VC3:
Trustworthy Data Analytics in the Cloud Using Intel SGX

Mohammad H. Mofrad & Spencer L. Gray
University of Pittsburgh
Thursday, December 1, 2016

hasanzadeh@cs.pitt.edu & slg94@cs.pitt.edu
Preface

• Introduction
  • Article information
  • Cloud security challenges
  • Intel SGX
  • MapReduce framework

• VC3 (Verifiable Confidential Cloud Computing)
  • Threat model
  • Design overview
  • Region integrity
  • Job deployment
  • Job execution and verification
  • Results

Title: VC3: trustworthy data analytics in the cloud using SGX
Conference: 36th IEEE Symposium on Security & Privacy
h5-index: 59
Authors: F. Schuster et al
Affiliation: Microsoft Research
Partnership: Intel Corporation
Citation: 56
Year: 2015

Cloud Computing Security Challenges

• Information leakage in the cloud site
  • Malicious cloud
  • Malicious admin user
  • Attacks
  • Government

• Cloud users seek the followings:
  • Confidentiality + Integrity of their code & data
  • Verifiability of execution of the code over data

Threat Model

• Assumes a strong, but not all powerful adversary

• Adversary can:
  • Control the software stack (the hypervisor and the OS)
  • Physically access the hardware of an SGX processor
  • Read/modify any data leaving a processor
  • Read/modify/replay any network packets
  • Access any job running on the cloud simultaneously
Threat Model

• There are some limitations

• Adversary can’t
  • Access the hardware of any machine at the data center
  • Denial of Service attack
  • Analyze network traffic
  • Side channel attacks
  • Fault injections

• Users are assumed to write code reasonably well
  • Assuming users can only have low level defects
  • Users will not intentionally leak information
Intel Software Guard Extension (SGX)

- **ENCLAVE?**
- Hardware-based protection
- User level execution
- Small TCB

Reduced attack surface with Intel SGX

Design Overview

- $E^-$ = user written map and reduce functions compiled and encrypted
  - Data is always encrypted when on cloud
  - Is only in plaintext when it is being run on a trusted processor chip

- $E^+$ = public code for key exchange and execution protocols
  - The operating system is not a part of $E^+$ to reduce trusted code base
  - The enclave code is designed to not need the OS

- $F$ = untrusted code for worker nodes

---

Design Overview

• Enclaves contain their own stack and heap, but share memory with F
  • This allows E+ code to interact outside of the enclave
• F helps run the enclave, but **region self-integrity** prevents it from corrupting enclave data
MapReduce Framework

Motivation Behind Region Self-Integrity

• Enclave code shares virtual address space with untrusted code
  • Allows efficient communication outside of the enclave
  • Broadens attack surface
• Enclaves could dereference a corrupted pointer from the shared memory region
• Enclave could leak data in this way

Region Read-Write-Integrity

• Region Write-Integrity
  • Guarantees that any write that uses a pointer only writes to variables whose addresses are used in the code
  • Example: Using a pointer to write to &var_name or using an array
  • Prevents memory corruption and memory leaks

• Region Read-Write-Integrity
  • Same write guarantee as above
  • Reading a pointer only reads an address inside the enclave
  • Prevents data from being injected into the enclave
Enforcing the Integrity

- Both Region-Read and Region-Read-Write Integrities are enforced by a special compiler
- Code in the enclaves is written by VC3 and the user
  - We compile the code in E^- and E^+, not the cloud
- Compiler inserts code for dynamic evaluations of these integrities
  - Can’t always determine a violation statically
- When a violation is found, program is stopped (safer)
  - Could have also just masked the bits in the address
- Tradeoff between security and efficiency
Cryptographic Assumptions

- $m \ | \ n$: Concatenation of two messages $m$ & $n$
- $\text{PRF}_k(\text{text})$: pseudo-random function
- $H(\text{text})$: Collision-resistant cryptographic hash (HMAC + SHA-256)
- $\text{EDigest}(C)$: Digest of an enclave’s initial content $C$
- $\text{PKGen}$: Creates public key $p_k$, $s_k$
- $\text{PKEnc}_{p_k}(\text{text})$: Encrypts text under $p_k$
- $\text{Esig}_p(c)$: The identity $p$ jointly signs $H(\text{text})$ and $\text{EDigest}(C)$
- $\text{Enc}_k(\text{text}, \text{data})$: Encryption of text with associated data
- $\text{Dec}_k(\text{cipher}, \text{data})$: Decryption of cipher with associated data

Job Deployment

- Environment
  - Working on a unmodified version of Hadoop
  - Small TCB by Keeping Hadoop, OS, and hypervisor out of the TCB

**Trusted part**

1. Map() & reduce() functions
2. Encryption them
3. Bind them with some code that enables some cryptographic operations
4. Upload them on the Cloud

**Untrusted part**

1. In a *Worker Node*, the Cloud OS loads the code in an Enclave
2. The hosted code in the enclave will run a key exchange protocol
3. The Hadoop will run the code in the distributed manner

Job Deployment - Cloud attestation

• Processors
  • **Quote Enclave** for inter-platform enclave attestation
  • A symmetric key that verifies the enclave authenticity (genuine SGX processor)

• Machines
  • **Cloud QE** is a pair of public/private key for each
  • Public key + sealed private key in the Cloud QE

• Create an identity from **Processor + Machine**
  • $E_{\text{sig}}_{\text{SGX}}(C, \text{text})$ | $E_{\text{sig}}_{\text{cloud}}(C, \text{text})$
Job Deployment – Key Exchange

• Existing Hadoop’s communication channel

• VC3’s MapReduce Job (e.g. a MapTask):

• Adding an in-band variant of key exchange

• Lightweight key exchange job before the actual job
  1. Run key exchange job ($C_{j,u}$)
  2. Create job credential ($JC_w$)
  3. Run actual job ($J_w$)

Job Deployment – Setting up a MapReduce Job

1. User authentication $pk_u$

2. Invoking nodes to compute the job enclave ($C_{j,u}$)
   \[ C_{j,u} = E^+ \mid Enc_{k_{code}}(E^+) \mid j \mid pk_u \]

3. Each node $w$ starts working on and creates the $m_w$ message
   \[ m_w = PKEnc_{pky}(k_w) \]
   Then requests quotes from the SGX and Cloud QE
   \[ p_w = m_w \mid ESig_{SGX\ Cloud}(C_{j,u})(m_w) \]

4. The task process verifies the $p_w$ and signs the code identity $C_{j,u}$ and creates
   Job credential $JC_w$
   \[ JC_w = Enc_{kw}(K_{code}, k) \text{ where } k = k_{job} \mid k_{in} \mid k_{inter} \mid k_{out} \mid k_{prf} \]

5. Each node resumes $E^+$ and decrypts the $JC_w$
Job Execution and Verification - Set Up

• Each input split is bound to a fresh unique ID $L_{in}$
• $R =$ Number of logical reducers for job
• The user encrypts each input split:

$$\text{Input}' = \text{Enc}_{k_{in}}(L_{in})\{\text{Input}\}$$

User chooses $B_{in}$ as a subset of input splits

Job $| k_{job} | R | B_{in}$
Job Execution and Verification - Mapping

• Mapper in the cloud receive splits
  • Keep track of ID’s to prevent replay attacks

• Mappers produce intermediate key-value pairs
  • Key-value pairs with identical keys must be processed by same reducer

• After processing inputs, mappers create a closing intermediate key value pair
  • Allows a reducer to exit if it receives duplicate key-value pairs or not enough key-value pairs
  • Detects if cloud services dropping jobs or attempting replay attack

Job Execution and Verification - Verification

- Verifier receives map messages and reducer messages
- Verifies integrity of job by checking first to see if it received correct amount of each
- Verifier prevents replay attacks between different jobs
  - Each message contains a job specific ID
- Performance Cost for verification is very small

Evaluation

• Deploy Hadoop on a cluster of 8 machines
  • Windows server 2016 R2 64-bit
  • CPU: 2.9 GHz Core i5 Haswell family
  • HDD: 250GB SSD capability
  • Benchmarks: 7 I/O and computation intensive programs

<table>
<thead>
<tr>
<th>Application</th>
<th>LLOC</th>
<th>Input size</th>
<th>E- size</th>
<th># map tasks</th>
</tr>
</thead>
<tbody>
<tr>
<td>User usage</td>
<td>224</td>
<td>41 GB</td>
<td>18 KB</td>
<td>665</td>
</tr>
<tr>
<td>I/O volumes</td>
<td>241</td>
<td>94 GB</td>
<td>16 KB</td>
<td>1530</td>
</tr>
<tr>
<td>Options</td>
<td>6098</td>
<td>1.4 MB</td>
<td>42 KB</td>
<td>96</td>
</tr>
<tr>
<td>Word count</td>
<td>103</td>
<td>10 GB</td>
<td>18 KB</td>
<td>162</td>
</tr>
<tr>
<td>Pi</td>
<td>88</td>
<td>8.8 MB</td>
<td>15 KB</td>
<td>16</td>
</tr>
<tr>
<td>Revenue</td>
<td>96</td>
<td>70 GB</td>
<td>16 KB</td>
<td>256</td>
</tr>
<tr>
<td>Key search</td>
<td>125</td>
<td>1.4 MB</td>
<td>12 KB</td>
<td>96</td>
</tr>
</tbody>
</table>

Experiments

- Execution time of running MapReduce jobs in the Hadoop cluster
  - Baseline Hadoop
  - VC3 with encrypted mapper & reducer inputs & outputs (1% avg. overhead)
  - W (region write integrity) 4.5%
  - WR (region read-write integrity) 8%

Experiments

- Execution time of running the map phase of MapReduce jobs in isolation
- Eliminating most of...
  - Hadoop internal I/Os
  - Enclave operations
  - Cryptographic Ops.

Discussion

• **Good design decision:**
  - Keeping TCB small
  - IND-CPA assumption

• **VC3 threat model assumptions:**
  - hardware attacks, e.g. power analysis
  - Side-channel
  - Replay attack

• **Encryption overhead**
  - cryptographic operations
  - Copying data to and from enclave
  - Enclave creation & transitions
  - Trade off between performance and security

• **Simulating SGX capabilities at that time:**
  - Intel SGX emulator
  - Microsoft C++ compiler

Discussion

• Combining clear hardware security with the cloud brings **confidentiality** and **integrity** to the Cloud but ...
  
  • **Information leakage**: Learning the intermediate <key, value> distribution
    
    MapReduce job $\rightarrow$ (A set of map tasks) $\rightarrow$ (A set of reduce tasks)
    
    **FIX**: Padding, clustering, and shuffling

• **Replay attacks**: Having $C_{j,u}$ and $C_{J,w}$ and adversary can launch arbitrarily reply parts of a job that the processor participated before
  
  This amplify other side-channel attacks against confidentiality
  
  **FIX**: hardcoding the job’s specification into mapper

---

Discussion

• **VC3** Brings data **confidentiality** and **integrity** using cryptographic operations
  
  • **Advantage**: Even if an adversary or a privileged Cloud user access data, they will just see encrypted data.
  
  • **Advantage**: The Cloud provider cannot fake the results and the customer can **verify** the results of computation

• Working on an unmodified version of **Hadoop** make it easy for research community to easily attach their framework to the existing Hadoop-based products.

---
