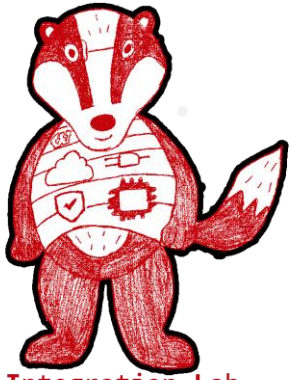
Architecting Trigger-Action Platforms for Security, Performance and Functionality

Deepak Sirone Jegan[*], Michael Swift[*], Earlence Fernandes[+]

* University of Wisconsin - Madison

+ University of California San Diego

Systems, Cloud, and Architecture



Integration Lab

Trigger-Action Platforms

User

Trigger-Action Platforms

Trigger Services (Event Sources)

User

Trigger-Action Platforms

Trigger Services (Event Sources)

Action Services (Event Sinks)

User

Trigger-Action Platforms

Trigger Services (Event Sources)

Event Processing + Forwarding

Action Services (Event Sinks)

User

Trigger-Action Platforms

Trigger Service (Office365)

Action Service (Dropbox)

If email subject contains “IFTTT” then append email body to file “processed.txt”

User

Trigger-Action Platforms

Trigger Service (Office365)

Action Service (Dropbox)

If email subject contains “IFTTT” then append email body to file “processed.txt”

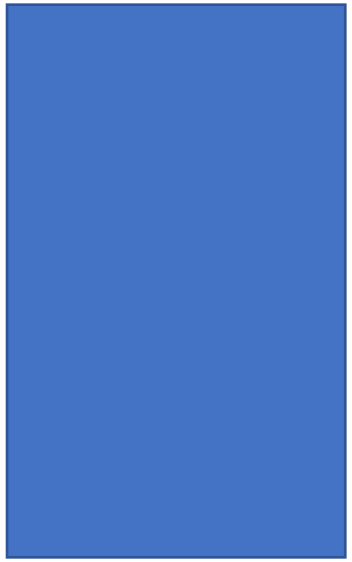
```
var searchResult = Office365Mail.newEmailFrom.Subject.search("IFTTT");

if (searchResult != -1) {
    Dropbox.appendToTextFileDb.append(Office365Mail.newEmailFrom.Body, "processed.txt");
} else {
    Dropbox.appendToTextFileDb.skip();
}
```

User

Trigger-Action Platforms

Trigger Service (Office365)

Trigger Action Platform (TAP)



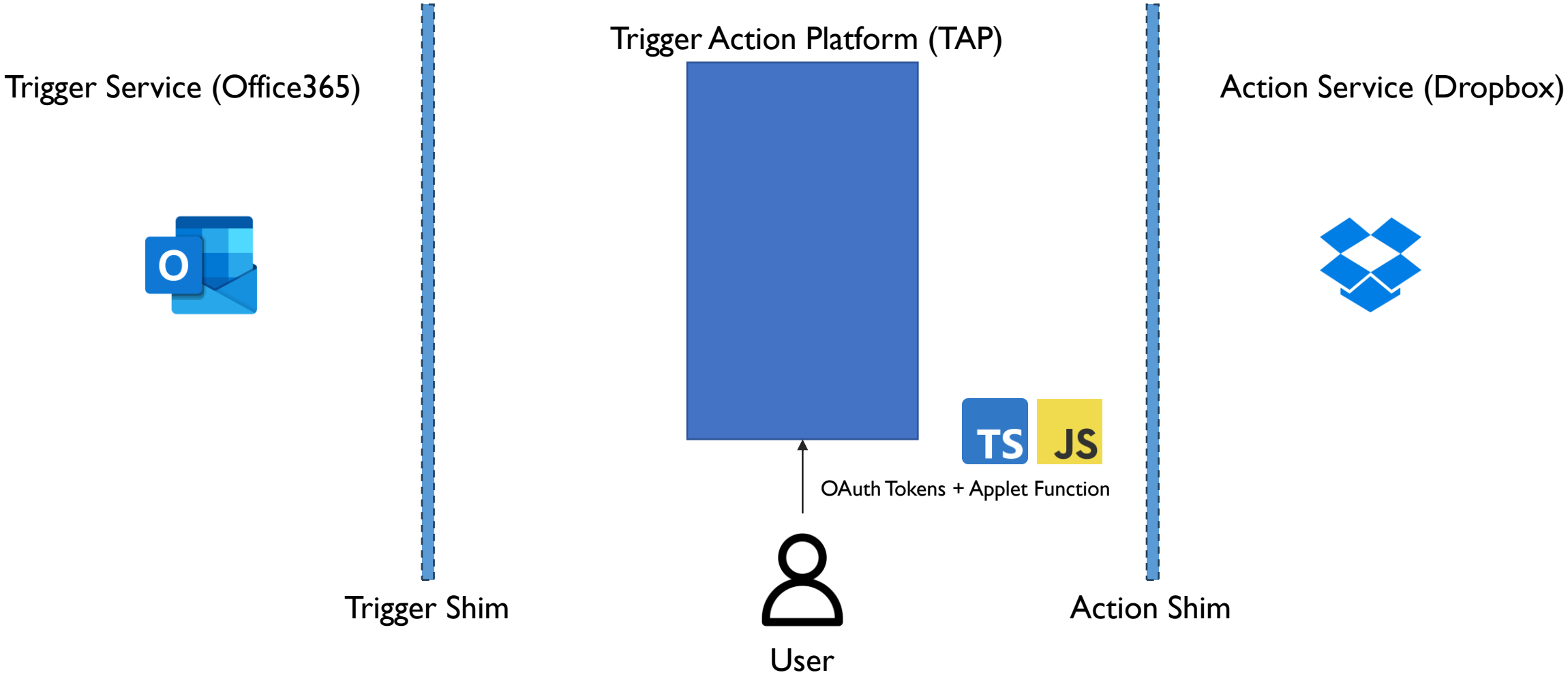
Action Service (Dropbox)

Trigger Shim

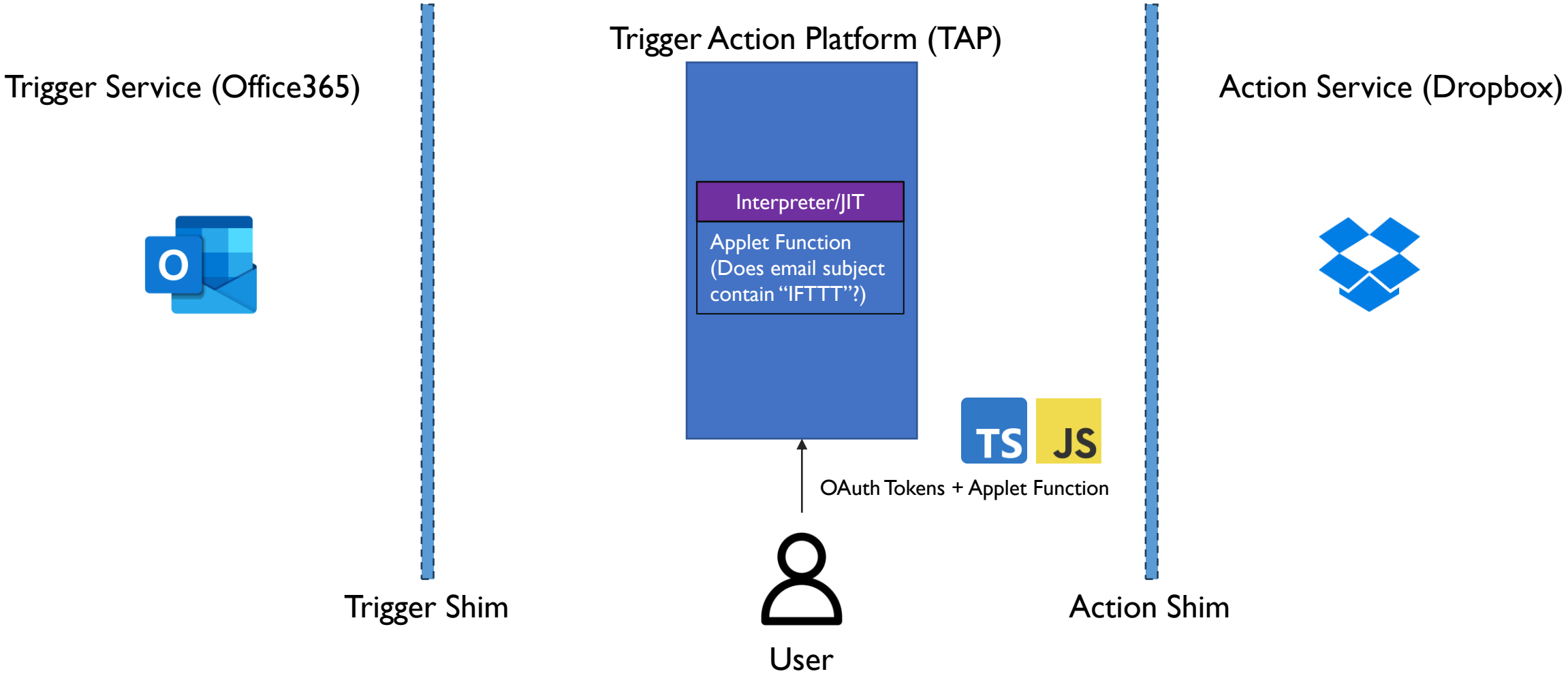
User

Action Shim

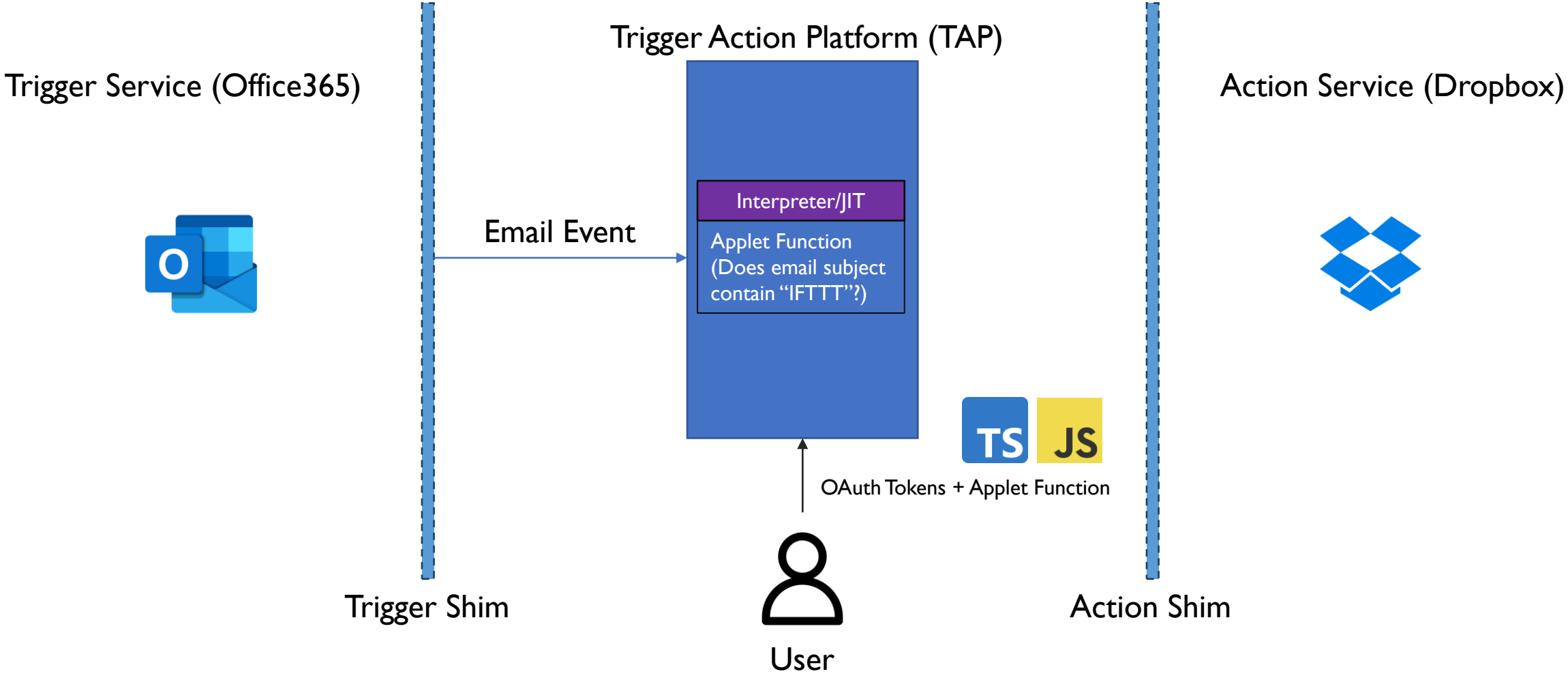
Trigger-Action Platforms



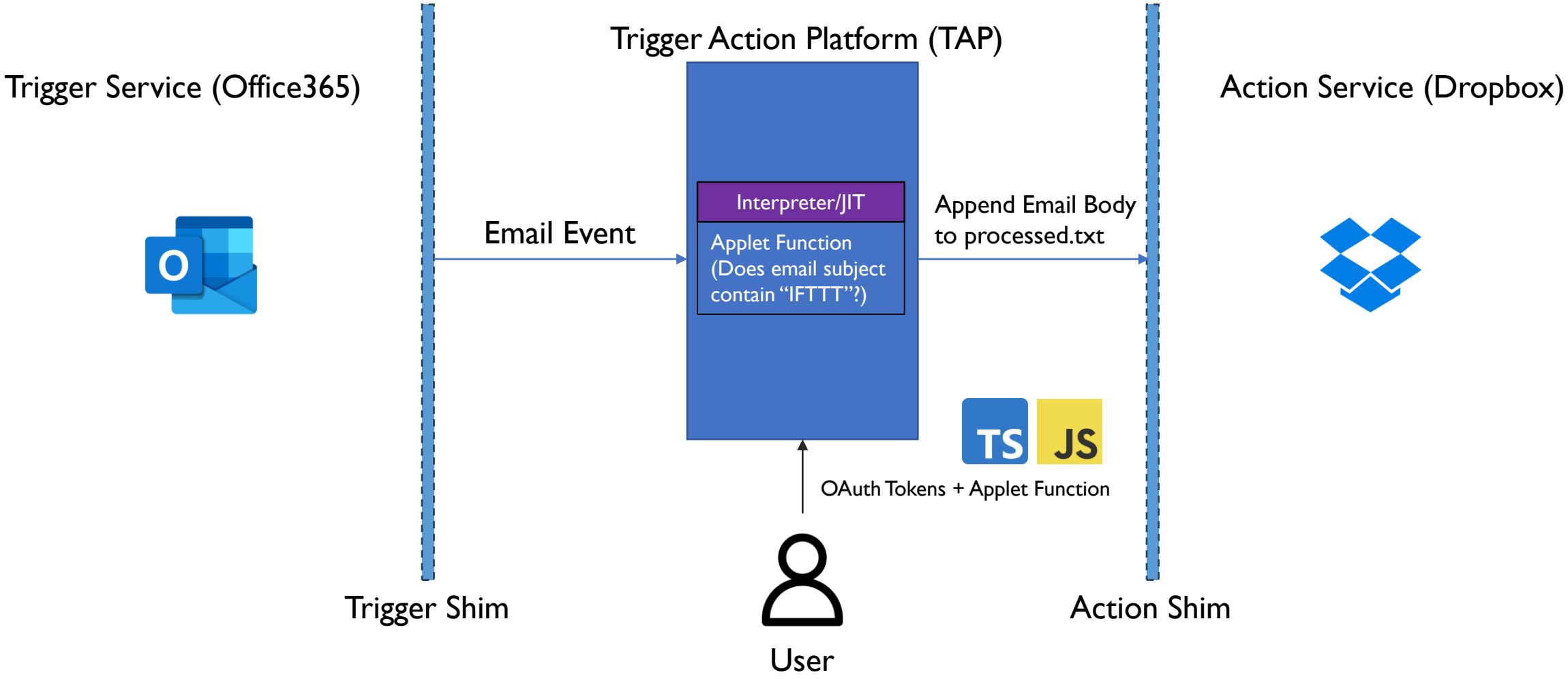
Trigger-Action Platforms



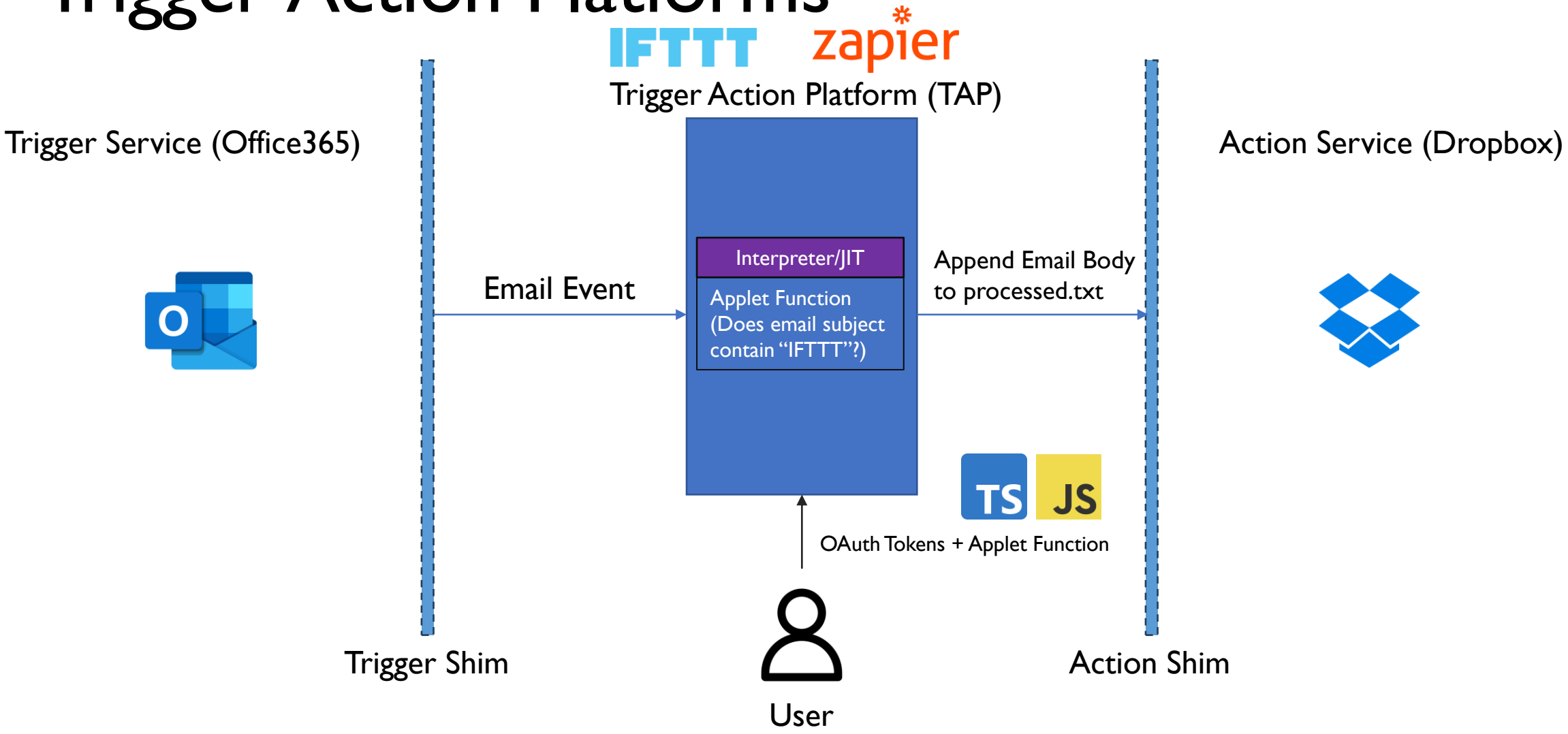
Trigger-Action Platforms



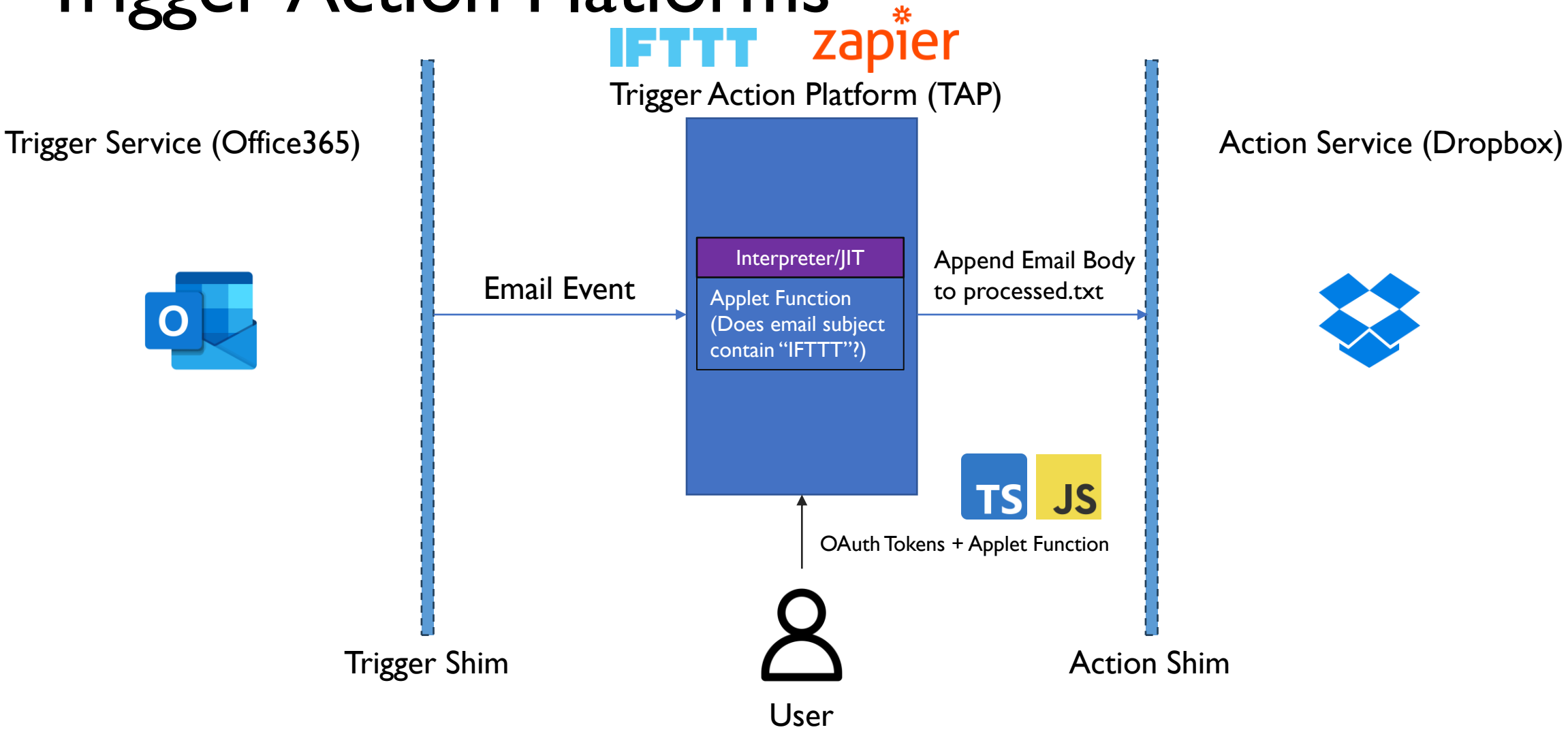
Trigger-Action Platforms



Trigger-Action Platforms

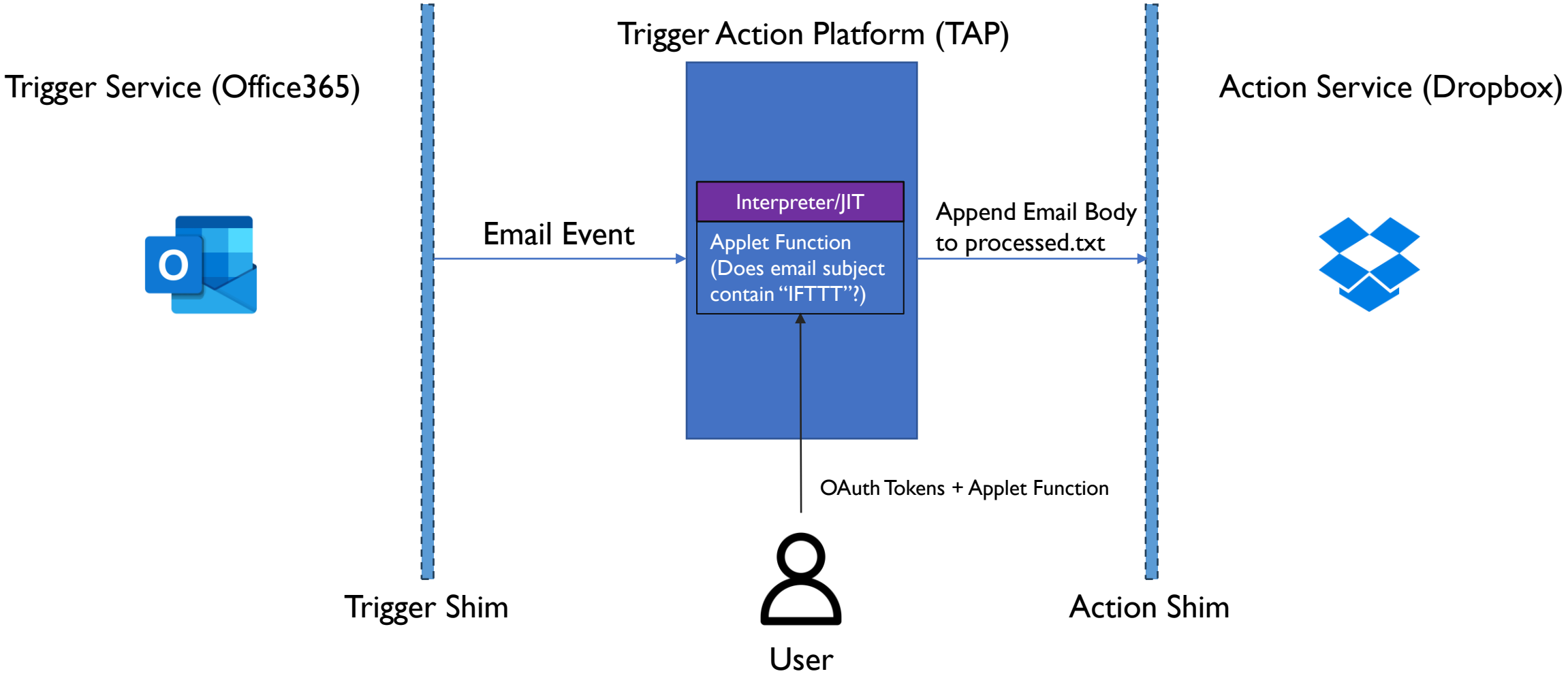


Trigger-Action Platforms

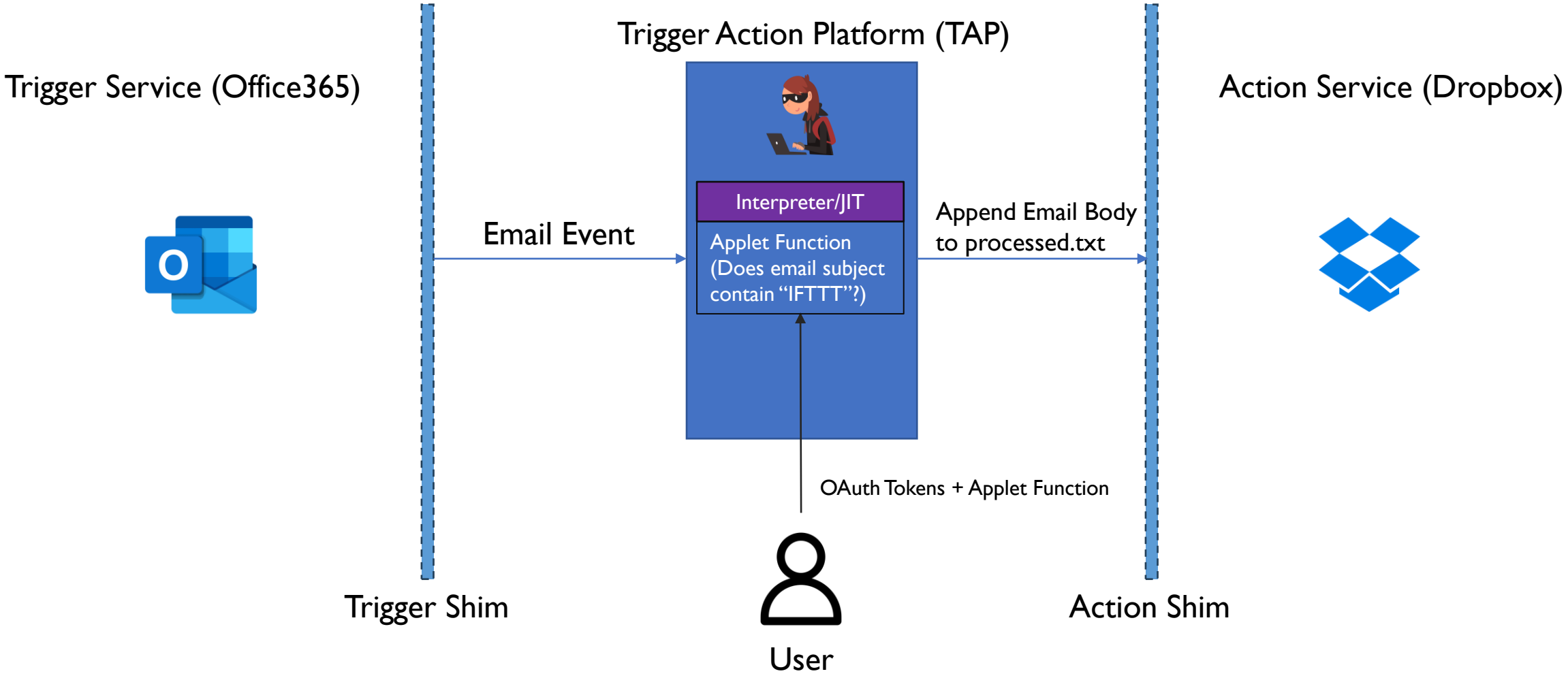


Millions of users must place complete trust in the correct operation of the Trigger Action Platform

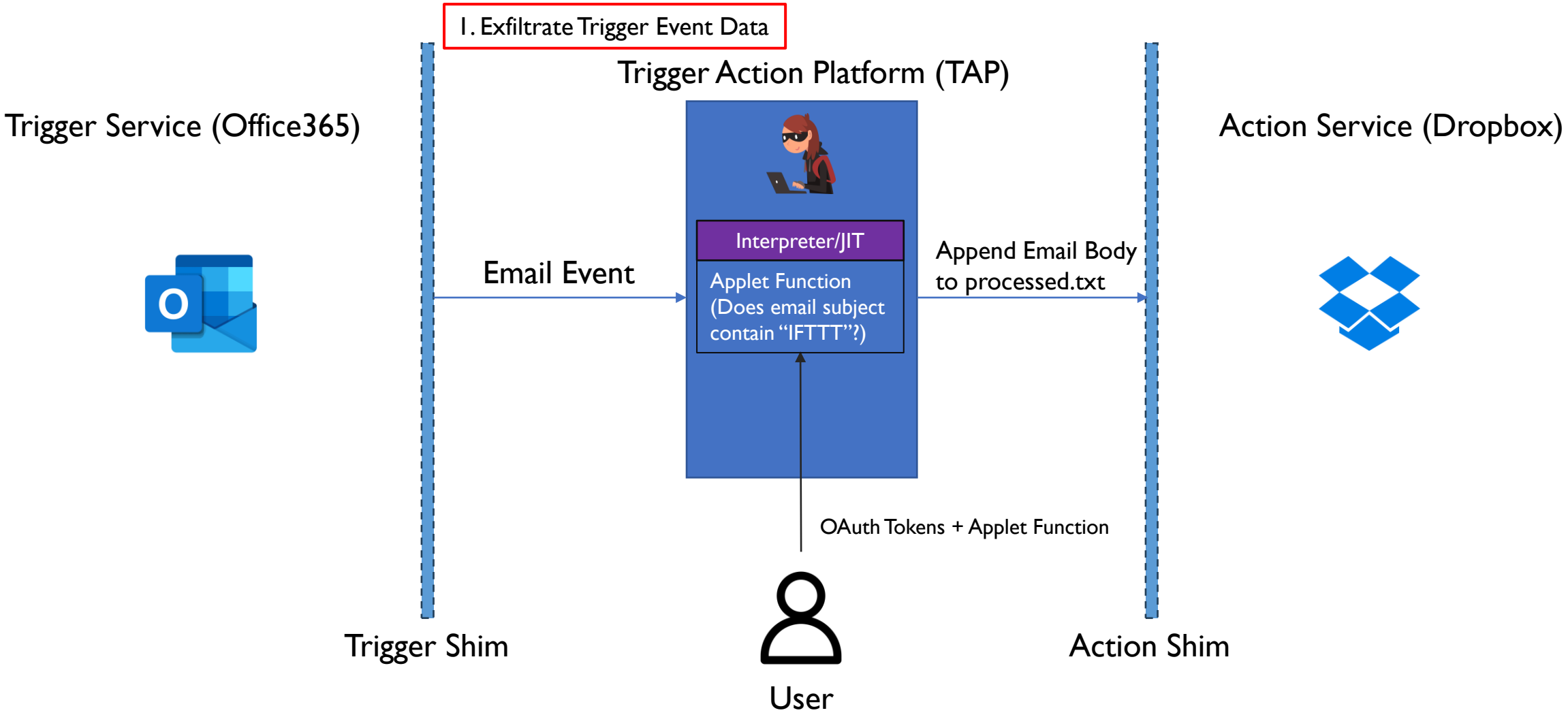
Threat Model



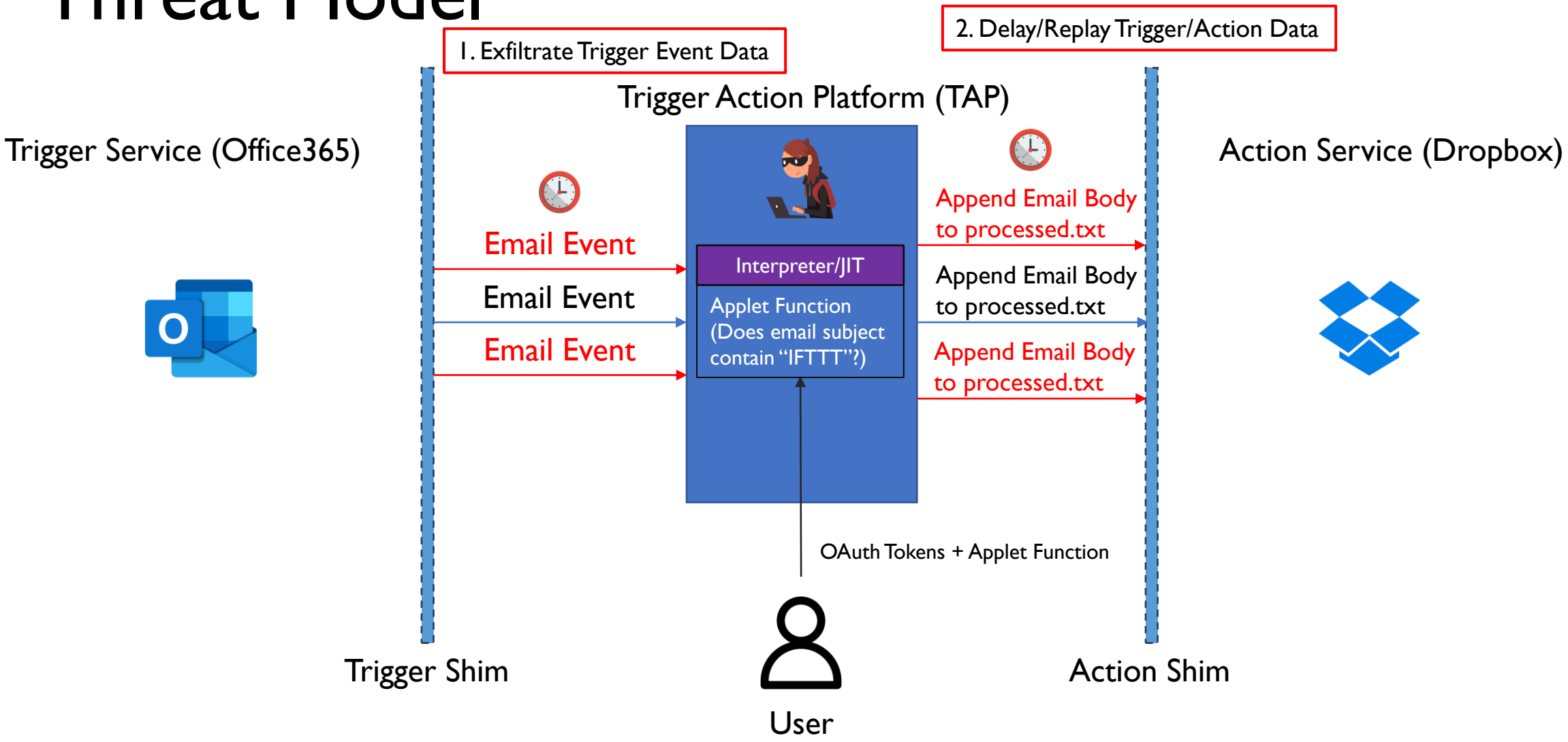
Threat Model



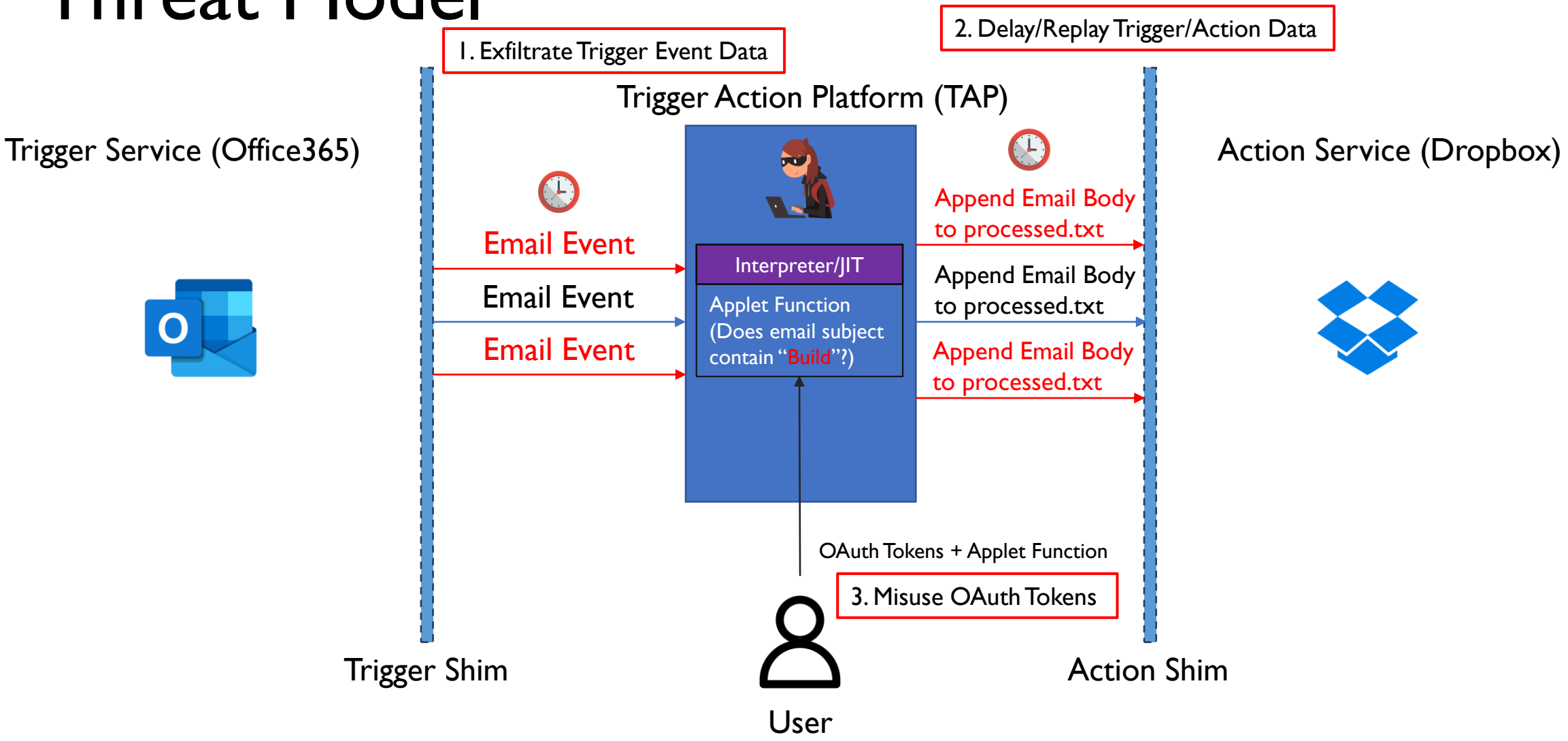
Threat Model



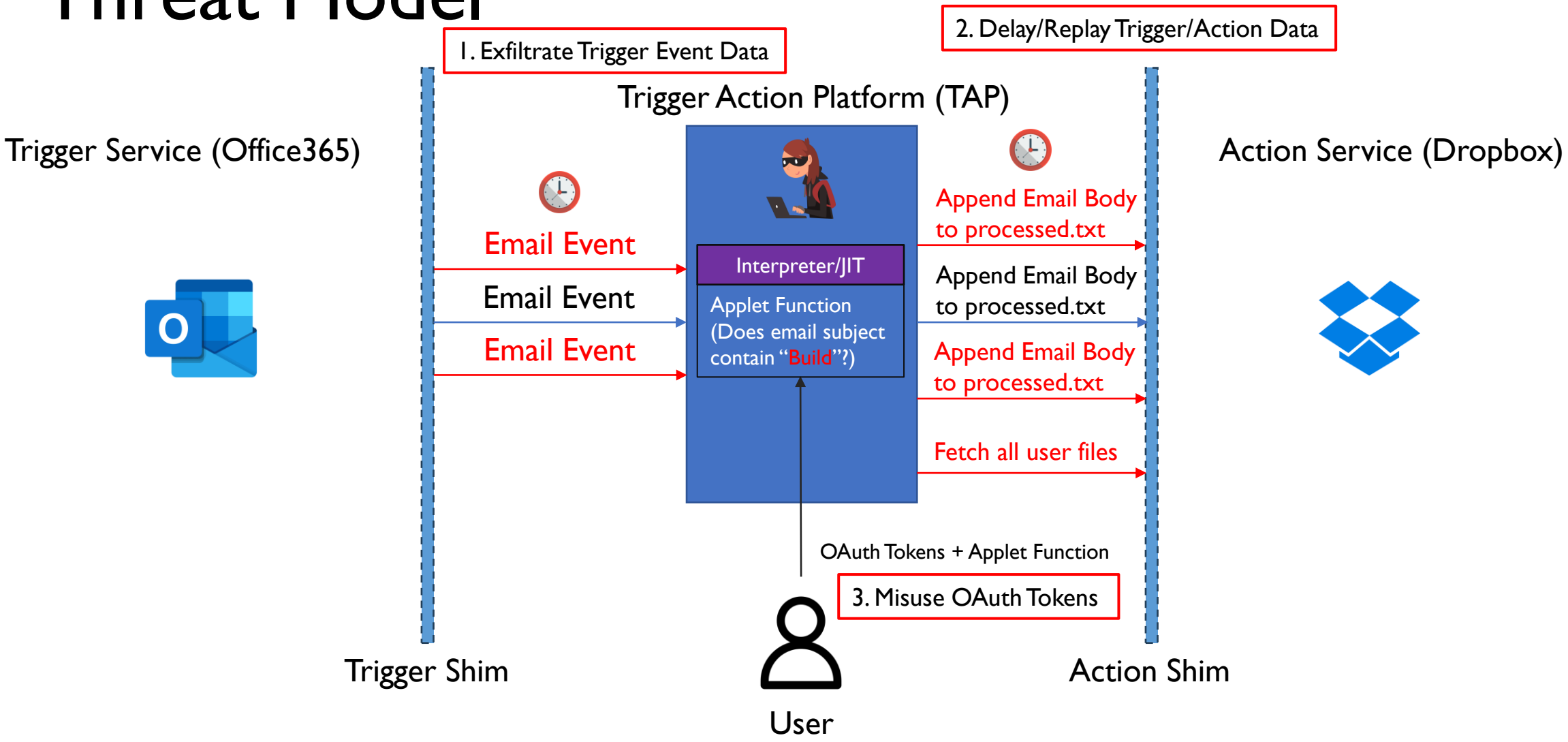
Threat Model



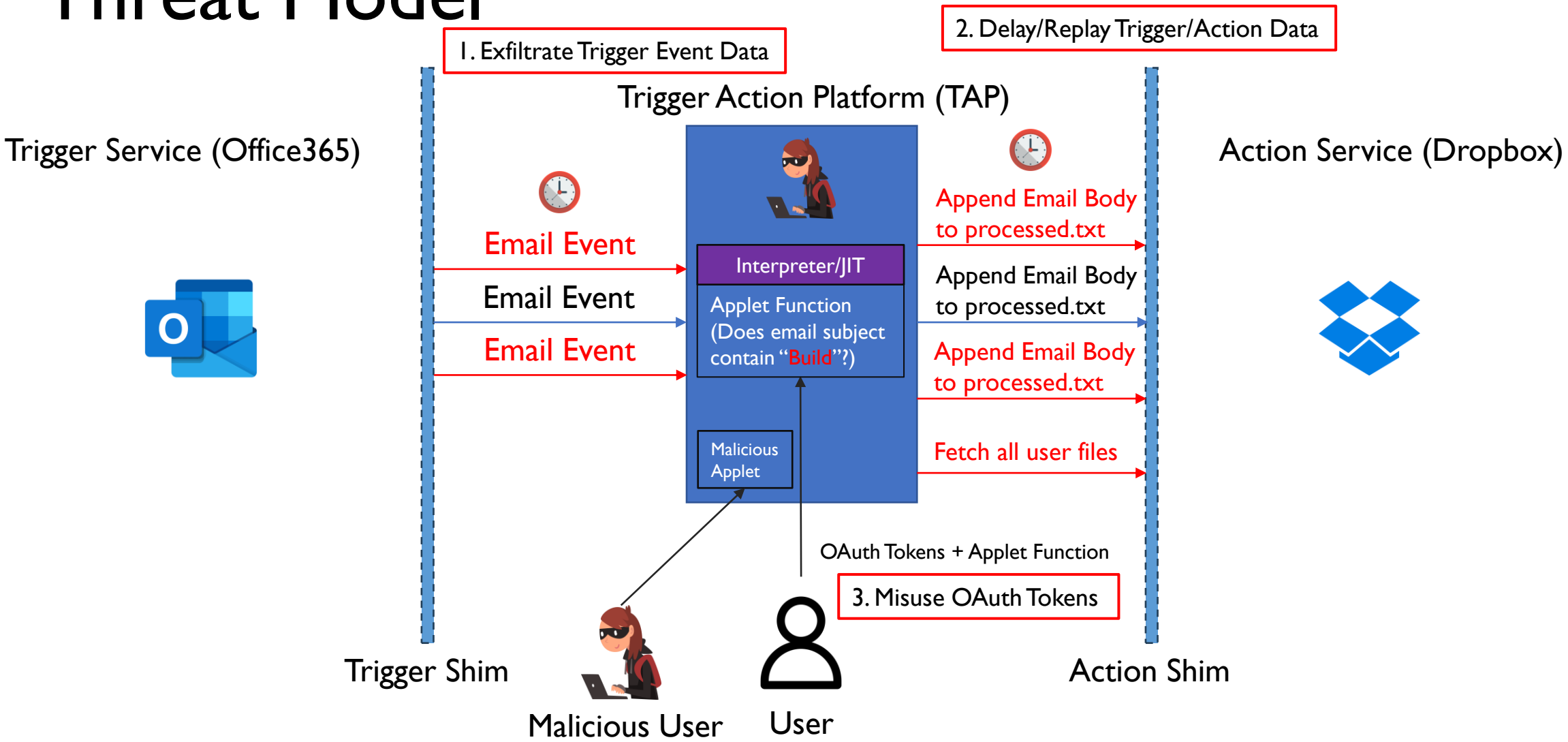
Threat Model



Threat Model



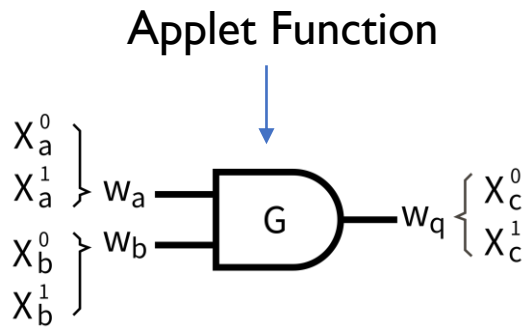
Threat Model



Can we design a TAP that can defend against these attacks?

Existing Approaches

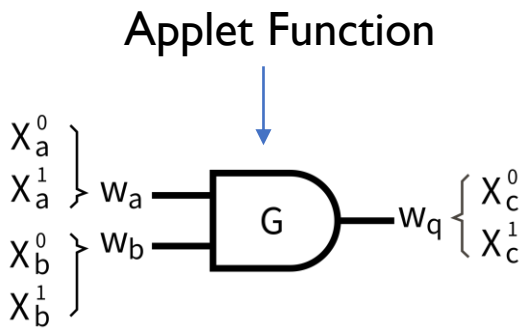
Existing Approaches



eTAP [S&P '21], Walnut [arXiv '20]

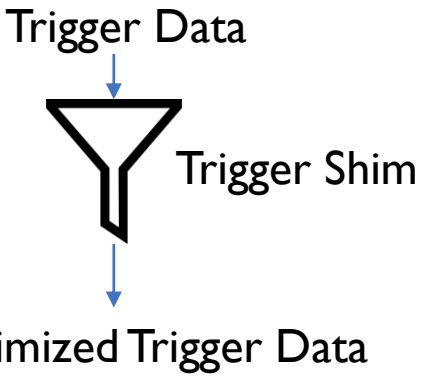
High Runtime Overhead and Restricted Programming Model

Existing Approaches



eTAP [S&P '21], Walnut [arXiv '20]

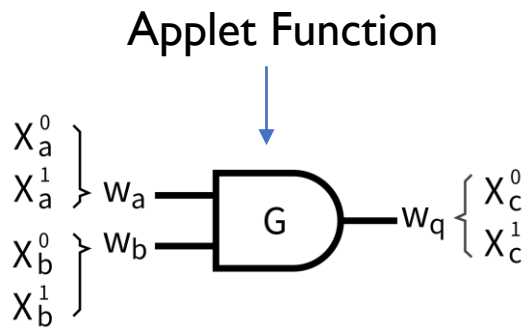
High Runtime Overhead and Restricted Programming Model



minTAP [USENIX Sec '22]

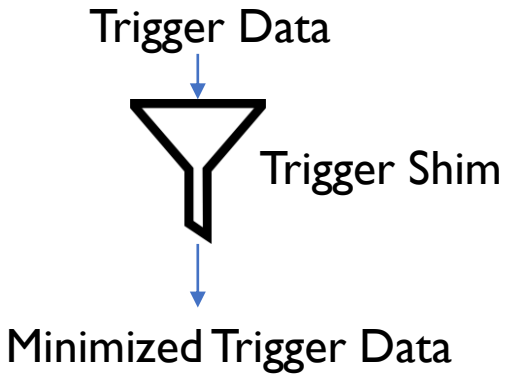
No Applet Execution Integrity

Existing Approaches



eTAP [S&P '21], Walnut [arXiv '20]

High Runtime Overhead and Restricted Programming Model



minTAP [USENIX Sec '22]

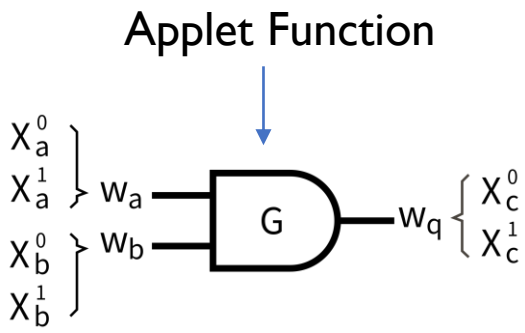
No Applet Execution Integrity

Trigger Service → Action Service

DTAP [NDSS '18], OTAP[ESORICS '20]

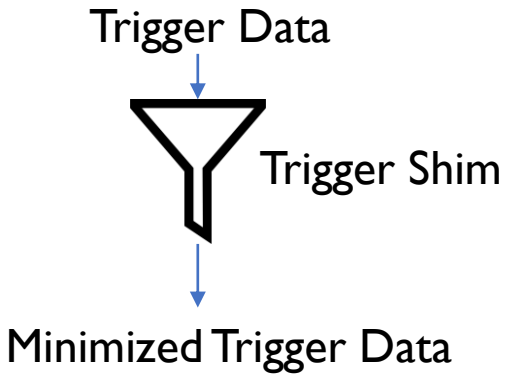
No Support for Applets

Existing Approaches



eTAP [S&P '21], Walnut [arXiv '20]

High Runtime Overhead and Restricted Programming Model



minTAP [USENIX Sec '22]

No Applet Execution Integrity

Trigger Service → Action Service

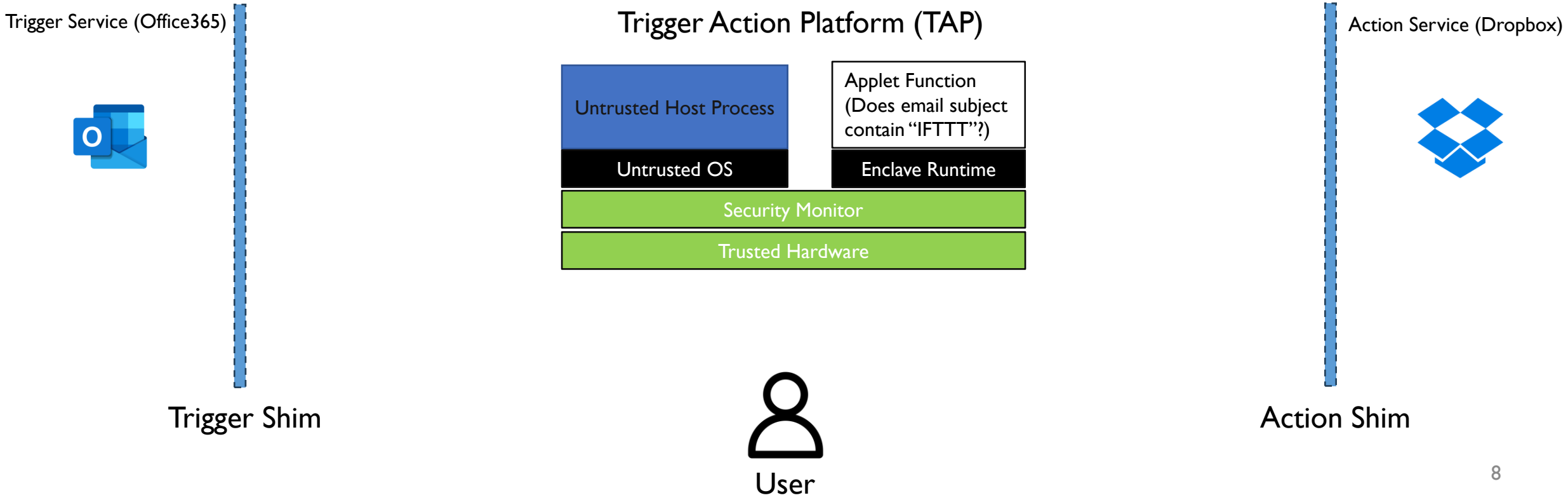
DTAP [NDSS '18], OTAP[ESORICS '20]

No Support for Applets

Can we design a TAP that can defend against these attacks along with high performance and functionality?

Our Approach: TAPDance

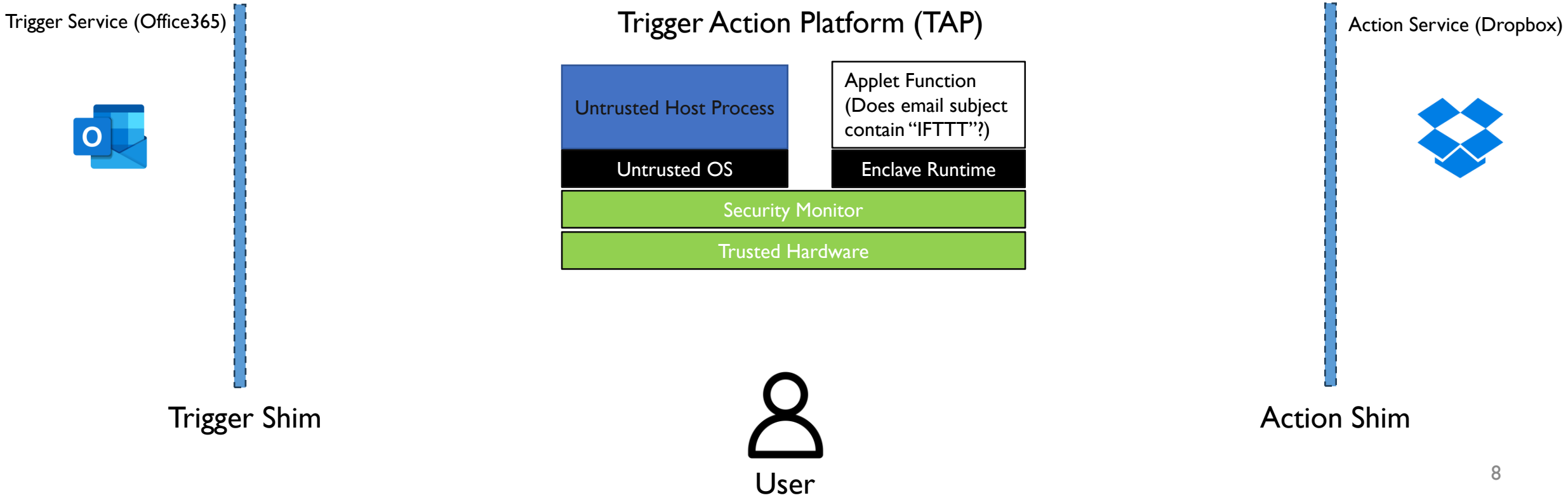
Insight: TAPs have a restricted execution model; Run Applet inside a RISC-V Keystone hardware enclave



Our Approach: TAPDance

Insight: TAPs have a restricted execution model; Run Applet inside a RISC-V Keystone hardware enclave

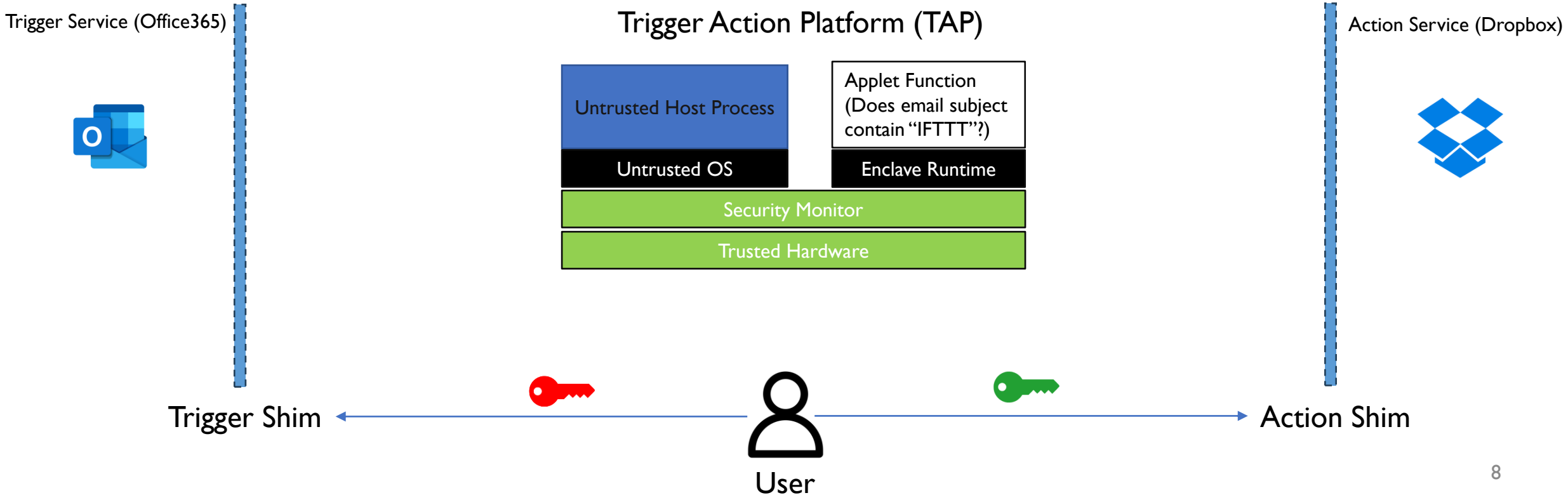
Keystone Security Monitor provides customizability



Our Approach: TAPDance

Insight: TAPs have a restricted execution model; Run Applet inside a RISC-V Keystone hardware enclave

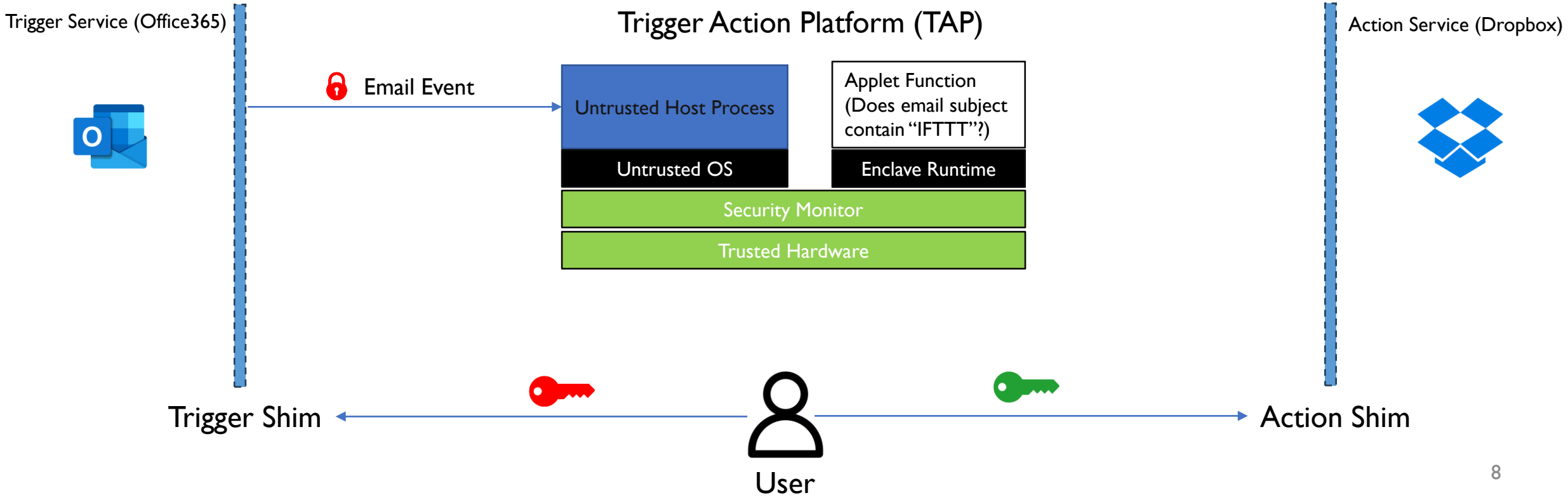
Keystone Security Monitor provides customizability



Our Approach: TAPDance

Insight: TAPs have a restricted execution model; Run Applet inside a RISC-V Keystone hardware enclave

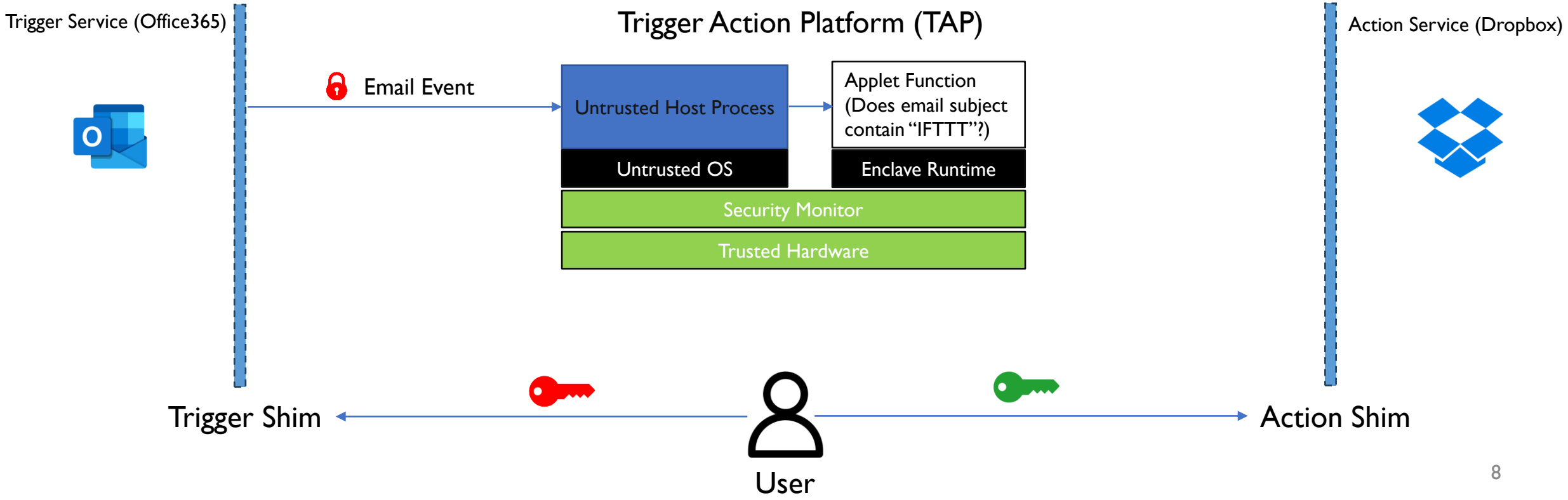
Keystone Security Monitor provides customizability



Our Approach: TAPDance

Insight: TAPs have a restricted execution model; Run Applet inside a RISC-V Keystone hardware enclave

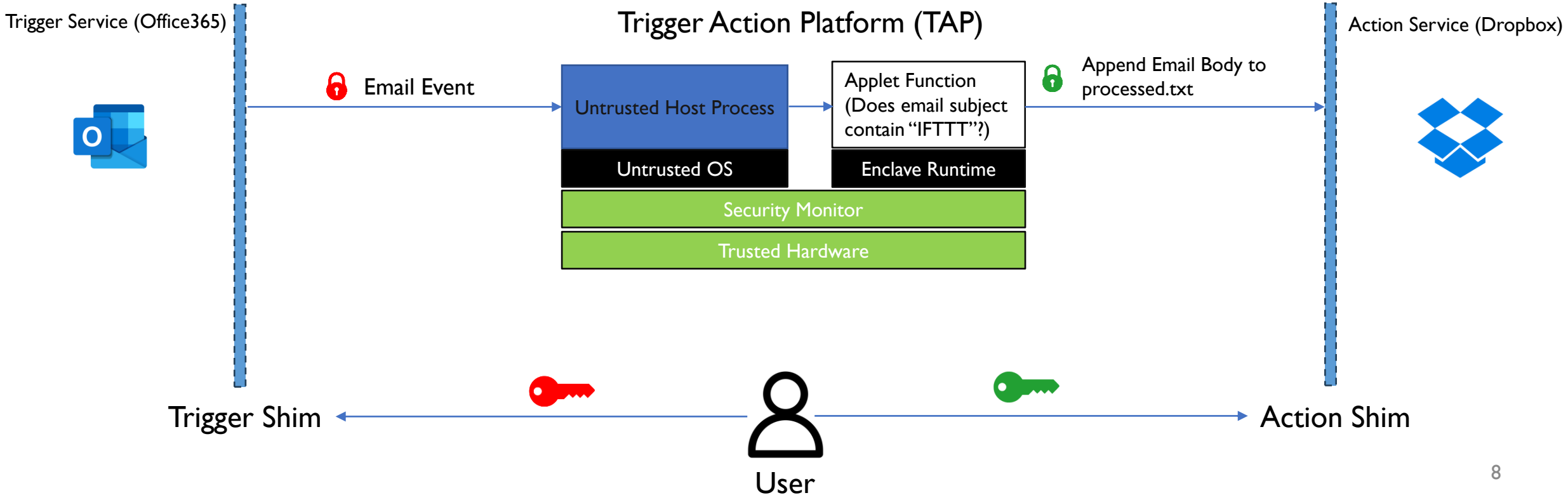
Keystone Security Monitor provides customizability



Our Approach: TAPDance

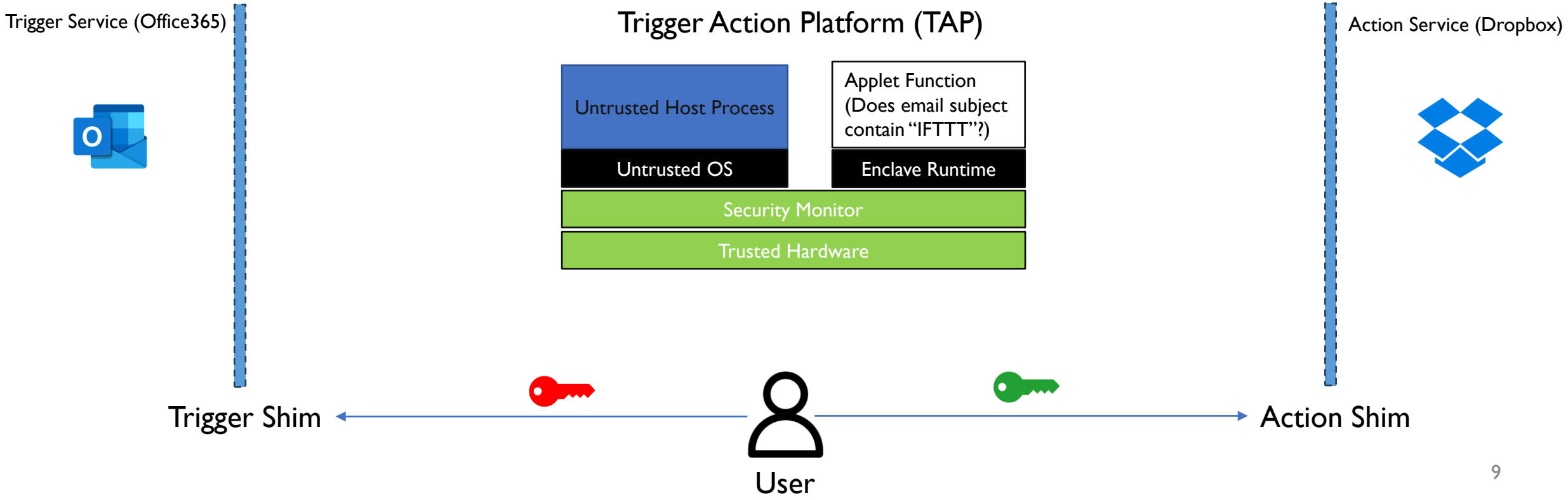
Insight: TAPs have a restricted execution model; Run Applet inside a RISC-V Keystone hardware enclave

Keystone Security Monitor provides customizability



Challenge 1: Freshness and Replay Protection

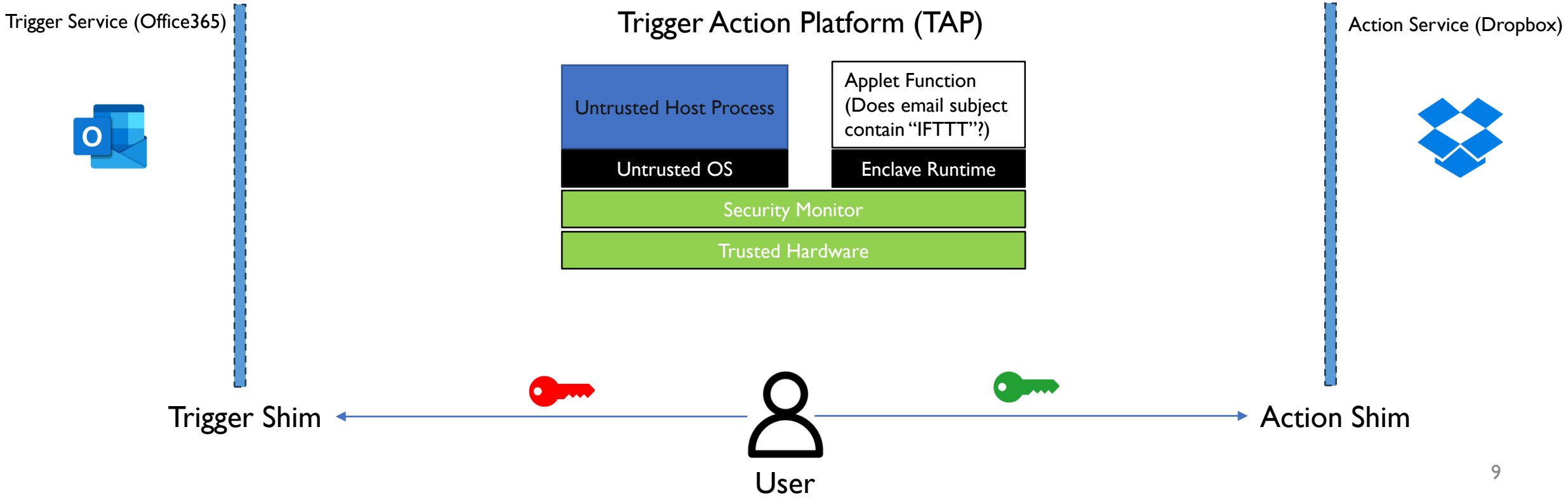
Problem: Freshness and Replay protection are not primitives offered by Enclaves



Challenge 1: Freshness and Replay Protection

Problem: Freshness and Replay protection are not primitives offered by Enclaves

Idea 1: Get Trigger Shim to associate and tag event data with nonce maintained by Security Monitor

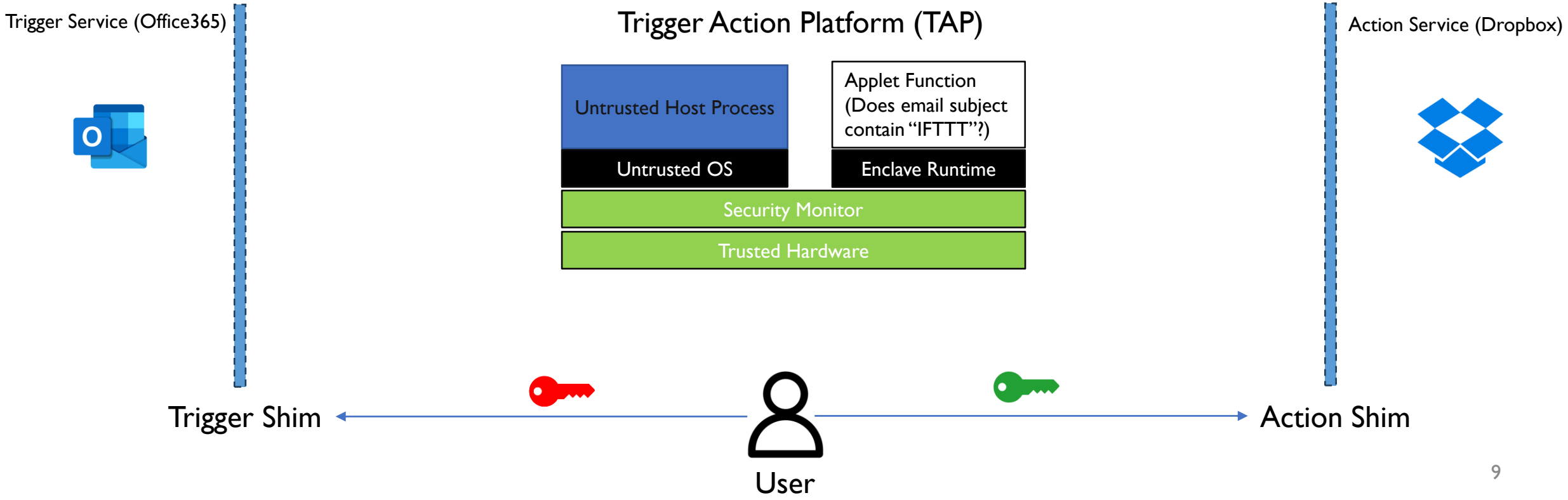


Challenge 1: Freshness and Replay Protection

Problem: Freshness and Replay protection are not primitives offered by Enclaves

Idea 1: Get Trigger Shim to associate and tag event data with nonce maintained by Security Monitor

Idea 2: Security Monitor provides a secure time service to the applet enclave

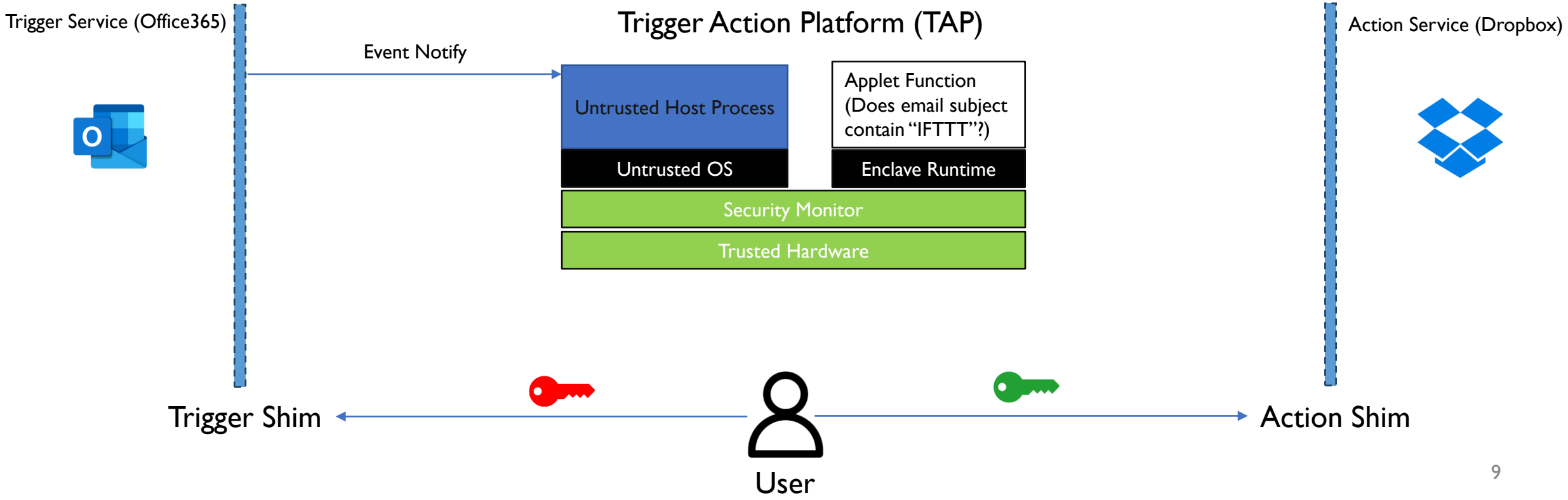


Challenge 1: Freshness and Replay Protection

Problem: Freshness and Replay protection are not primitives offered by Enclaves

Idea 1: Get Trigger Shim to associate and tag event data with nonce maintained by Security Monitor

Idea 2: Security Monitor provides a secure time service to the applet enclave

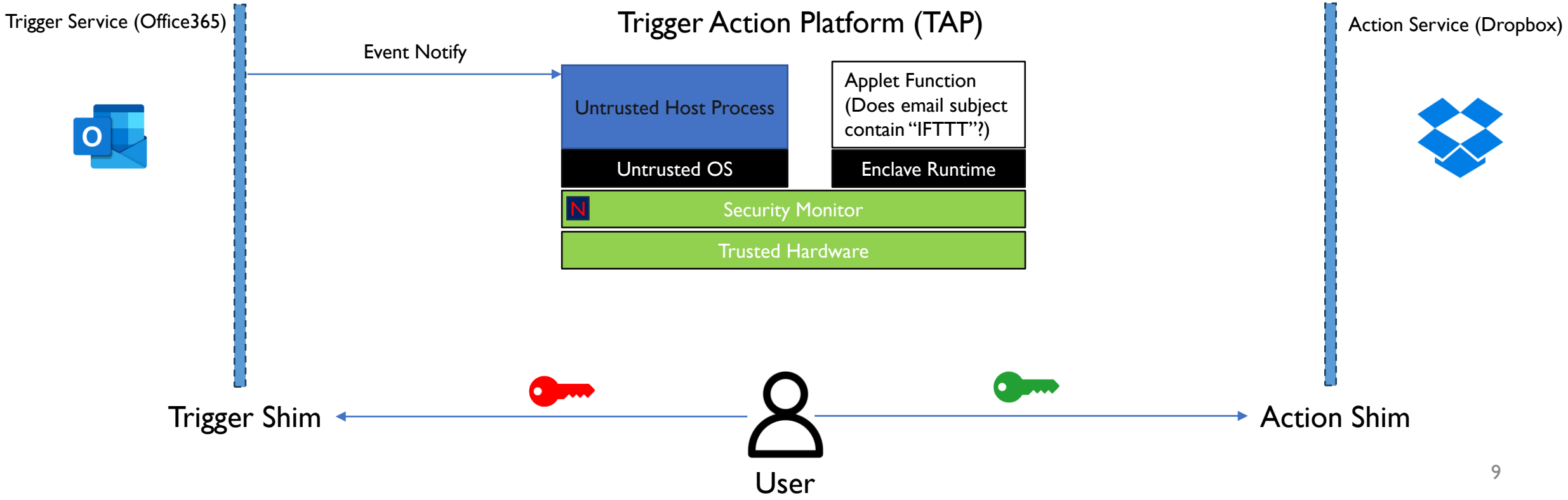


Challenge 1: Freshness and Replay Protection

Problem: Freshness and Replay protection are not primitives offered by Enclaves

Idea 1: Get Trigger Shim to associate and tag event data with nonce maintained by Security Monitor

Idea 2: Security Monitor provides a secure time service to the applet enclave

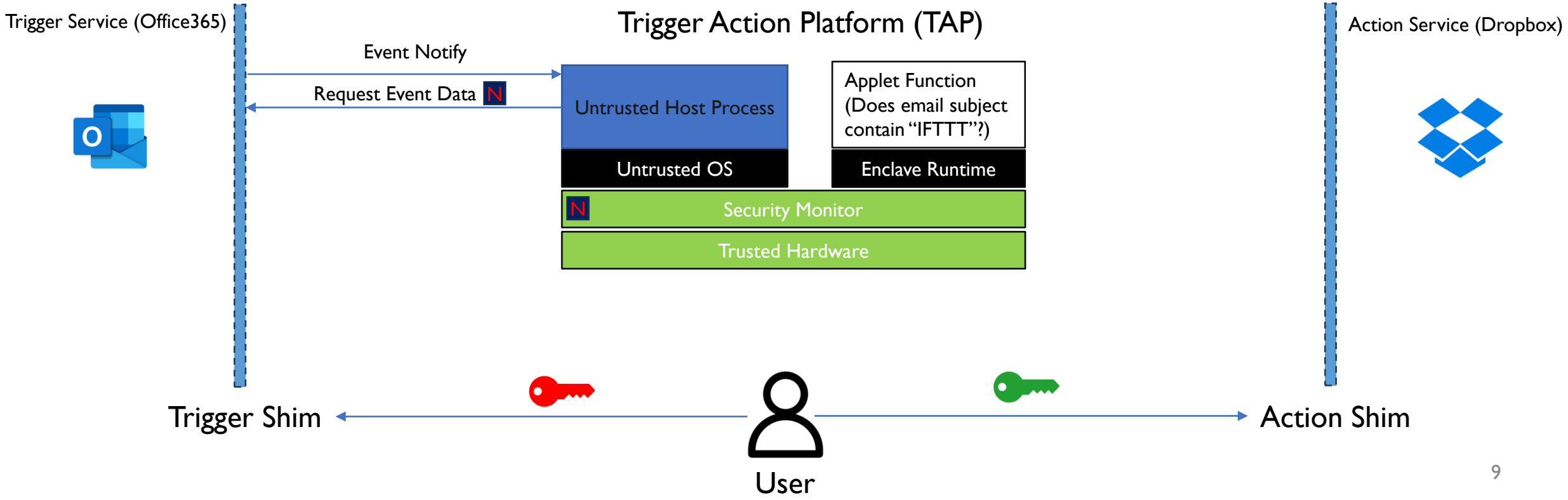


Challenge 1: Freshness and Replay Protection

Problem: Freshness and Replay protection are not primitives offered by Enclaves

Idea 1: Get Trigger Shim to associate and tag event data with nonce maintained by Security Monitor

Idea 2: Security Monitor provides a secure time service to the applet enclave

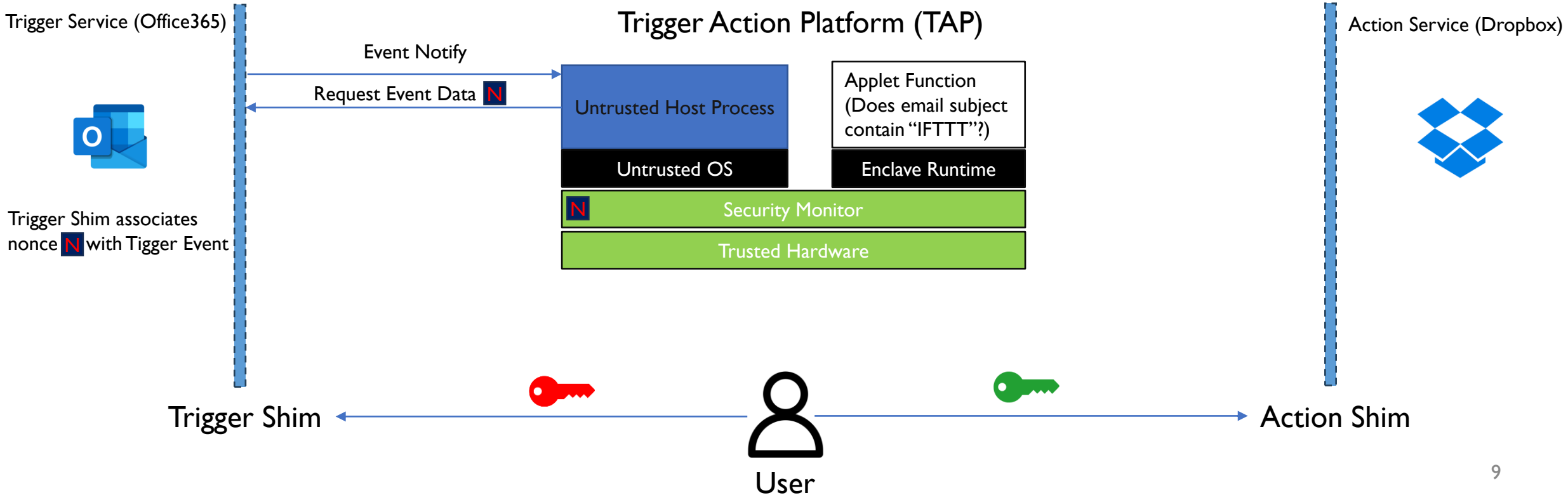


Challenge 1: Freshness and Replay Protection

Problem: Freshness and Replay protection are not primitives offered by Enclaves

Idea 1: Get Trigger Shim to associate and tag event data with nonce maintained by Security Monitor

Idea 2: Security Monitor provides a secure time service to the applet enclave

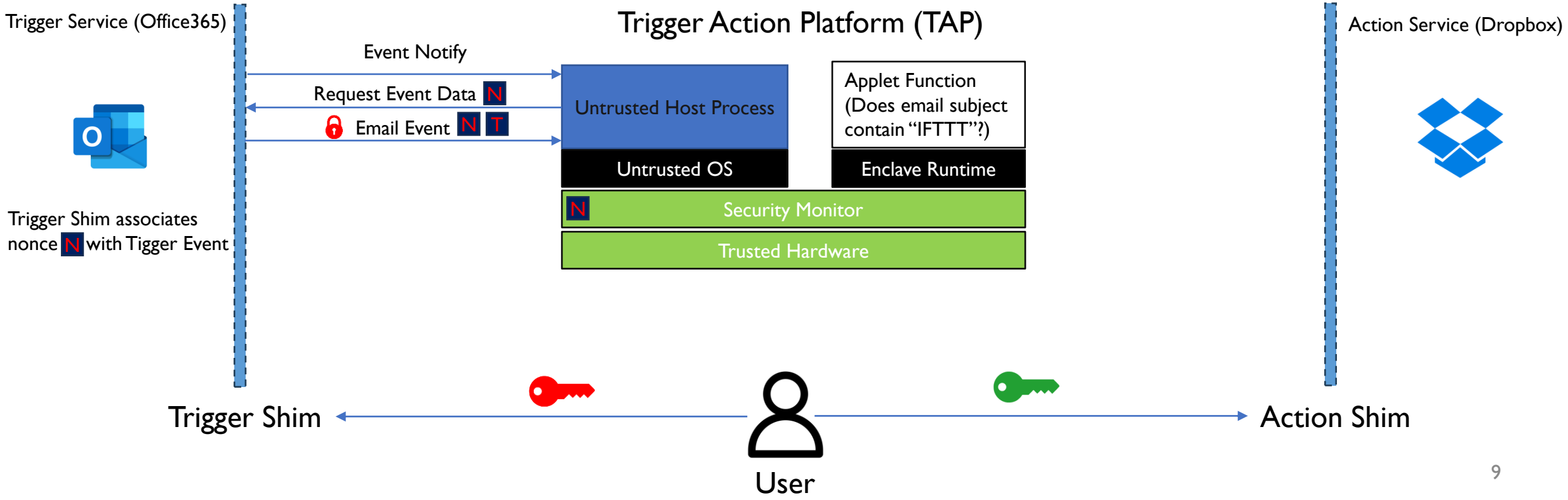


Challenge 1: Freshness and Replay Protection

Problem: Freshness and Replay protection are not primitives offered by Enclaves

Idea 1: Get Trigger Shim to associate and tag event data with nonce maintained by Security Monitor

Idea 2: Security Monitor provides a secure time service to the applet enclave

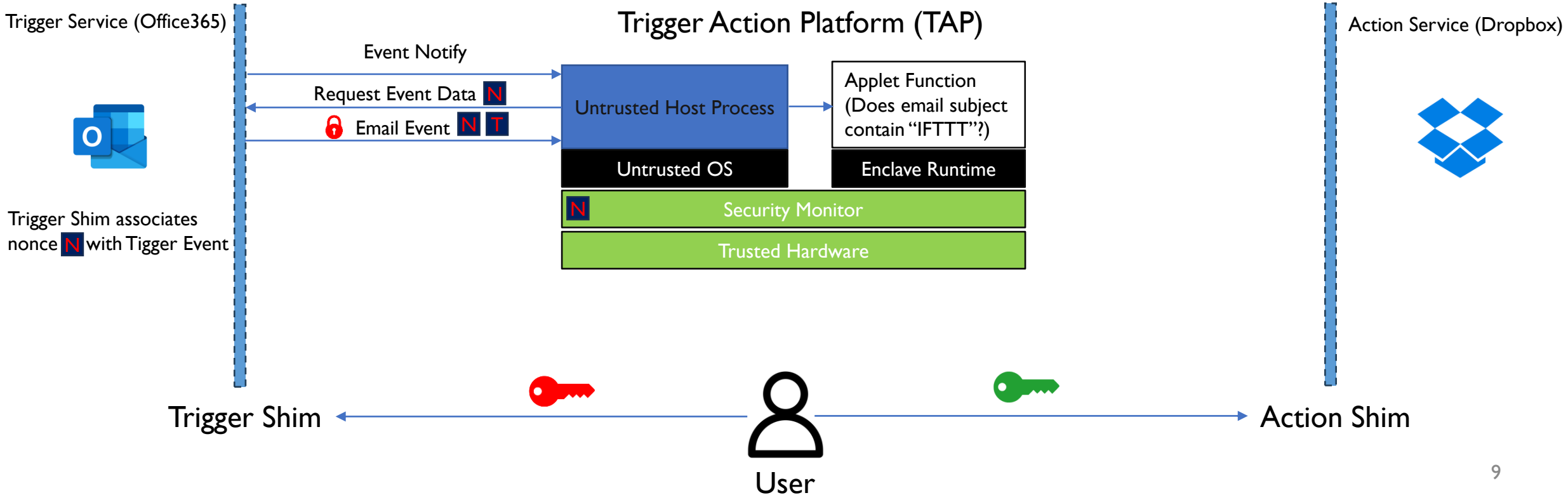


Challenge 1: Freshness and Replay Protection

Problem: Freshness and Replay protection are not primitives offered by Enclaves

Idea 1: Get Trigger Shim to associate and tag event data with nonce maintained by Security Monitor

Idea 2: Security Monitor provides a secure time service to the applet enclave

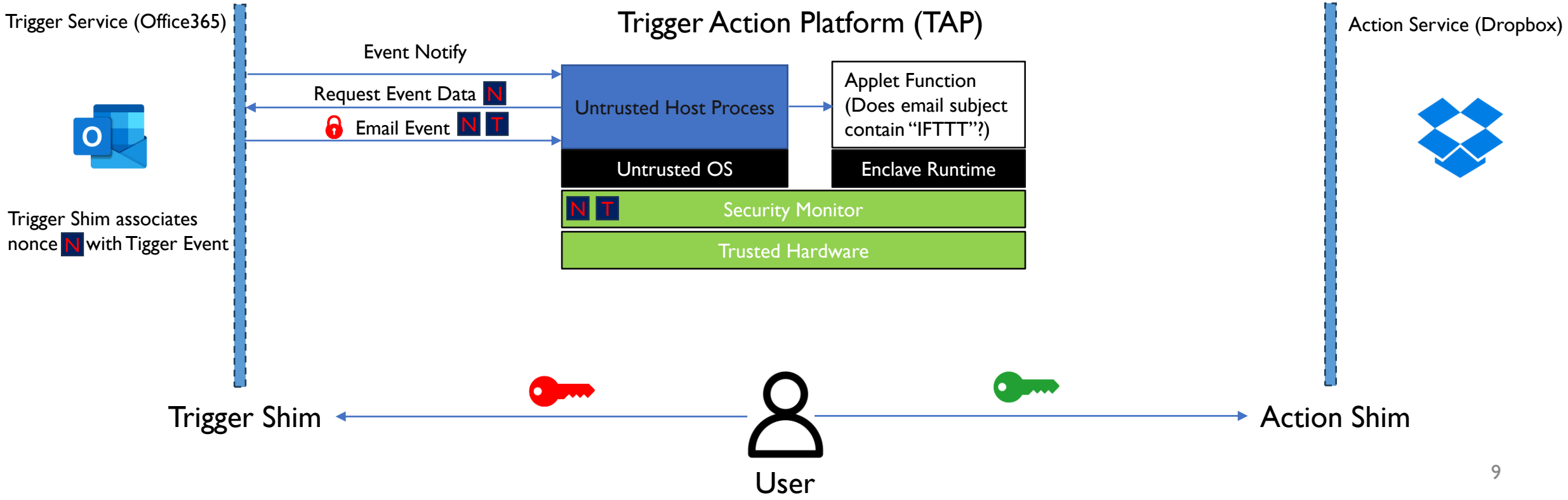


Challenge 1: Freshness and Replay Protection

Problem: Freshness and Replay protection are not primitives offered by Enclaves

Idea 1: Get Trigger Shim to associate and tag event data with nonce maintained by Security Monitor

Idea 2: Security Monitor provides a secure time service to the applet enclave

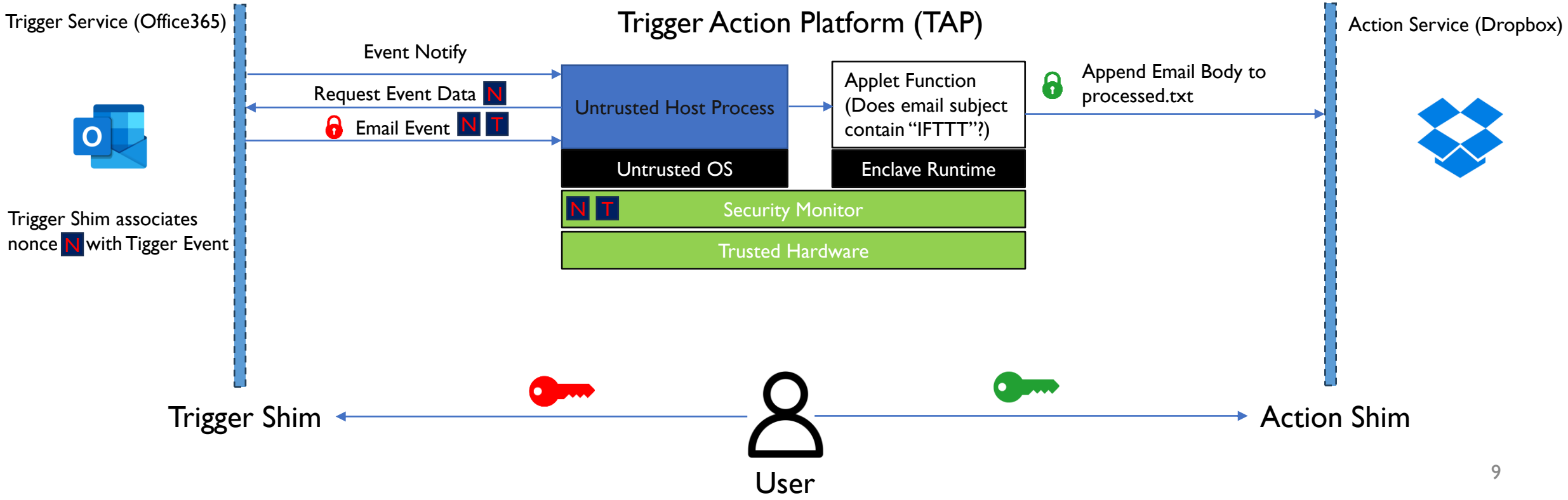


Challenge 1: Freshness and Replay Protection

Problem: Freshness and Replay protection are not primitives offered by Enclaves

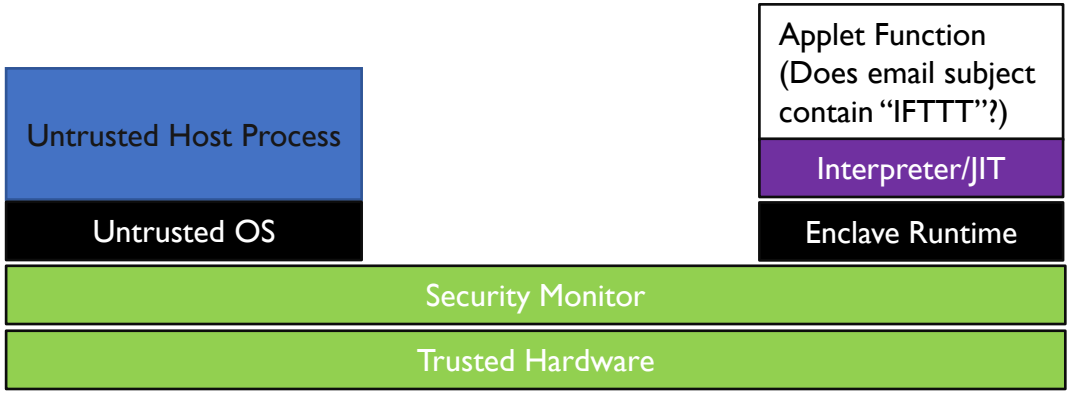
Idea 1: Get Trigger Shim to associate and tag event data with nonce maintained by Security Monitor

Idea 2: Security Monitor provides a secure time service to the applet enclave



Challenge 2: Reducing Enclave TCB

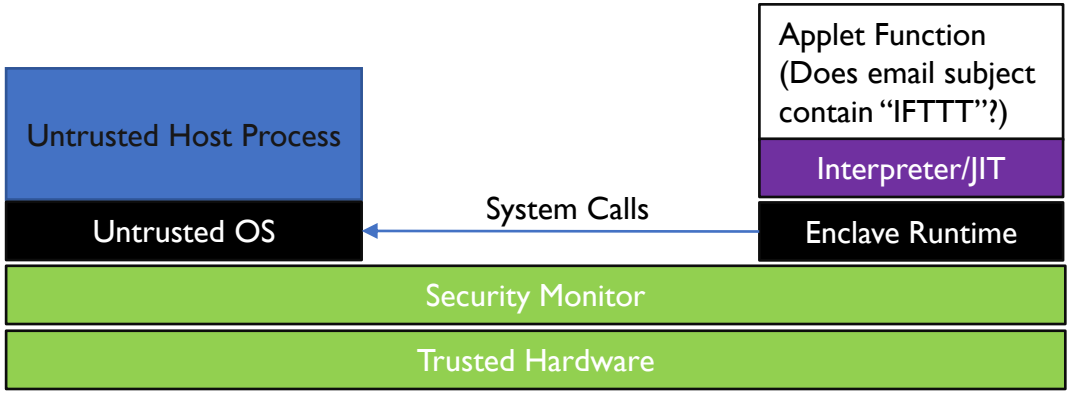
Problem: Running an interpreter/JIT in the enclave increases the size of the software TCB in the enclave



Challenge 2: Reducing Enclave TCB

Problem: Running an interpreter/JIT in the enclave increases the size of the software TCB in the enclave

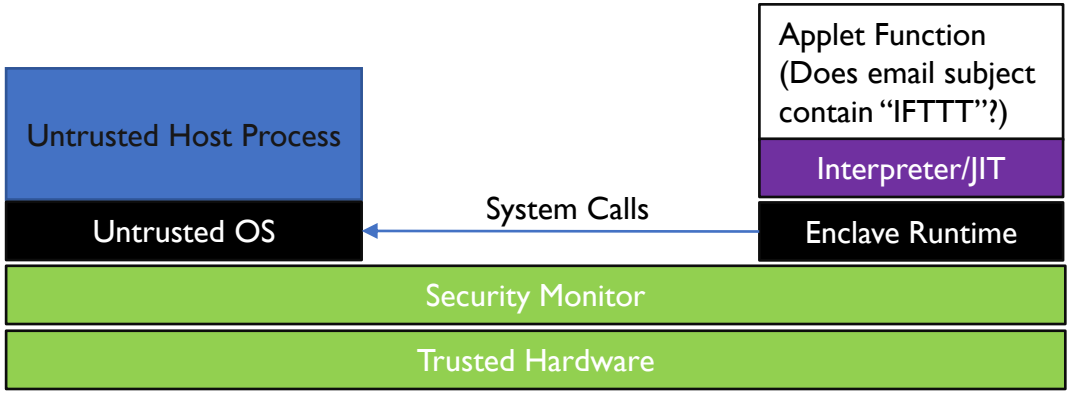
109 KLoC (Node.js frontend) + 935 KLoC (V8 Backend) + Large Syscall Interface to Untrusted OS



Challenge 2: Reducing Enclave TCB

Problem: Running an interpreter/JIT in the enclave increases the size of the software TCB in the enclave

109 KLoC (Node.js frontend) + 935 KLoC (V8 Backend) + Large Syscall Interface to Untrusted OS



How to run TypeScript applets while reducing the software TCB of the enclave?

Challenge 2: Reducing Enclave TCB

Insight: Applets are pure computations that do not require all the features of an interpreter or OS support

Challenge 2: Reducing Enclave TCB

Insight: Applets are pure computations that do not require all the features of an interpreter or OS support

Idea 1: Restrict allowed susbset of TypeScript and compile to machine code with LLVM IR as the intermediate step

Challenge 2: Reducing Enclave TCB

Insight: Applets are pure computations that do not require all the features of an interpreter or OS support

Idea 1: Restrict allowed susbset of TypeScript and compile to machine code with LLVM IR as the intermediate step

Idea 2: Restrict the host interface to allow only 10 external calls to the untrusted host OS

Challenge 2: Reducing Enclave TCB

Insight: Applets are pure computations that do not require all the features of an interpreter or OS support

Idea 1: Restrict allowed susbset of TypeScript and compile to machine code with LLVM IR as the intermediate step

Idea 2: Restrict the host interface to allow only 10 external calls to the untrusted host OS

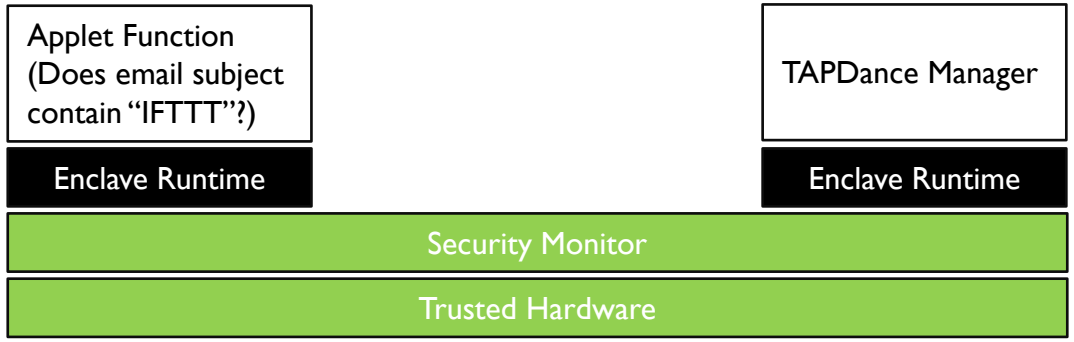
```
var searchResult = Office365Mail.newEmailFrom.Subject.search("IFTTT");

if (searchResult != -1) {
    Dropbox.appendToTextFileDb.append(Office365Mail.newEmailFrom.Body,
"processed.txt");
} else {
    Dropbox.appendToTextFileDb.skip();
}
```

```
@0 = private unnamed_addr constant [6 x i8] c"IFTTT\00", align 1
@1 = private unnamed_addr constant [14 x i8] c"processed.txt\00", align 1

define i64 @__rule_function() {
entry:
  %0 = call double @Office365Mail_newEmailFrom_Subject_search(i8* getelementptr inbounds ([6
x i8], [6 x i8]* @0, i32 0, i32 0))
  %searchResult = alloca double, align 8
  store double %0, double* %searchResult, align 8
  %1 = load double, double* %searchResult, align 8
  %2 = fcmp one double %1, -1.000000e+00
  br i1 %2, label %if.true, label %if.false

if.true:                                          ; preds = %entry
  %3 = call i8* @Office365Mail_newEmailFrom_Body()
  call void @Dropbox_appendToTextFileDb_append(i8* %3, i8* getelementptr inbounds ([14 x i8],
[14 x i8]* @1, i32 0, i32 0))
  br label %if.end

if.end:                                           ; preds = %if.true, %if.false
  ret i64 0

if.false:                                         ; preds = %entry
  call void @Dropbox_appendToTextFileDb_skip()
  br label %if.end
}
```

Challenge 3: Runtime Attestation

Idea 1: Have a long running attestation manager enclave that the user attests and provides keys to

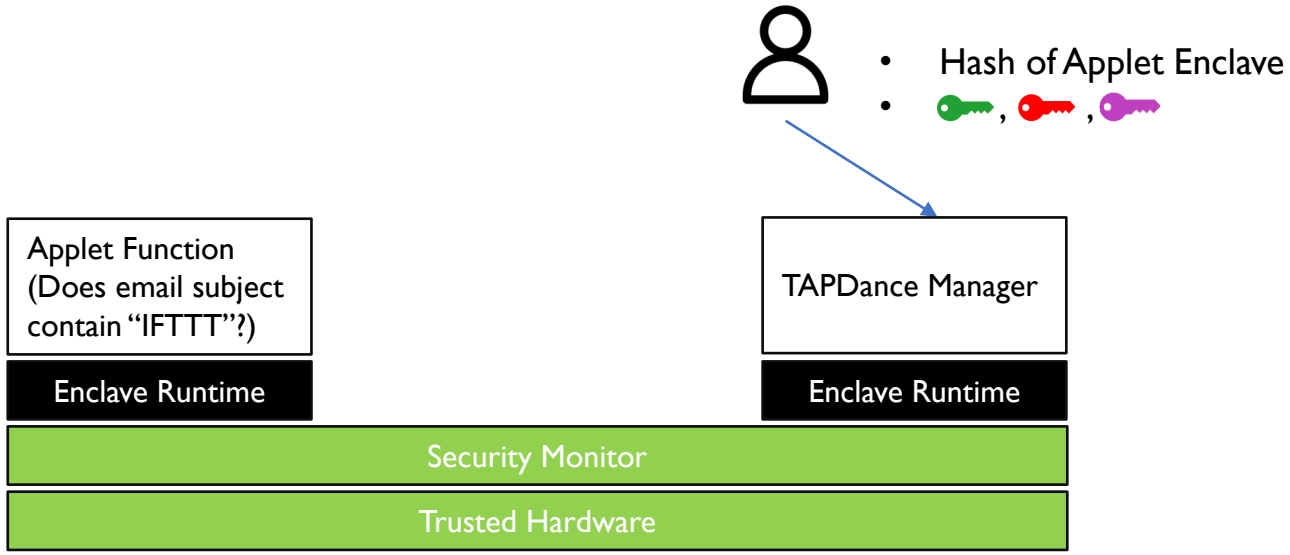
Idea 2: Encrypt Applet Binary 🔑 such that it decrypts and runs only after successful attestation



Challenge 3: Runtime Attestation

Idea 1: Have a long running attestation manager enclave that the user attests and provides keys to

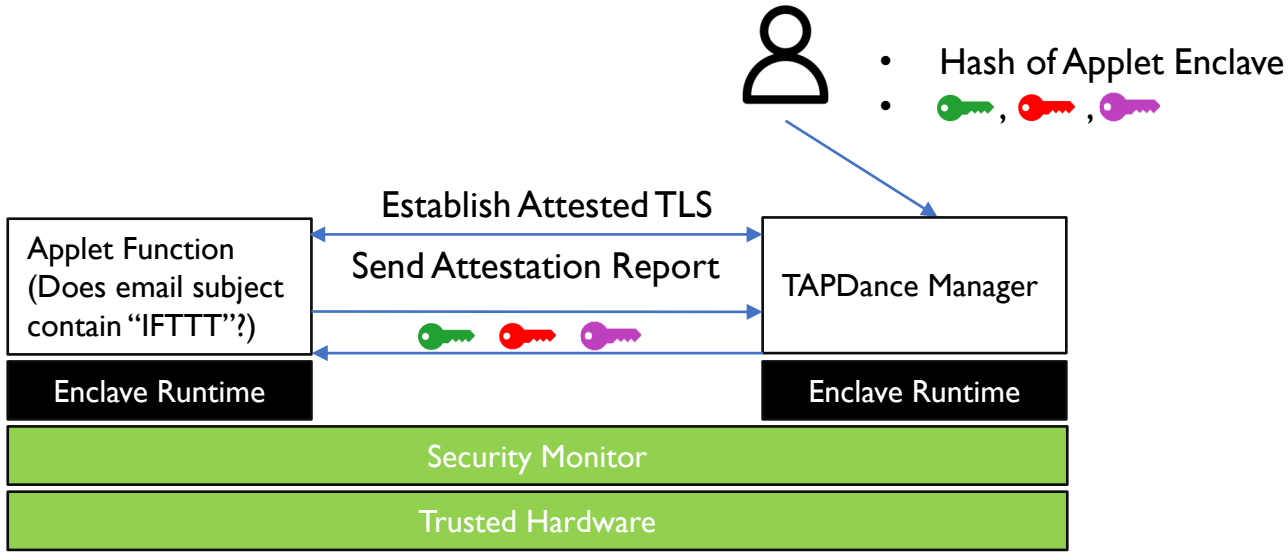
Idea 2: Encrypt Applet Binary such that it decrypts and runs only after successful attestation



Challenge 3: Runtime Attestation

Idea 1: Have a long running attestation manager enclave that the user attests and provides keys to

Idea 2: Encrypt Applet Binary such that it decrypts and runs only after successful attestation



Summary of Problems and Solutions

1. Exfiltrate Trigger Event Data

2. Delay/Replay Trigger/Action Data

3. Misuse of OAuth Tokens

4. Size of Enclave TCB to run TypeScript Applets

5. Malicious User

Summary of Problems and Solutions

1. Exfiltrate Trigger Event Data → Use of Enclaves + Encryption of Trigger and Action Data

2. Delay/Replay Trigger/Action Data

3. Misuse of OAuth Tokens

4. Size of Enclave TCB to run TypeScript Applets

5. Malicious User

Summary of Problems and Solutions

1. Exfiltrate Trigger Event Data → Use of Enclaves + Encryption of Trigger and Action Data

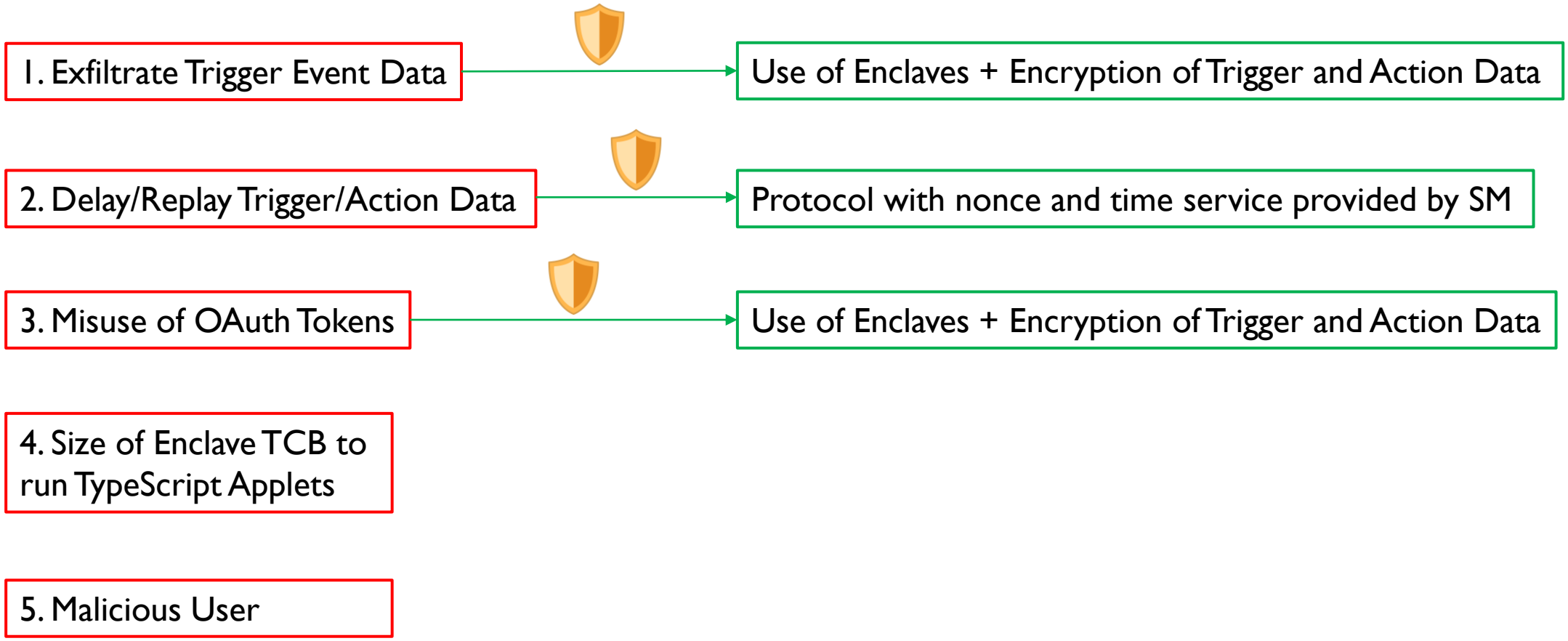
2. Delay/Replay Trigger/Action Data → Protocol with nonce and time service provided by SM

3. Misuse of OAuth Tokens

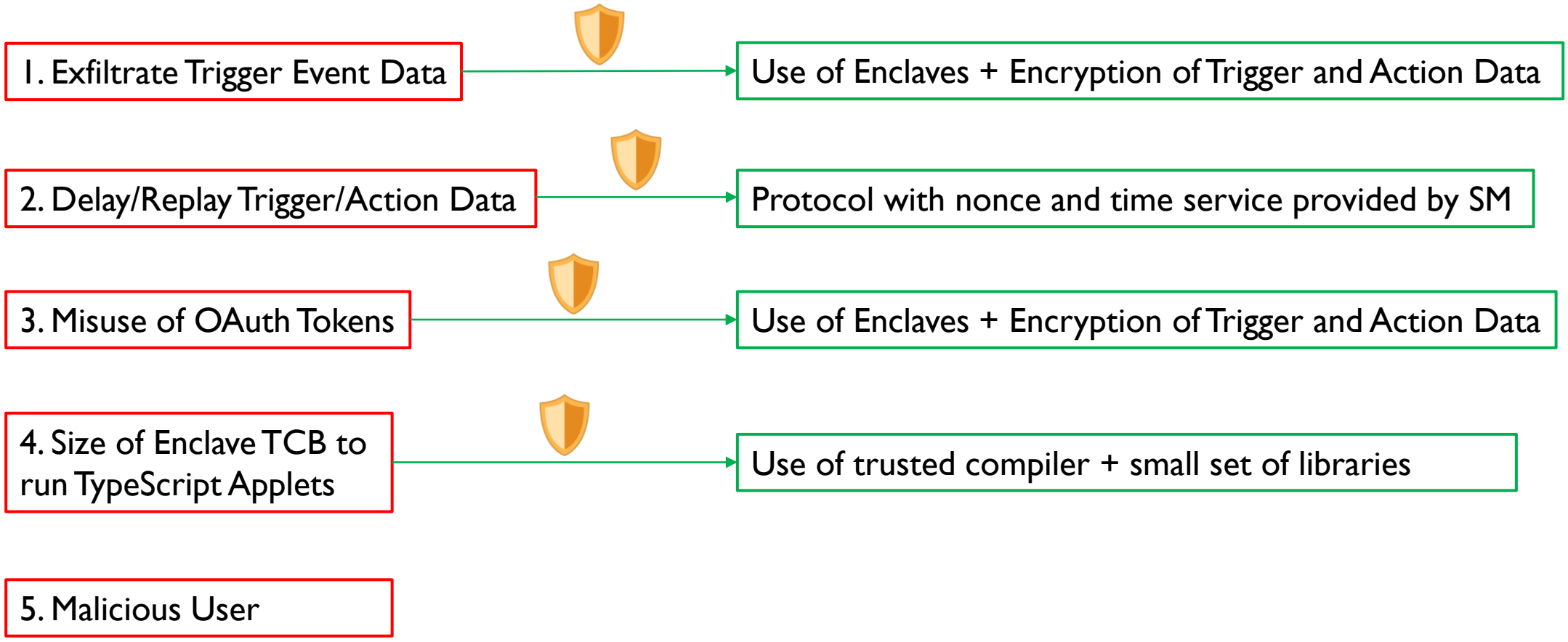
4. Size of Enclave TCB to run TypeScript Applets

5. Malicious User

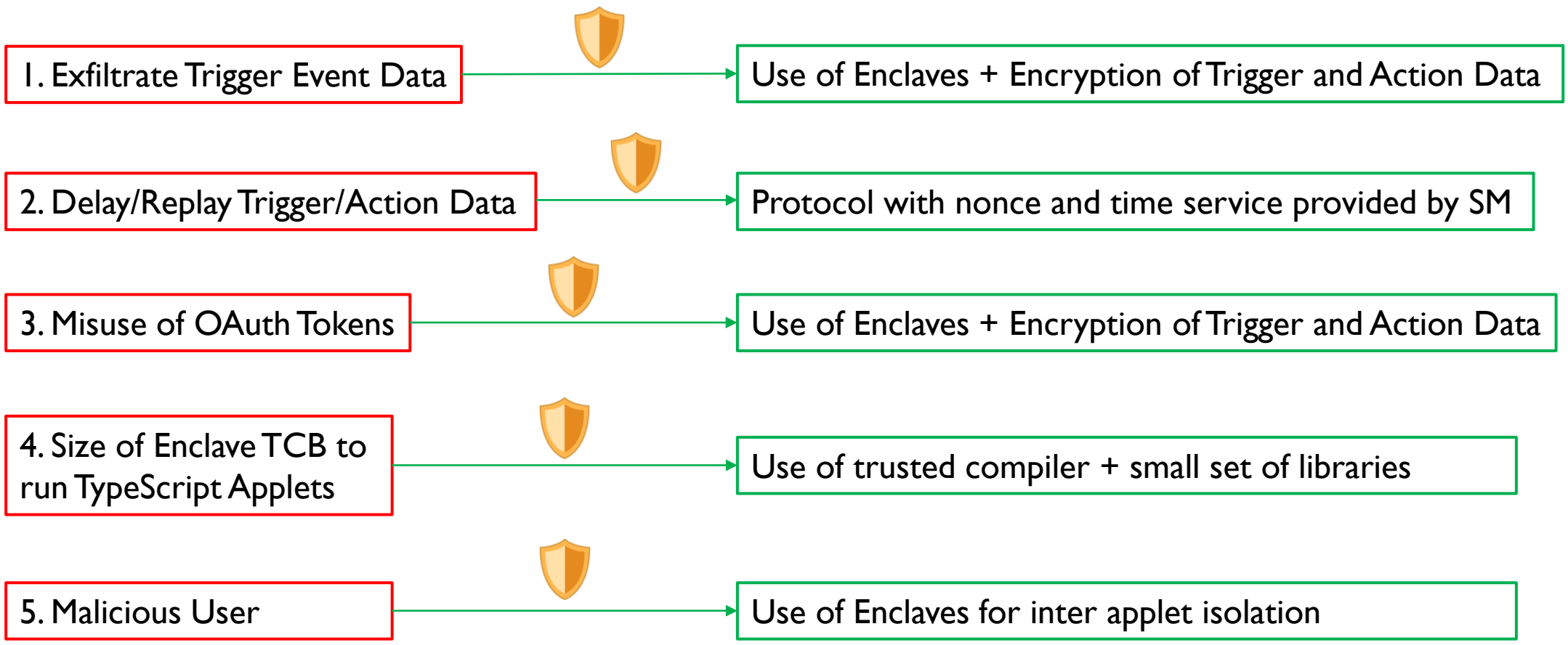
Summary of Problems and Solutions



Summary of Problems and Solutions



Summary of Problems and Solutions



Evaluation – Functionality + Performance

Evaluation – Functionality + Performance

Our TypeScript compiler can successfully compile 642/683 applets from the minTAP dataset

Evaluation – Functionality + Performance

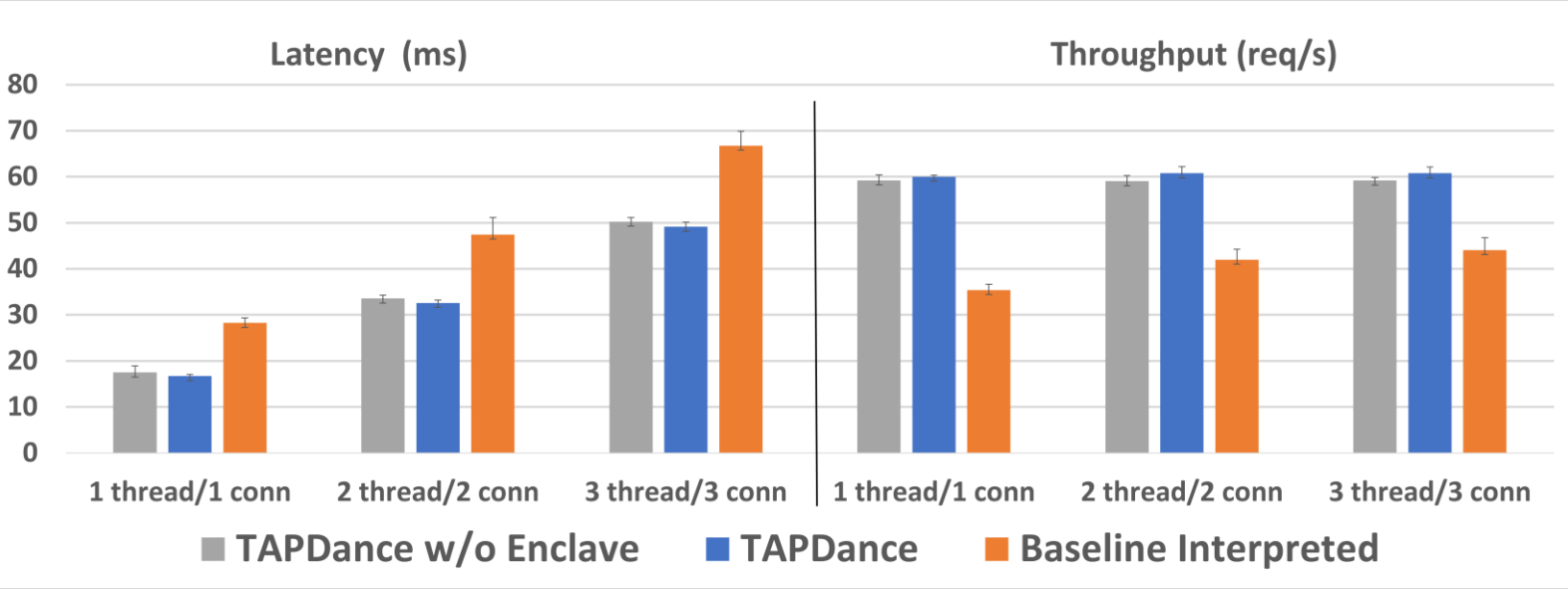
Our TypeScript compiler can successfully compile 642/683 applets from the minTAP dataset

Evaluated TAPDance on StarFive VisionFive v1 RISC-V Board by porting Keystone

Trigger and Action Services run on 32 core Intel Xeon E5-2630 processor running at 2.4 GHz with 128 GB RAM

Interpreted Baseline: Regular process running applets using Node.js v14.8.0

TAPDance w/o Enclave: Regular process running compiled applets



Evaluation – Functionality + Performance

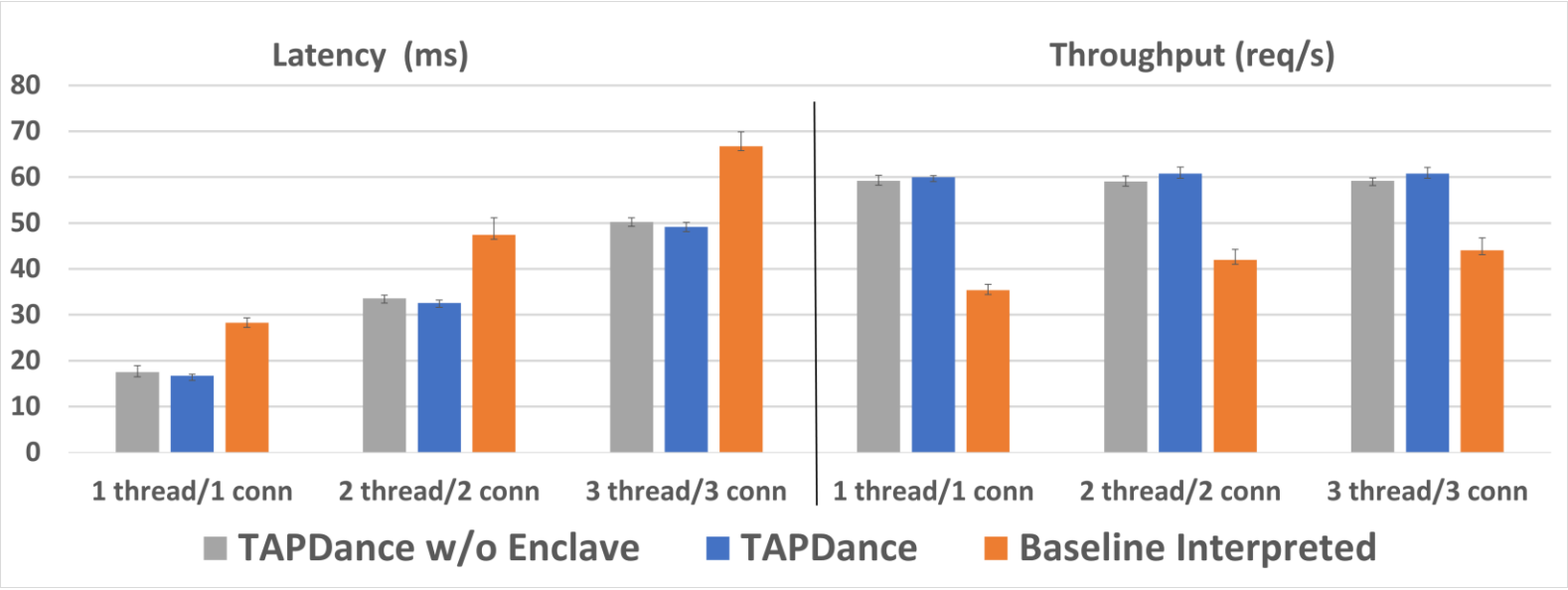
Our TypeScript compiler can successfully compile 642/683 applets from the minTAP dataset

Evaluated TAPDance on StarFive VisionFive v1 RISC-V Board by porting Keystone

Trigger and Action Services run on 32 core Intel Xeon E5-2630 processor running at 2.4 GHz with 128 GB RAM

Interpreted Baseline: Regular process running applets using Node.js v14.8.0

TAPDance w/o Enclave: Regular process running compiled applets



TAPDance has 32% lower latency than baseline

TAPDance has 33% higher throughput than baseline

Conclusion

- Current TAP architectures are fundamentally insecure – trigger and action data exposed to untrustworthy TAP system code
- Our insight is that applets are pure functions
- TAPDance uses the unique hardware protection of RISC-V to provide data privacy and applet execution integrity
- The performance loss is negligible

https://github.com/multifacet/tap_artifact/

Questions?