentication and access control

Holford et al. presented a self-defending object (SDO) that can authenticate users while invoking a method [1]. An authentication token is passed as a parameter to the object’s public methods to be able to examine whether the caller has the permission to invoke the method.

(2) Self-destructing data

Roxana Geambasu et al. proposed a self-destruction system called Vanish, which periodically destroys expired access keys, so as to prevent all further access to the encrypted information [2]. Based on a similar concept, Jinbo Xiong et al. proposed an ABE-based secure document self-destruction (ADS) scheme for the sensitive documents in the cloud [3]. Meanwhile, Fengshun Yue et al. created a similar self-destruction scheme called SSDD for electronic data [4]. However, Scott Wolchok et al. pointed out that the Vanish approach can be compromised by attackers gaining the keys by continuously crawling their storage [5]. Although Lingfang Zeng et al. improved the Vanish approach to avoid the sniffer attacks [6], their approach was still confined to an external security framework rather than being integrated with a programming language runtime system.

(3) Programming language protection mechanism API

The Java standard API provides an internal protection mechanism for sensitive data via the signedObject and the sealedObject mechanisms [7]. The signedObject stores a signature, thus protecting its serializable representation. Without the valid digital signature, the protected object cannot be extracted. The sealedObject, on the other hand, encrypts the original object, encapsulating the result with a cryptographic cipher. However, once the original object has been recovered, the sensitive data is no longer protected. Meanwhile, since the implementation of these mechanisms is solely in the Java layer, a native library could be used to intercept and compromise the security mechanisms provided by these objects.

(4) Differential privacy

After Dwork et al. [8] first introduced differential privacy as a mechanism to guarantee user data privacy, it gradually has been embraced by both industry and academia. PSI enables non-experts to conveniently share and explore privacy-sensitive datasets that support differentially private queries [9]. Oblivm auto-generates cryptographic implementations by using a custom compiler for secure computation [10]. In addition, by using randomized response, RAPPOR collects information from end-user client software with differential privacy guarantee [11]. ADS ensures the communication security between clients and untrusted servers [12].

As compared with these works, the proposed ObEx approach relies neither on permissions nor on encryption. The objects and all their copies will reliably destroy the sensitive data after a given time period or number of accesses. In the meantime, the actual data management operations have been moved entirely to the runtime system layer, so as to avoid the threats emanating from the interception of bytecode. Hence, the proposed ObEx objects have potential to improve data security and privacy in all managed execution environments, with IoT and mobile applications being particularly promising as an application area. To maximize the positive impact, we plan to advocate that the support of ObEx objects be incorporated into the Java Virtual Machine and exposed to all Java developers via a standard API.


**How this project differs from related work:**

1) We extend the concept of a programming language object’s lifecycle and present a novel programming model with a policy-based, runtime-assured, lifecycle enforcement mechanism that ensures the security and privacy of sensitive data in distributed managed language execution environments.

2) We also embrace differential privacy as an internal protection mechanism to revisit the problem of preserving user privacy.

**Milestones planned for the upcoming year:**

1) Technical background analysis and literature review (1st quarter)
2) Design and Implementation (2nd quarter)
3) Evaluation (3rd quarter)
4) Publishing the findings in high-quality conference proceedings and archival journals (4th quarter)

**Deliverables planned for the upcoming year:**

1) Design and Implement of the proposed ObEx objects on a major mobile platform (e.g., Android)
2) Publish papers describing the technology in data security and language design/impl. conferences
3) Present the findings at the annual S2ERC Showcase

**How the project may be transformative and/or benefit society:**

Based on our preliminary work and the study of the related state of the art, we argue that integrating the proposed ObEx programming mechanism with the Java Virtual Machine (JVM) can improve the security and privacy of data in IoT and mobile applications, particularly in the edge computing environments. To that end, we can take advantage of the newly introduced SuLong framework, which enables the execution of native C/C++ code on the JVM by means of LLVM translation and virtualization. Once the runtime of proposed ObEx framework is integrated with the JVM, its derivative technologies, such as the Android platform, will immediately enjoy the ObEx privacy-enhancing benefits.

**Research areas of expertise needed for project success:**

1) Software Engineering
2) Software Security and Privacy (authentication and data privacy)
3) Programming Language Implementation (virtual machines, language design, runtime systems)

**Potential member company benefits:**

The goal of this project is to create a novel programming mechanism exposed via an intuitive standard API supported by a portable distributed runtime system to be potentially integrated with the built-in virtual runtimes of mobile and IoT devices. Any company facing the challenges of ensuring the security and privacy of pervasive sensitive data, commonly generated by sensors, would benefit from this solution, which will provide an assured and standardized mechanism for improving the security and privacy of that type of data.

**Progress to date:**

We have prototyped an initial proof of concept of ObEx objects and tested their basic operations on example toy applications. These initial results give us confidence in our ability to execute the proposed research plan and to deliver all the expected benefits.

| Project Start Date: 11/15/2017 | Estimated Knowledge Transfer Date: 11/16/2018 |

The Executive Summary is used by corporate stakeholders in evaluating the value of their leveraged investment in the center and its projects. It also enables stakeholders to discuss and decide on the projects that provide value to their respective organizations. Ideally, the tool is completed and shared in advance of IAB meetings to help enable rational decision making.