Privacy-Preserving Object Detection with Veracruz

Mathias Brossard*, Guilhem Bryant*, Xinxin Fan†, Alexandre Ferreira*, Edmund Grimley-Evans*,
Christopher Haster*, Derek Miller*, Dominic P. Mulligan*, Hugo J.M. Vincent*, Shale Xiong*, Lei Xu¶
*Systems Research Group, Arm Ltd, Cambridge UK & Austin, TX
†IoTeX, Menlo Park, CA 94025
¶The Department of Computer Science, Kent State University, Kent, OH 44240
Email: *{firstname.lastname}@arm.com †xinxin@iotex.io ¶lxu12@kent.edu

***Abstract*—Ucam is a user-centric, blockchain-based and end-to-end secure home IP camera system designed by IoTeX. In a Ucam system, all the video clips captured by the camera are encrypted using a user-controlled symmetric key before storing them in the cloud. In this demo paper, we describe the system architecture and implementation of a privacy-preserving object detection proof-of-concept (PoC) for the Ucam system using Veracruz, a confidential computing framework developed by Arm Research. The resulting PoC demonstrates the viability of bringing the object detection capability to the Ucam system without leaking users sensitive information.**

***Index Terms*—End-to-End Encryption, Object Detection, Confidential Computing, Privacy-preserving Computing, AWS Nitro System**

I. Introduction

Ucam [1], [3] is a user-centric, blockchain-based and end-to-end secure home IP camera system designed by IoTeX [2] to address a number of vulnerabilities in the existing solutions. One of salient features of Ucam is the realization of end-to-end encryption that utilizes user-specified secret keys to encrypt video clips and live streaming videos. While the end-to-end encryption offers strong privacy protection and ensures that only camera owners are able to access the videos captured by their devices, it becomes a challenge for providing the advanced object detection capability for Ucam customers.

To enable object detection on encrypted video clips, we implement a privacy-preserving object detection proof-of-concept (PoC) for the Ucam system in this demo paper. Our PoC is built upon Veracruz, a privacy-preserving collaborative computing framework developed by Arm Research [4], [5], and leverages the AWS Nitro system as well as Kubernetes [6] to run the computation securely and in a scalable way. The PoC demonstrates the viability of adding the object detection feature to the Ucam system without jeopardizing camera owners' privacy.

II. Preliminaries

A. Veracruz: Privacy-Preserving Collaborative Computing

Veracruz [4] is a framework for defining and deploying collaborative, privacy-preserving computations amongst a group of mutually mistrusting individuals. Veracruz uses strong isolation technology (a mixture of trusted hardware and high-assurance hypervisor-based isolation), along with remote attestation protocols, to establish a safe, "neutral ground" within which a collaborative computation takes place on an untrusted device. More specifically, Veracruz computations are WebAssembly binaries that use the WebAssembly System Interface (WASI). WebAssembly acts both as a sandbox, pinning down the behaviour of the program, and allows us to abstract over the different strong isolation technologies. For the design methodology and use cases of Veracruz, an interested reader is referred to [5].

B. AWS Nitro System

The AWS Nitro system [7] is a new Amazon Elastic Compute Cloud (EC2) feature that allows users to create isolated compute environments (a.k.a, enclaves) to protect and process sensitive data within EC2 instances. Attestation is an important feature of Nitro enclaves which allows a remote host to verify an enclave's identity and that only authorized code is running inside the enclave. The attestation process is conducted via the Nitro Hypervisor, which uses a series of measurements that are unique to an enclave to produce a signed attestation document. An attestation document contains key information (e.g., enclave's public key, hashes of the enclave images and applications, etc.) of an enclave and allows the remote host to determine whether to grant the enclave access to the requested operation. For more information about the AWS Nitro system, an interested reader is referred to [8].

III. System Architecture and Workflow

A. System Architecture

The system architecture of privacy-preserving object detection for Ucam is illustrated in Fig. 1. While the current implementation is based on AWS solutions, the proposed architecture can be easily adapted to other cloud platforms. The proposed system consists of the following roles and components:

- *Ucam*: A Ucam is a secure home IP camera which encrypts captured video clips using a user-controlled symmetric key.
- *AWS S3*: S3 is an object storage service in AWS for storing encrypted video clips.
- *User Application*: A user application deals with generating user policy and creating credentials for the user proxy accessing the AWS S3 and Veracruz instance.
- *User Proxy*: The user proxy is a service that retrieves encrypted video clips from AWS S3 and sends them to the Veracruz instance on behalf of a Ucam user.

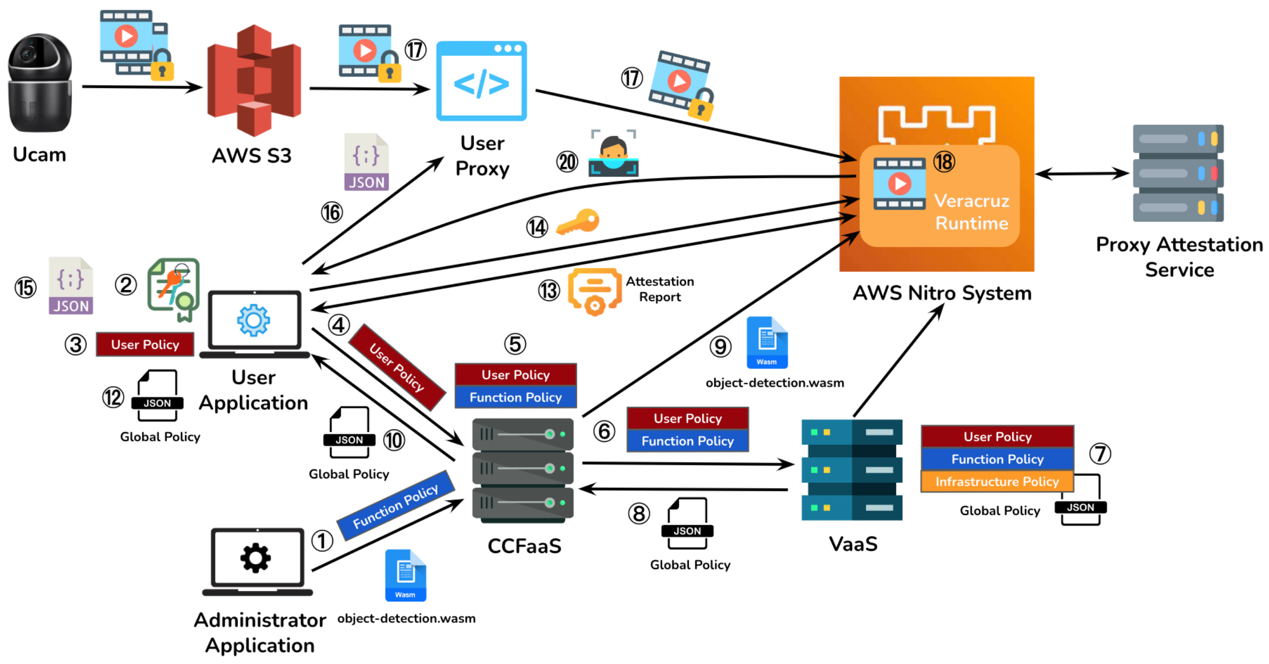

Fig. 1. The system architecture of privacy-preserving object detection for Ucam using Veracruz

- *Administrator Application*: An administrator application is responsible for registering confidential computing functions (e.g., an object detection algorithm) with CCFaaS.
- *Confidential Computing as a Service (CCFaaS)*: The CCFaaS provides RESTful APIs that allow confidential computing functions to be instantiated within a Veracruz environment.
- *Veracruz as a Service (VaaS)*: The VaaS provides RESTful APIs that allow multiple instances of Veracruz to be allocated, attested and loaded in a Kubernetes-based, Nitro-enabled cloud.
- *AWS Nitro System*: The AWS Nitro enclaves create trusted execution environments (TEEs) that allow users to perform computations on sensitive data.
- *Proxy Attestation Service (PAS)*: The PAS provides a unified interface that allows a single client to attest a Veracruz runtime on different TEE platforms without having to know the details of the various attestation flows.

To realize privacy-preserving object detection with Veracruz, a key step is to create a public global policy configuration file that parameterizes the computation in question and contains other relevant metadata.

B. Veracruz Instantiation

A Veracruz instance is dedicated to compute the result $\pi(D_1, \ldots, D_n)$ in a delegated third-party machine. Internally, Veracruz utilises a virtual in-memory filesystem (VFS) and both programs, π, and data, D_i, are uniformly treated as files. During instantiation, Veracruz requires a *global policy* (described in JSON), auditable by all users, to provide essential information: the *topology* of the computation, and attestation and infrastructure *metadata*. The former grants different persmissions to different entities, both users and programs. In particular, it is crutial to protect sensitive information amongst mutually distructed users. In Veracruz, a user is identified by a X.509 certificate, which is used in the TLS connection. After authentication, a user can provision (write to VFS) programs and data, and retrieve (read from VFS) the result of the computation from the VFS in Veracruz. All access to the VFS is bounded by the permission specifies in the policy. Furthermore, if a user provisions a program, it must match the expected hash explicitly included in the policy. Each program is also granted with certain permissions to VFS; this implicitly specifies the inputs and outputs, and necessary intermediate buffers (files) for the execution.

The metadata of a Veracruz contains necessary information for managing and communicating with the instance, and, more importantly, attestation. The former contains experation date and URL of this instance. The latter congains expected hashes of Veracruz runtime and the underline TEE, e.g. Nitro. Since Veracruz abstracts attestation process by PAS, its URL is also included in the policy.

C. System Workflow

In this section, we will refer the step numbers in Fig. 1. We now introduce the three phases in our system, *policy creation*, steps 1 to 11, *attestation*, steps 12, 13 and 14, and *privacy-preserving object detection*, step 15 to 21.

1) Global Policy Creation: In this phase, the user application, administrator application, CCFaaS, and VaaS work together to create a global policy for realizing the privacy-preserving object detection using Veracruz. The global policy creation process is as follows:

① The administrator application creates a function policy file and registers the object detection function that is compiled to WebAssembly (i.e., object-detection.wasm) with CCFaaS;
② The user application generates a key pair and the corresponding X.509 certificate for the user proxy;
③ The user application creates a user policy file which specifies: i) object-detection.wasm is the function to be executed in TEE; ii) The user proxy provides the input (i.e., encrypted video clips); and iii) The user application receive the output (i.e., the object detection result);
④ The user application executes a remote procedure call (RPC) to the CCFaaS with the created user policy;
⑤ The CCFaaS validates the request and creates a valid policy request that is the combination of the user and function policies;
⑥ The CCFaaS executes a RPC to the VaaS with the combined user and function policy and requests the VaaS to provision a Veracruz instance running that policy;
⑦ The VaaS validates the request and starts a Veracruz instance in the allocated AWS EC2 Nitro system with a global policy that is the combination of the user, function and infrastructure policies;
⑧ The VaaS returns the global policy to the CCFaaS;
⑨ The CCFaaS load the object detection code into the provisioned Veracruz instance;
⑩ The CCFaaS returns the global policy to the user application after the program code is loaded successfully;
⑪ The CCFaaS and VaaS waits for the next confidential computation request.

2) Remote Attestation and Key Provisioning: In this phase, the user application performs a remote attestation with the aid of the PAS to validate integrity of the program code and data inside the AWS Nitro enclave and provisions the video decryption key in the enclave after verifying the attestation report successfully. The remote attestation and key provisioning process is as follows:

⑫ The user application receives the global policy from the CCFaaS;
⑬ The user application directly connects to the Veracruz instance using the connection information in the global policy and verifies that the SHA-256 hash of the enclave runtime matches the result in the attestation report;
⑭ The user application creates a file in the Veracruz instance with the video decryption key.

3) Privacy-Preserving Object Detection: In this phase, the user application instructs the user proxy to retrieve an encrypted video clip from AWS S3 and sends it to the Veracruz instance for object detection. The encrypted video clip is decrypted inside the enclave before applying the object detection algorithm. Once the computation is completed, the result can be securely retrieved by user application via TLS. The object detection process is as follows:

⑮ The user application creates a JSON object which contains: i) The name of the AWS S3 bucket; ii) The file name of the encrypted video clip: iii) The credentials for accessing the file; iv) The endpoint of the Veracruz instance; and v) The previously generated key and X.509 certificate for the user proxy;
⑯ The user application executes a RPC to the user proxy with the created JSON object;
⑰ The user proxy starts an instance of itself that retrieves the encrypted video clip from the AWS S3 bucket and sends it to the Veracruz instance;
⑱ The Veracruz instance decrypts the video clips using the previously provisioned decryption key, performs the object detection algorithm and sends the results to the output file;
⑲ The user proxy instance closes the connection to the Veracruz instance at the end of file transmission and terminates itself;
⑳ The user application retrieves the results and terminates the Veracruz instance;
㉑ The AWS EC2 Nitro instance is returned to the pool of free instances.

The full orchestrated end-to-end demo is deployed using docker containers on a Kubernetes (k8s or k3s) infrastructure on AWS. The source code and deployment documentation are available on Github[1].

REFERENCES

[1] Ucam. https://ucam.iotex.io/.
[2] IoTeX. https://iotex.io/.
[3] X. Fan, Z. Zhong, Q. Chai and D. Guo, Ucam: A User-Centric, Blockchain-Based and End-to-End Secure Home IP Camera System, *The 16th EAI International Conference on Security and Privacy in Communication Networks (SecureComm 2020)*, ser. LNICST 336, N. Park et al. (Eds.), Berlin, Germany: Springer-Verlag, pp. 311-323, 2020
[4] Veracruz: Privacy-Preserving Collaborative Compute. https://veracruz-project.com/.
[5] M. Brossard, G. Bryant, B. El Gaabouri, X. Fan, A. Ferreira, E. Grimley-Evans, C. Haster, E. Johnson, D. Miller, F. Mo, D.P. Mulligan, N. Spinale, E. Van Hensbergen, H.J.M. Vincent, and S. Xiong, Private Delegated Computations Using Strong Isolation, https://arxiv.org/abs/2205.03322, 2022.
[6] Kubernetes. https://kubernetes.io/.
[7] AWS Nitro System. https://aws.amazon.com/ec2/nitro/.
[8] AWS Nitro Enclaves User Guide. https://docs.aws.amazon.com/enclaves/latest/user/enclaves-user.pdf.

[1]https://github.com/veracruz-project/veracruz-examples/tree/main/i-poc