Securing Time in Untrusted Operating Systems with TimeSeal

Fatima M. Anwar, Luis Garcia, Xi Han, Mani Srivastava
Vulnerabilities in Time

**Causes**
- Malicious *delays*
- Inconsistent *rates*
- Induced *drifts*
- Non monotonic *timers*

**Consequence**
- Time discontinuities, Fuzzy Time
Attack Strategies

No Attack
Attack Strategies (1)

Offset Manipulation
Attack Strategies (2)

Effect on a Single Time Source:

Rate Manipulation
Attack Strategies (2)

Relative Effect on Two Time Sources:

Decrease Time Advance Rate

Increase Time Advance Rate

Rate Manipulation

error = 0  error = 2  error = 4
Time Attack Consequences

Forge Timestamp

Location theft

Grid Attack

Violate DRM
Do Trusted Platforms provide Secure Time?
Securing Time Challenges

1. A Trusted Timer
2. Secure Access to the Trusted Timer
3. Secure Timekeeping Software
4. Secure Access to Global Time on the Network
SGX “Trusted” Time

1. Trusted Timer
   SGX trusted time

Platform Service Enclave

Operating System
SGX “Trusted” Time

1. Trusted Timer

Platform Service Enclave

SGX Measured Time (sec) vs. Actual Time (sec)

- True SGX tick
- 1 second resolution
Attack on SGX “Trusted” Time

1. Trusted Timer
   
   Platform Service Enclave
   
   2. NO Secure Access to Local Time

   Operating System
   
   Application Enclave
   
   Timekeeping software

   Delay Attack
Attack on SGX “Trusted” Time

1. Trusted Timer
   - SGX trusted time

2. NO Secure Access to Local Time

3. NO Secure Timekeeping Scheduling Attack

Platform Service Enclave
Application Enclave

Operating System

Timekeeping software
Attack on SGX “Trusted” Time

Result:
Attack on SGX “Trusted” Time

Result:

![Graph showing time dilation and compression](image)

The effect of delay attacks can also be visualized in two ways: fuzzy SGX time and time dilation & compression. The graph illustrates how SGX time measures 1sec durations on y-axis, with actual time period (sec) on x-axis. Time advances without a fundamental fixed frequency, resulting in either time dilation or compression across different intervals. Thus, delay attacks cause the timekeeping service maintained by an enclave process to dilate and constrict, and we establish that adjusting a thread's frequency, resulting in either time dilation or compression, is not secure.

We implement a delay attack in the OS by delaying an enclave timekeeping thread, which in reality are dilated and compressed durations. However, an enclave polling SGX time would not miss any SGX tick, and the attacker is still able to schedule out all threads of a process in order to maintain stealthiness—since they know the exact time whenever the process wakes up. Time is not a dependable indicator in any manner.

It is to be noted that an attack strategy of delaying time to not miss any SGX tick, and the attacker is still able to schedule out all threads of a process in order to maintain stealthiness, is worse for the system as a single time jump can easily be detected. Prior knowledge of the system and physical clock characteristics helps the attacker launch an attack that degrades system performance without detection. An attacker may stay undetected throughout the damage, an attacker may stay undetected throughout the damage, an attacker may stay undetected throughout the damage, an attacker may stay undetected throughout the damage, an attacker may stay undetected throughout the damage, an attacker may stay undetected throughout the damage, an attacker may stay undetected throughout the damage, an attacker may stay undetected throughout the damage, an attacker may stay undetected throughout the damage, an attacker may stay undetected throughout the damage, an attacker may stay undetected throughout the damage.

We assume that an attacker does not trust the OS, as well as delay attacks on SGX time packets. The attacker also knows that packets by intercepting all transmitted/received packets. Therefore, it launches attacks on SGX “Trusted” Time.

We list all the challenges, we are in a position to state our threat model. Our threat model considers TEE as well as delay attacks on SGX time packets. The attacker is also capable of launching scheduling attacks on SGX enclave threads as established in EAL 5. We assume that an attacker does not trust the OS but to maintain stealthiness, it would choose to delay SGX time increments every sec, referred to as SGX tick. The attacker also knows that SGX notion of 1sec fluctuates within a uniform distribution of 0 to 1sec. The result in our threat model incorporates incremental, random, and distribution based delay attacks on SGX packets. The attacker that degrades system performance without detection is worse for the system as a single time jump can easily be detected. Prior knowledge of the system and physical clock characteristics helps the attacker launch an attack that degrades system performance without detection. An attacker may stay undetected throughout the damage, an attacker may stay undetected throughout the damage, an attacker may stay undetected throughout the damage, an attacker may stay undetected throughout the damage, an attacker may stay undetected throughout the damage, an attacker may stay undetected throughout the damage, an attacker may stay undetected throughout the damage, an attacker may stay undetected throughout the damage, an attacker may stay undetected throughout the damage, an attacker may stay undetected throughout the damage.
After listing all the challenges, we are in a position to state our threat model. Our threat model considers TEE as well as delay attacks on SGX time packets.

We implement a delay attack in the OS by delaying processing of SGX time packets. The attacker is also capable of launching scheduling attacks on SGX enclave threads. The attacker also knows that 'aesmd daemon' encapsulates PSE and handles SGX trusted time packets. Therefore, it launches attacks on SGX "Trusted" Time.

We assume that an attacker does not trust the OS and waits over a period of time before launching the attack. For example, the OS knows that 'aesmd daemon' transmits SGX packets by intercepting all transmitted/received packets. The attacker also knows that 'aesmd daemon' transmits SGX packets by a different value adds variations to the SGX packets.

The attacker also launches attacks on SGX "Trusted" Time. The attacker is capable of making SGX time fuzzy as established in EAL 5. It is to be noted that an attack strategy of delaying SGX time by a sec or so. This causes an application polling SGX time to/from aesmd daemon. The attacker also knows that SGX notion of 1sec fluctuates within a uniform distribution of 0 to 1sec. The result in another way shown in Figure 3b. An application relying on SGX time measures 1sec durations on y-axis, while 4sec varies between 2.2 to 5.5 sec.

The attacker is also capable of launching attacks on SGX "Trusted" Time. The attacker can accumulate more variation in delay value, the more error an adversary can accumulate.

We refer to attacks on timekeeping threads as well as delay attacks on SGX time packets. The attacker that degrades system performance without detection is worse for the system as a single time jump can easily damage, an attacker may stay undetected throughout the system's operation because consistent time uncertainty distorts their sense of elapsed time resulting in wastage of computation resources and no timely responses most important in online business services which becomes an availability issue rather than a security issue.

We strive to protect against attackers capable of making SGX time fuzzy as established in EAL 5. TEE strives to protect against attackers capable of launching delay attacks on SGX "Trusted" Time. TEE is a secure time architecture that overcomes on x-axis. Time advances without a fundamental fixed clock rate and distorts the passage of time. Therefore, delaying SGX packets by a different value adds variations to the measured period (sec) without any bounds delays SGX time packets for long enough to maintain stealthiness–if there are no threads running, the attacker that does not fear detection and manipulates time is considered a detectable denial of service. An attacker that degrades system performance without detection is worse for the system as a single time jump can easily damage, an attacker may stay undetected throughout the system's operation because consistent time uncertainty distorts their sense of elapsed time resulting in wastage of computation resources and no timely responses most important in online business services which becomes an availability issue rather than a security issue.

The effect of delay attacks can also be visualized in Figure 3: Two ways to visualize the effect of delay attacks on SGXtrusted time. Figure 3a shows how delay attacks affect SGX time. For example, SGX notion of 1sec fluctuates within a uniform distribution of 0 to 1sec. The result in another way shown in Figure 3b. An application relying on SGX time measures 1sec durations on y-axis, while 4sec varies between 2.2 to 5.5 sec. Attack on SGX "Trusted" Time

Result:

![Diagram](https://via.placeholder.com/150)
Attack on SGX “Trusted” Time

1. Trusted Timer
   SGX trusted time

2. NO Secure Access to Local Time
   Delay Attack

3. NO Secure Timekeeping
   Scheduling Attack

Platform Service Enclave

Application Enclave

Operating System

Timekeeping software
Attack on Network Packets

1. Trusted Timer
   - SGX
   - trusted time

2. NO Secure Access to Local Time

3. NO Secure Timekeeping
   - Scheduling Attack

4. NO Secure Access to Global Time
   - Time Transfer Attack

Platform Service Enclave

Application Enclave

Timekeeping software

Network
**TimeSeal: A Secure Time Architecture**

1. Build High Resolution clock
2. Overcome OS Scheduling attacks
3. Compensate for Delay attacks
4. Mitigate Time Transfer attacks

Prototyped TimeSeal on Intel SGX
TimeSeal: A Secure Time Architecture

Operating System

Network

Platform Service Enclave

Application Enclave

SGX trusted time

Timekeeping software
1. High Resolution SGX Clock

Network

Operating System

Application Enclave

TIMESEAL

Timekeeping software

counter 1

counter N

Platform Service Enclave

Platform Service Enclave

SGX trusted time
1. High Resolution SGX Clock

\[ t_{\text{local}} = \text{SGXticks} + \frac{\text{subticks}}{\text{MA(\text{subticks\_per\_second})}} \]
1. High Resolution SGX Clock

- **High Resolution SGX Clock**

![Graph showing measured duration vs. actual duration for high and low resolution clocks](image)

- **High Resolution Clock**
- **Low Resolution Clock**

**Figure 3:**
- (a) High-resolution SGX clock measures sub-sec durations.
- (b) Distribution plot showing measured vs. actual duration.

**Figure 4:**
- (a) Non-monoctonic clock with time discontinuities.
- (b) Monotonic clock with true time.
- (c) Monotonic clock with time discontinuities.

**5.1 Scheduling Attack and Mitigation**

To build a high-resolution SGX clock—which we call *high-res clock*—we use SGX ticks and subticks. To get high resolution, we develop a subtick service that interpolates SGX ticks. This interpolation mechanism allows us to measure fine time intervals.

**Achieved Mean Resolution**

- Achieved mean resolution of high-res clock is 0.1 msec.

**Distribution Plot**

- Measured duration vs. actual duration.
- Mean: 0.096 sec, std: 0.00044 sec.

**5.2 Monotonicity and Error Elimination**

To make the high-res clock monotonic, we revise our clock model by advancing local time from previous time, but also taking into account the accumulated error.

**Monotonic Clock Model**

- Our new clock model not only advances time but also compensates for accumulated error.
- We use a monotonic clock model that not only advances time but also compensates for accumulated error.
1. High Resolution SGX Clock

High Resolution Clock

![Graph showing Measured duration (sec) vs Actual duration (sec) and CDF of resolution (msec)]
1. High Resolution SGX Clock

High System Load

\[ t_{\text{local}} = t_{\text{prev local}} + \frac{\text{subticks}}{\text{MA(subticks per second)}} - \text{slew}(t_{\text{local at SGX tick}} - \text{SGX tick}) \]
TimeSeal Overcomes OS Scheduling Attacks

Network

Platform Service Enclave

Application Enclave

TIMESeal

Timekeeping software

counter 1

counter N

SGX trusted time

Time Transfer Attack

Scheduling Attack

Delay Attack
Overcome OS Scheduling Attacks

**Intuition:**
Randomize the *counting thread order* \((n)\) and *count interval* \((\text{count}_n)\)

\[
P(\text{resolution degradation}) = f(n, N, \text{count}_n)
\]
Secure Timekeeping Software

Mitigate Scheduling Attack

![Graph showing error (msec) vs. Number of threads counting]

- **Attack1 mean**
- **Attack1 95th percentile**
- **Attack2 mean**
- **Attack2 95th percentile**

The graph illustrates the effects of delay attacks and scheduling attacks over time. By adjusting the subtick in delay duration, our delay mitigation technique bounds the error to within tens of milliseconds under 50% system load. In Figure 8, we see a decrease in errors with an increase in number of threads counting threads. Policy B and C are equally resilient to high load scenarios and an attacker can't obfuscate an attack during normal threads. Systems under high load may give over 165msec with 11msec iqr for both errors.

For every application that needs secure time, T_EAL's model of polling PSE for SGX time is similar to clock traces. As shown in Figure 9a, the mean and standard deviation of errors increase with the number of threads resulting in error from 0 to 1sec under 50% system load. In Figure 8, the cumulative errors of delay attacks show that the high resolution TIME's model of polling PSE for SGX time is similar to clock advances with a msec resolution (blue dotted line). The clock traces show the effects of delay attacks and scheduling attacks related parameters discussed in Section 5.2.

Bound delay attacks above 1sec by adjusting the subtick in delay duration, our delay mitigation technique bounds the error to within tens of msec. Table 2 presents the results for single and multiple-thread contexts, where Error (msec) and Number of threads counting are shown for different policies.

- **Policy A**
- **Policy B**
- **Policy C**

Figure 8: Delay attacks of different durations

Figure 9: Delay duration (sec) vs. Error (msec)

- Attack 1 mean
- Attack 1 95th percentile
- Attack 2 mean
- Attack 2 95th percentile

Error is directly proportional to delay attack duration. The more the SGX time packet is delayed, the more error an attacker can accumulate. We test different delay attack intervals ranging from 0 to 1sec under 50% system load. In Figure 8, we see a decrease in errors with an increase in number of threads counting threads. Policy B and C are equally resilient to high load scenarios and an attacker can't obfuscate an attack during normal threads. Systems under high load may give over 165msec with 11msec iqr for both errors.
Secure Timekeeping Software

**Mitigate Scheduling Attack**

<table>
<thead>
<tr>
<th>Attack 1</th>
<th>Attack 2</th>
<th>Attack 3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Single Thread</td>
<td>N-1 Threads</td>
</tr>
<tr>
<td></td>
<td>$\mu$</td>
<td>$\sigma$</td>
</tr>
<tr>
<td>Policy A</td>
<td>0.5</td>
<td>3</td>
</tr>
<tr>
<td>Policy B</td>
<td>0.002</td>
<td>3</td>
</tr>
<tr>
<td>Policy C</td>
<td>8</td>
<td>2.7</td>
</tr>
</tbody>
</table>
TimeSeal Compensates for Delay Attacks

Diagram:
- Network
- Operating System
- Application Enclave
- TIMESeal
- Platform Service Enclave
- SGX trusted time
- Delay Attack
- Time Transfer Attack
- Scheduling Attack
- counter 1
- counter N
- Timekeeping software
Compensate for Delay Attacks

*Intuition:* Leverage domain specific knowledge

\[ t_{\text{local}} = t_{\text{prev\_local}} + \frac{\text{subticks}}{\text{MA}(\text{subticks\_per\_second})} \]
Compensate for Delay Attacks

Intuition: Leverage domain specific knowledge

\[ t_{\text{local}} = t_{\text{prev\_local}} + \frac{\text{subticks}}{\text{MA(\text{subticks\_per\_second})}} \]
Compensate for Delay Attacks

*Intuition:* Leverage domain specific knowledge

Bound *True SGX ticks to compensate for error*

\[ t_{local} = t_{prev \_ local} + \frac{\text{subticks}}{\text{MA}(\text{subticks \_ per \_ second})} \]
Compensate for Delay Attacks

*Intuition:* Leverage domain specific knowledge

Bound *True SGX ticks to compensate for error*

$$t_{local} = t_{prev\_local} + \frac{\text{subticks}}{\text{MA(\text{subticks\_per\_second})}} - \text{slew}(t_{local\_at\_SGXtick} - SGXtick)$$
TimeSeal Evaluation

Mitigate Delay+Scheduling Attacks

![Chart showing measured time, SGX true time, and SGX delayed time]
TimeSeal Evaluation

Mitigate Delay+Scheduling Attacks

![Graph showing error vs. delay attack duration]
**TimeSeal: A Secure Time Architecture**

1. **Trusted Timer**  
   - **SGX trusted time**

2. **Secure Access to Local Time**

3. **Secure Timekeeping**

**Diagram:**
- Network
- Operating System
- Application Enclave
- Platform Service Enclave
- Timekeeping software
  - counter 1
  - counter N

**Attacks:**
- Time Transfer Attack
- Scheduling Attack
- Delay Attack