ESORICS 2022

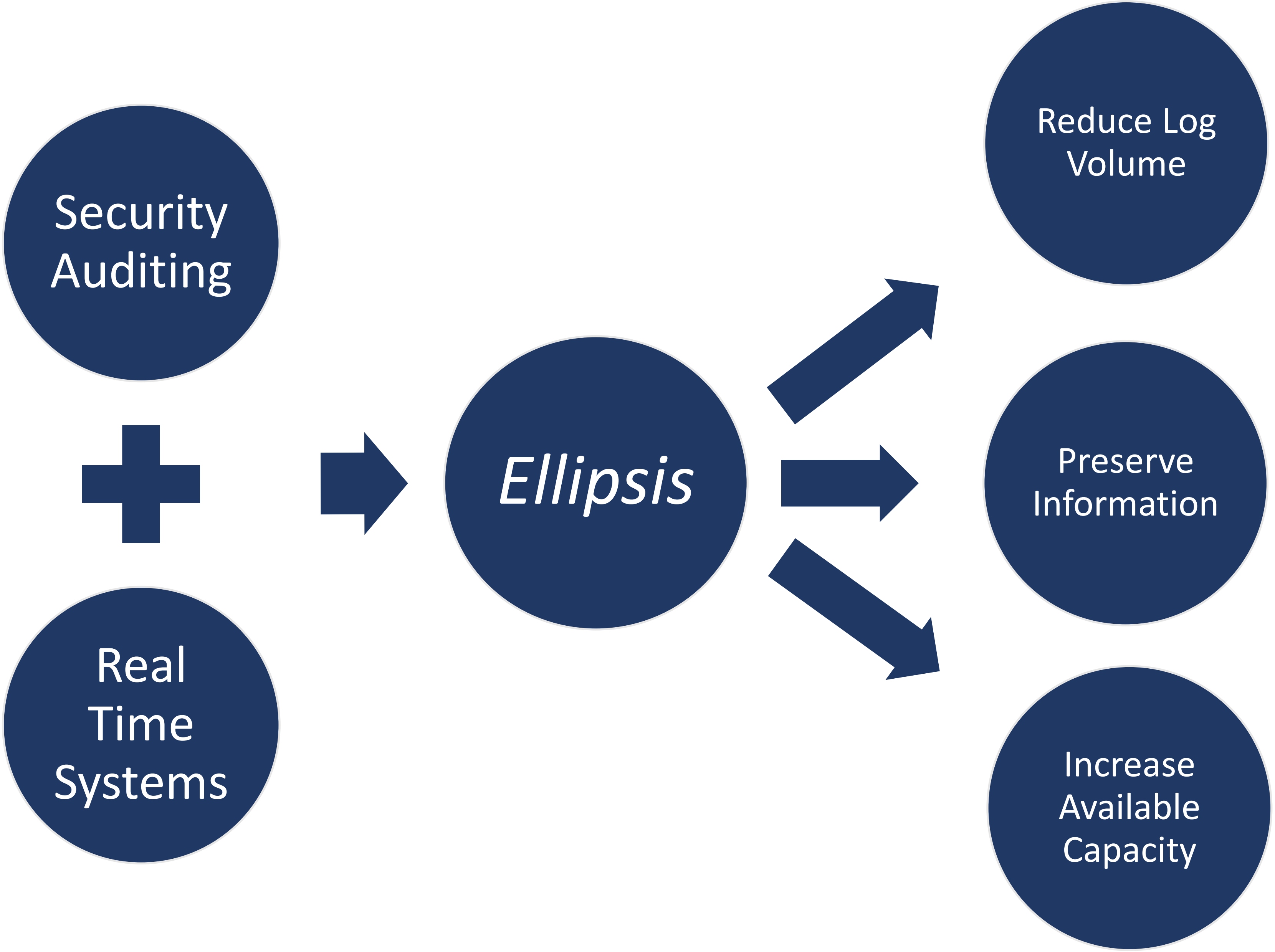
Towards Efficient Auditing for Real-Time Systems

Ayoosh Bansal, Anant Kandikuppa, Chien-Ying Chen, Monowar Hasan,
Adam Bates and Sibin Mohan

UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

WICHITA STATE UNIVERSITY

THE GEORGE WASHINGTON UNIVERSITY
WASHINGTON, DC



Security Auditing

Real Time Systems

Ellipsis

Reduce Log Volume

Preserve Information

Increase Available Capacity

Security Auditing

Security Relevant Events

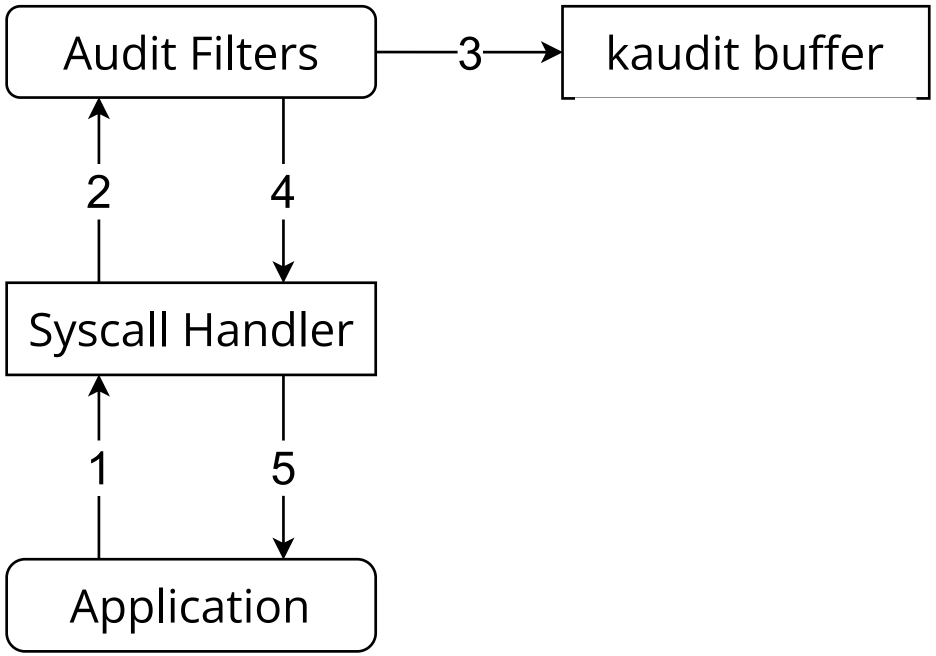
- Generate, process and record

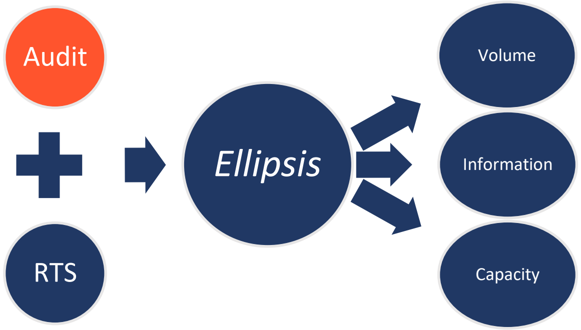
Threats and Incidents

- Forensic analysis
- Data and event provenance

Linux Audit

- System call auditing





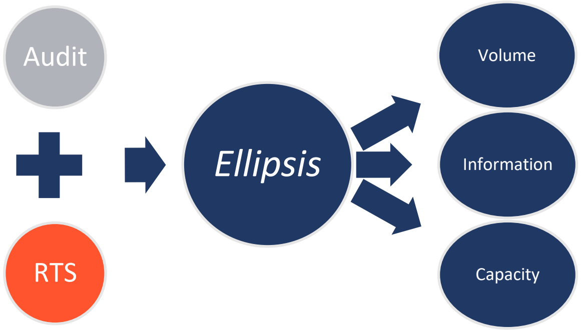
Real-Time Application Structure

Initialization: One time, non-Real-Time

Loop: **Periodic**, Real-Time

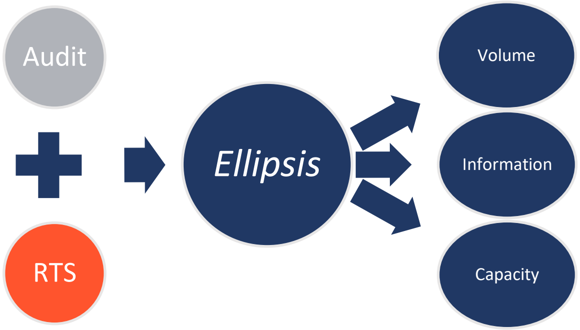
Exit: System shutdown or restart

```
init:       sensor = open()
            actuator = open()

loop:       read(sensor)
            read(sensor)
            compute ()
            write(actuator)
            write(actuator)
            sleep ()

exit:       close(sensor)
            close(actuator)
```



Auditing Real-Time Systems

- RTS are becoming complex and vulnerable
- RTS are part of infrastructure like power grids, which are vulnerable to attacks
- Event recorders in Autonomous Vehicles
- Auditing can help



Challenge: Audit Event Volume

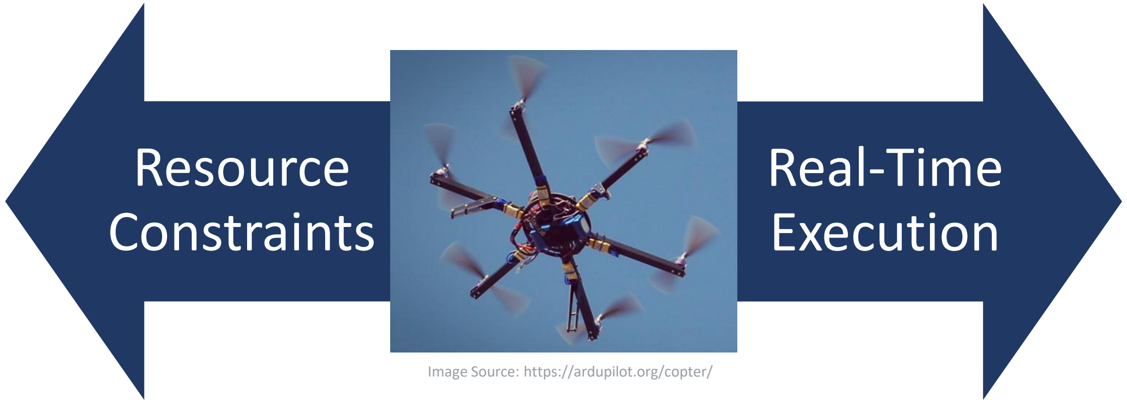
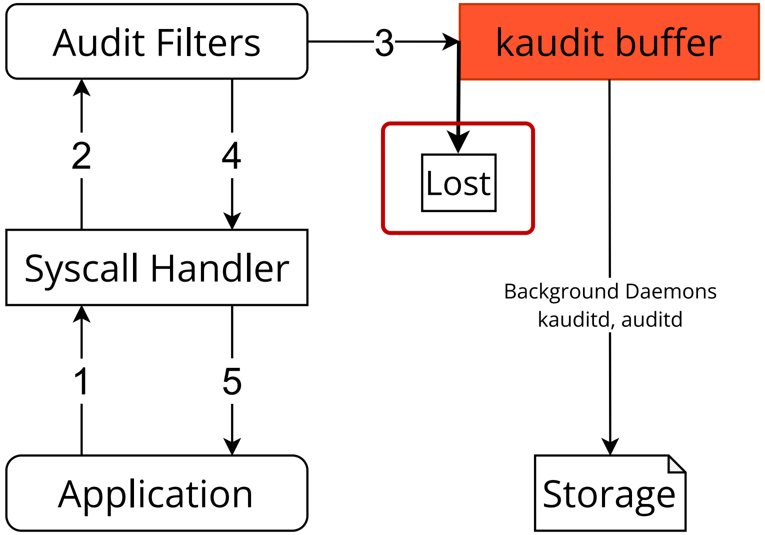
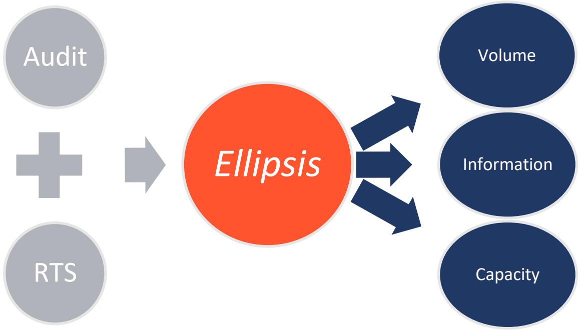


Image Source: https://ardupilot.org/copter/



Repeating Predictable Execution Paths

Comprehensive Analyses



Goals

Reduce audit record volume

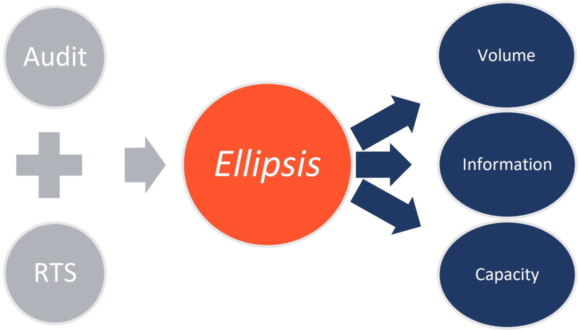
- Linux Audit loses audit records

Provide same security as Linux Audit

- Preserve information

Minimal complexity

- Fully automated processes
- Simple interfaces



Ellipsis

Template: Learn expected behavior

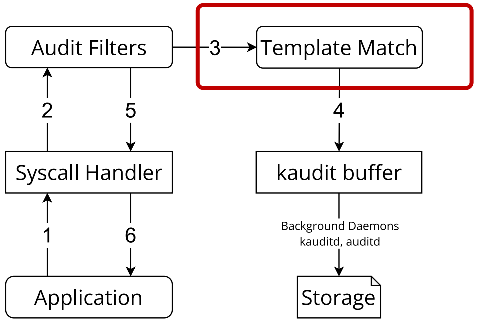
- *e.g.,* read read write write
- template generation automated

Match to template at **runtime**

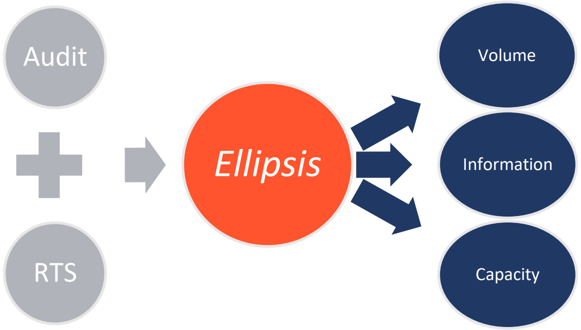
- order, arguments, **timing**
- auditctl extended: load templates, enable Ellipsis

Sequences reduced to **single record**

- 4 → 1
- reduction **before** insertion in buffer



```
init:     sensor = open()
          actuator = open()

loop:     read(sensor)
          read(sensor)
          compute ()
          write(actuator)
          write(actuator)
          sleep ()

exit:     close(sensor)
          close(actuator)
```



Ellipsis

Log retained if any deviation

- Order
- Arguments
- Timing

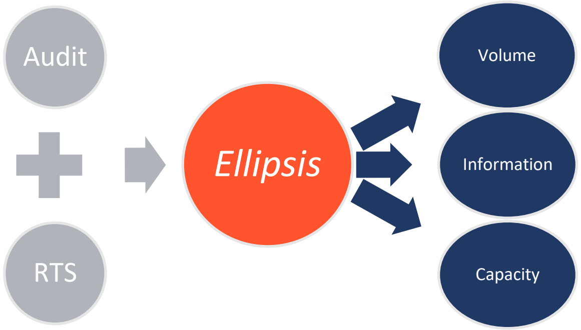
Ellipsis-HP

- Reduce multiple iterations
- ∞ → 1
- Ideally, ~100% reduction

Ellipsis

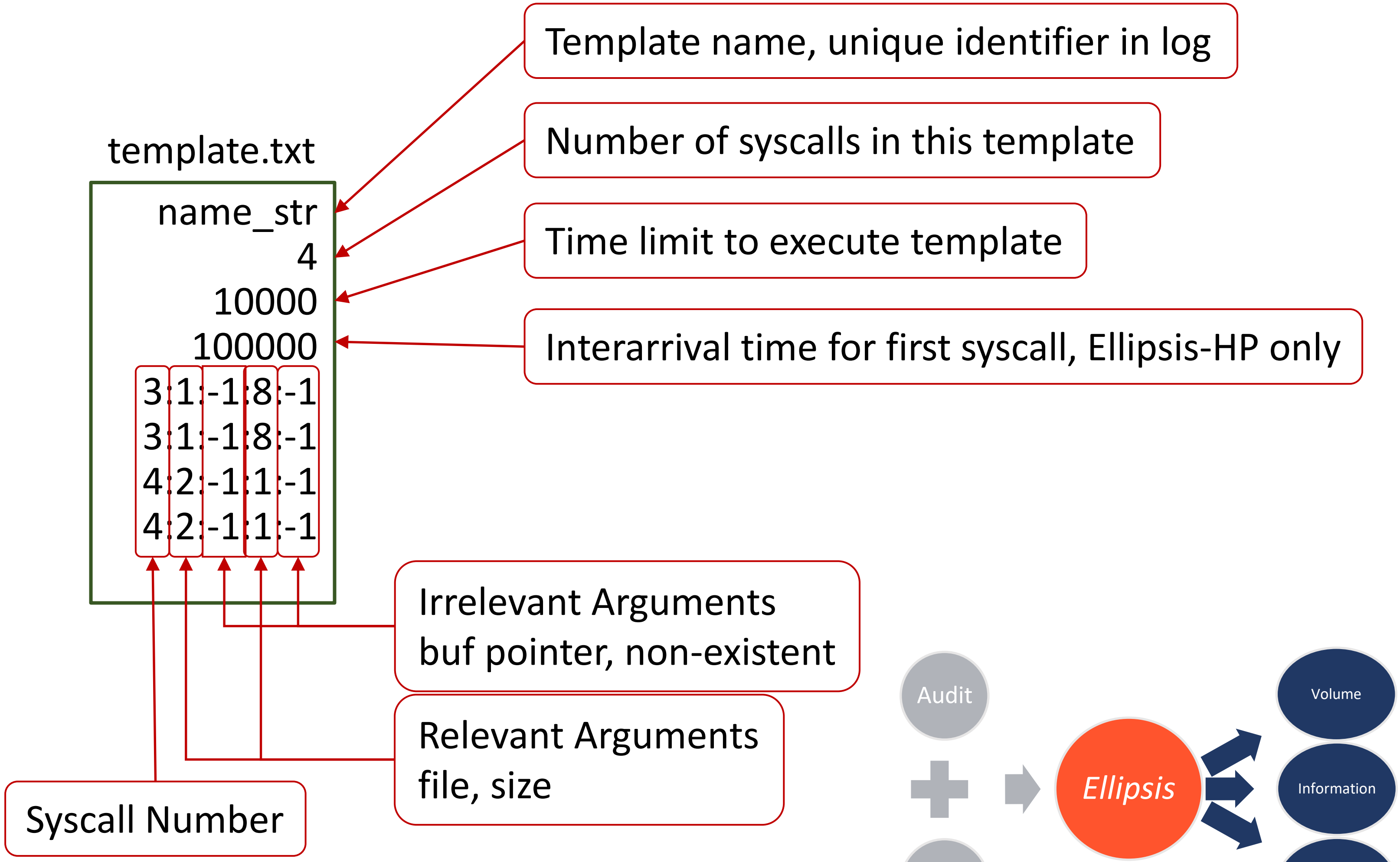
- **End to End** Process
- **Automated** Template Generation
- Runtime **Reduction**

```
init:     sensor = open()
          actuator = open()

loop:     read(sensor)        ┐
          read(sensor)        │
          compute ()          │
          write(actuator)     ├ ∞
          write(actuator)     │
          sleep ()            ┘

exit:     close(sensor)
          close(actuator)
```



Template

```
init:      sensor = open()
           actuator = open()

loop:      read(sensor)
           read(sensor)
           compute ()
           write(actuator)
           write(actuator)
           sleep ()

exit:      close(sensor)
           close(actuator)
```



template.txt

```
 name_str
        4
    10000
   100000
3:1:-1:8:-1
3:1:-1:8:-1
4:2:-1:1:-1
4:2:-1:1:-1
```

Template name, unique identifier in log

Number of syscalls in this template

Time limit to execute template

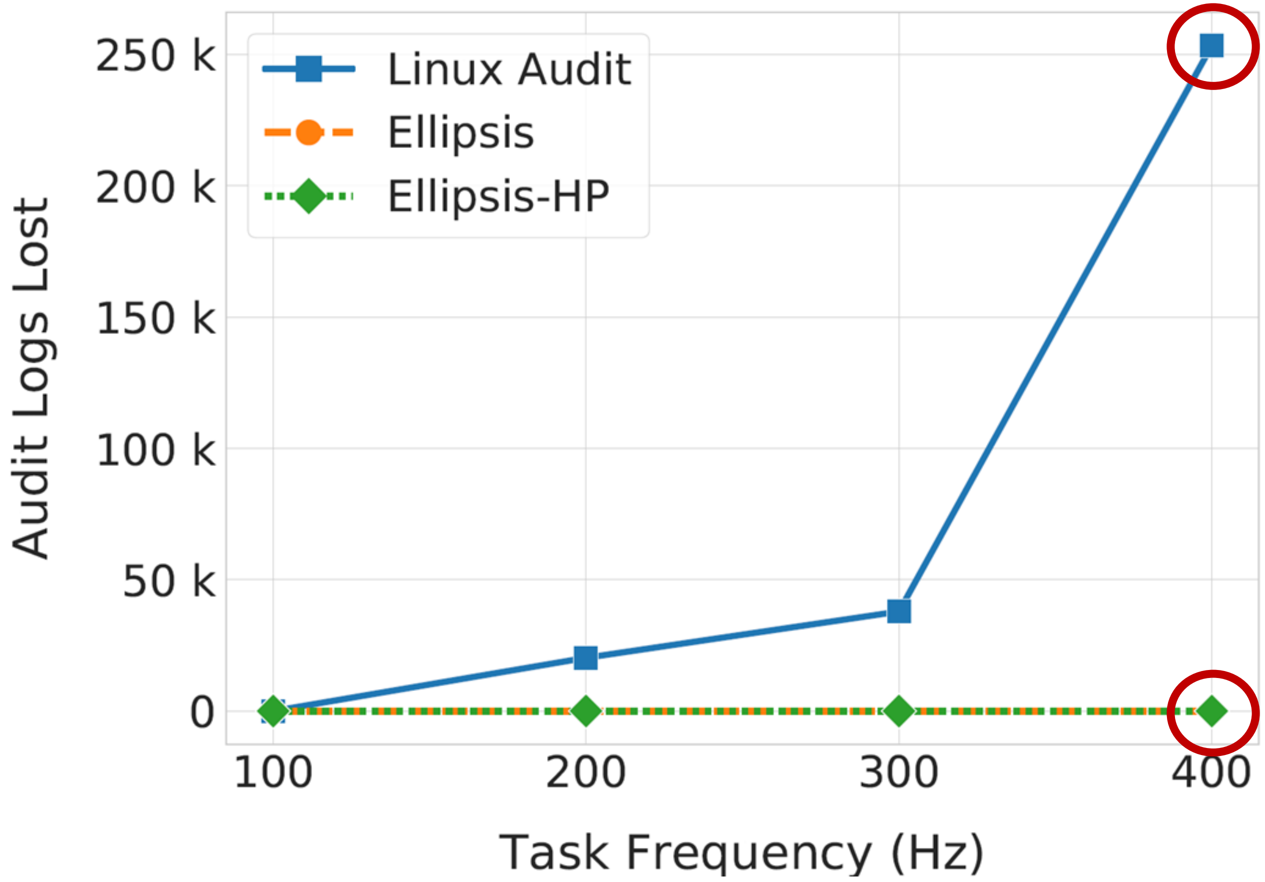
Interarrival time for first syscall, Ellipsis-HP only

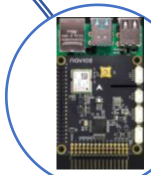
Irrelevant Arguments
buf pointer, non-existent

Relevant Arguments
file, size

Syscall Number

Evaluation | All Events Recorded





Evaluated with ArduPilot on Raspberry Pi 4 + Navio2 Hat

250K Events lost by Linux Audit in 250 seconds.

100% Events recorded by *Ellipsis*

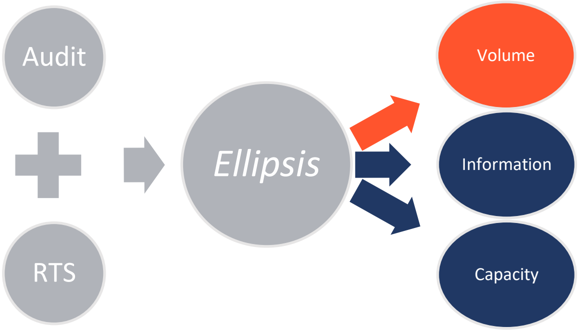
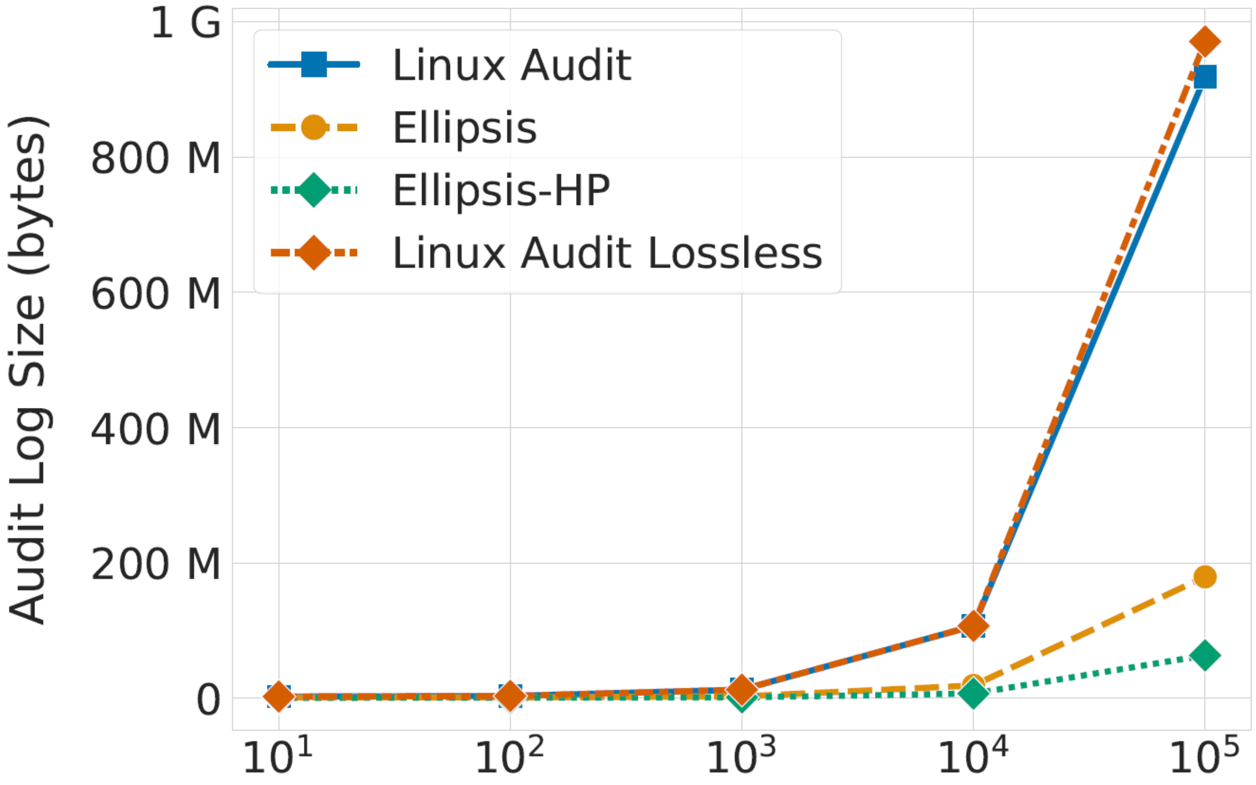


Image Source: https://navio2.emlid.com/

Evaluation | Log Volume Reduction



Iterations

(2.5 ms per iteration)

250 s Total Runtime

93% Less event generated

∞ RTS run for long times

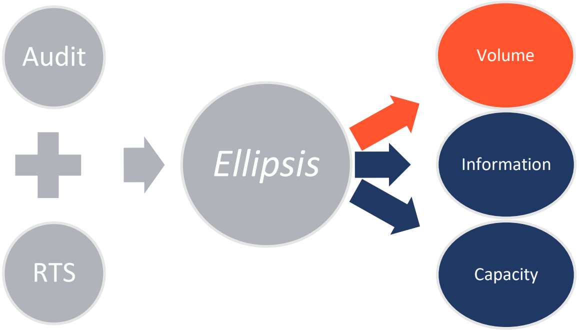


Image Source: https://navio2.emlid.com/

Security Analysis

Information “Lost”

- exact timestamps
- inter-thread order
- irrelevant arguments

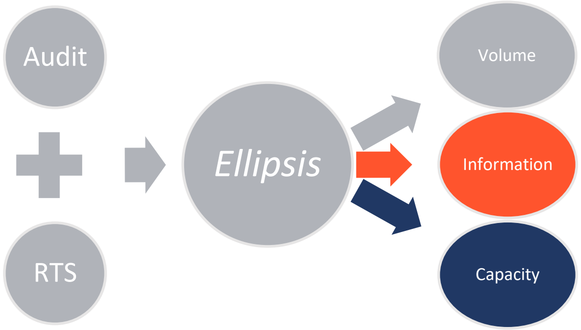
Ellipsis Checks

- template timing
- syscall order
- relevant arguments

Mitigation

- bounded uncertainty
- RTS isolate threads
- administrator defines relevance

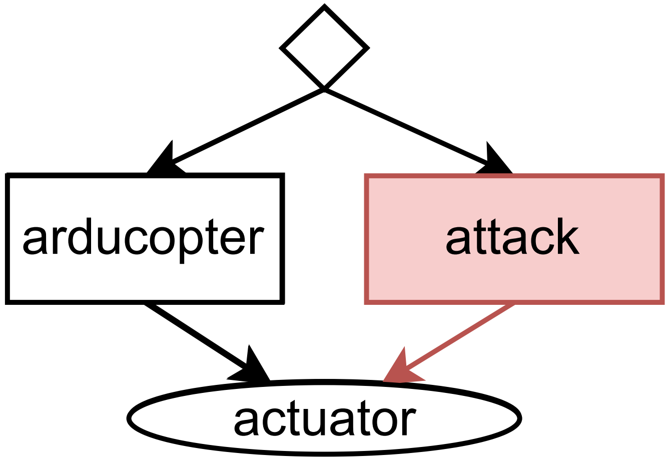
Security Goal
Provide same threat detection as Linux Audit



Security Analysis | Stealthy Attacks

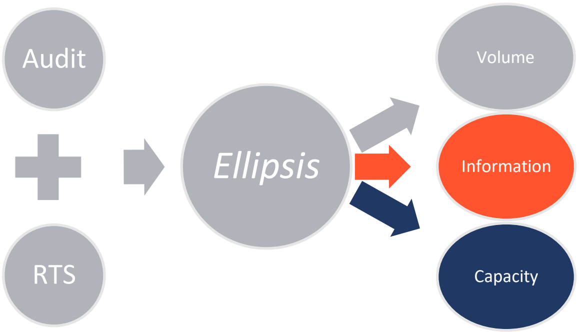
Stealthy Attack

- match template
- evade all checks
- have impact

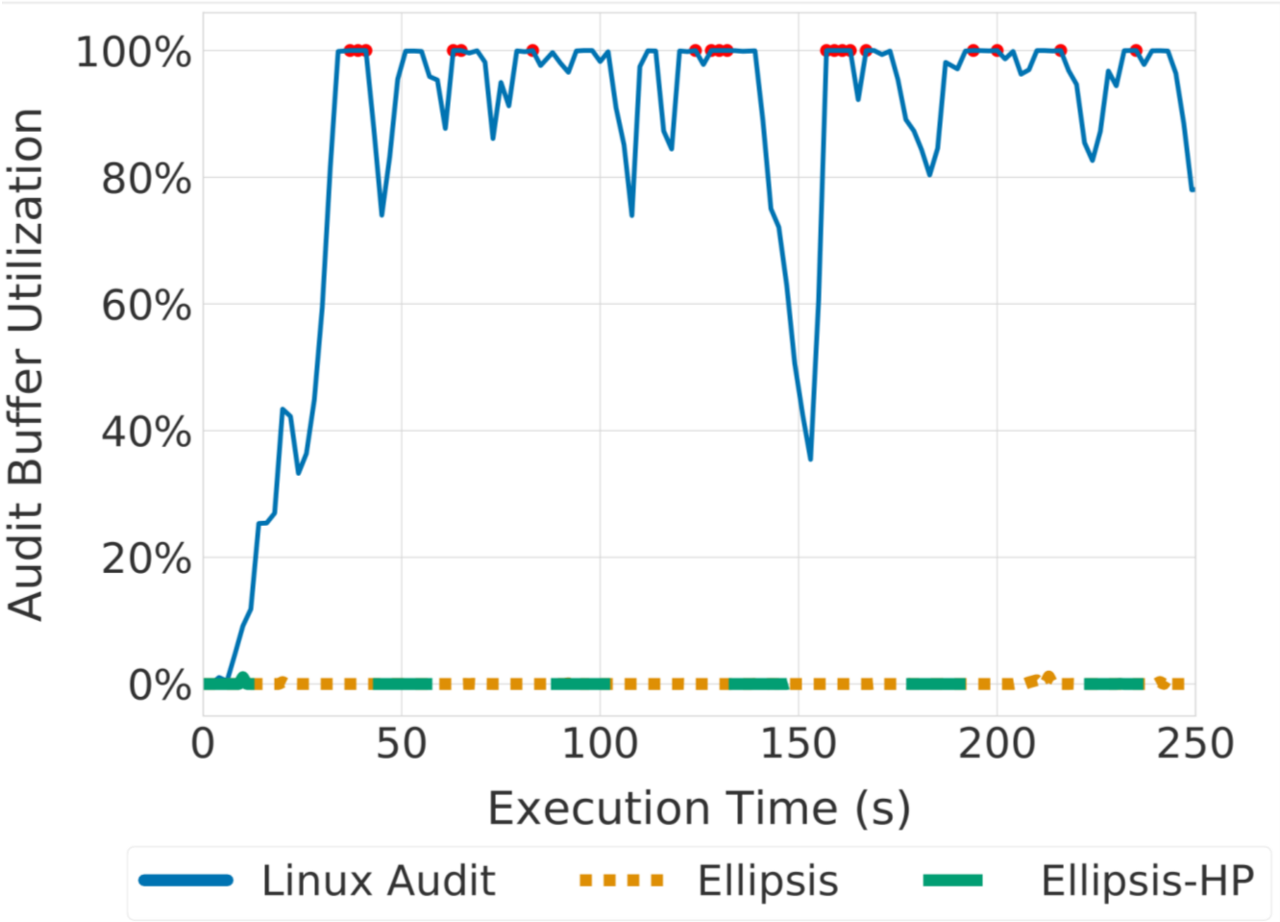


✓

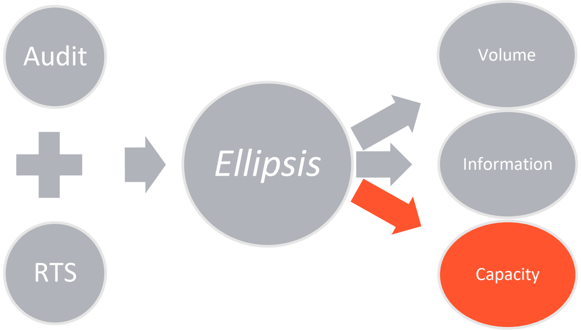
Security Goal
Provide same threat detection as Linux Audit



Evaluation | Increase Available Capacity



- 50K kaudit buffer at max stable size
- ✕ Linux Audit regularly overflows, events lost
- 5% *Ellipsis* buffer usage max
- 95% Capacity freed for security threats and malicious events



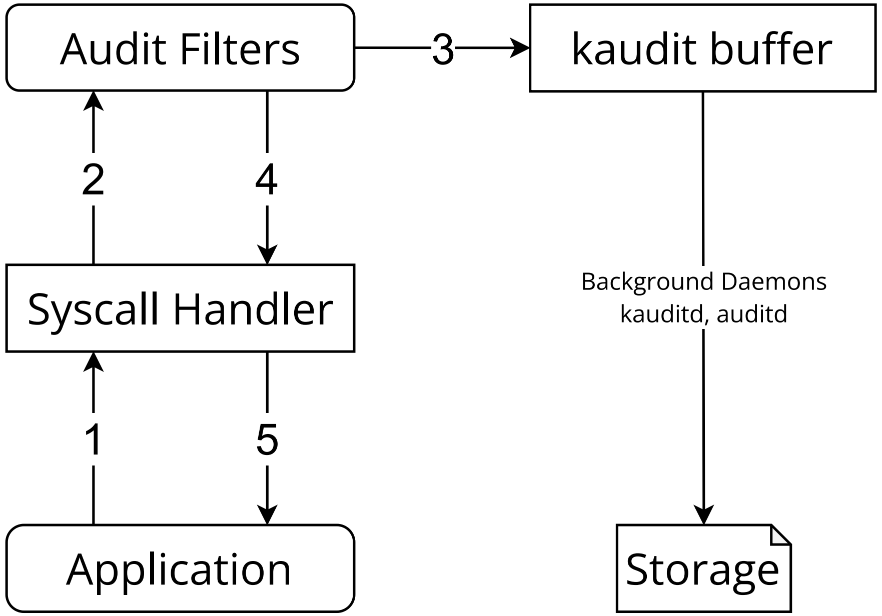
Prior State of the Art

Reduction in **persistent** storage [1,2]

- Events still lost at in **memory** buffer
- Can still compress Ellipsis log

Reduction at source [3,4]

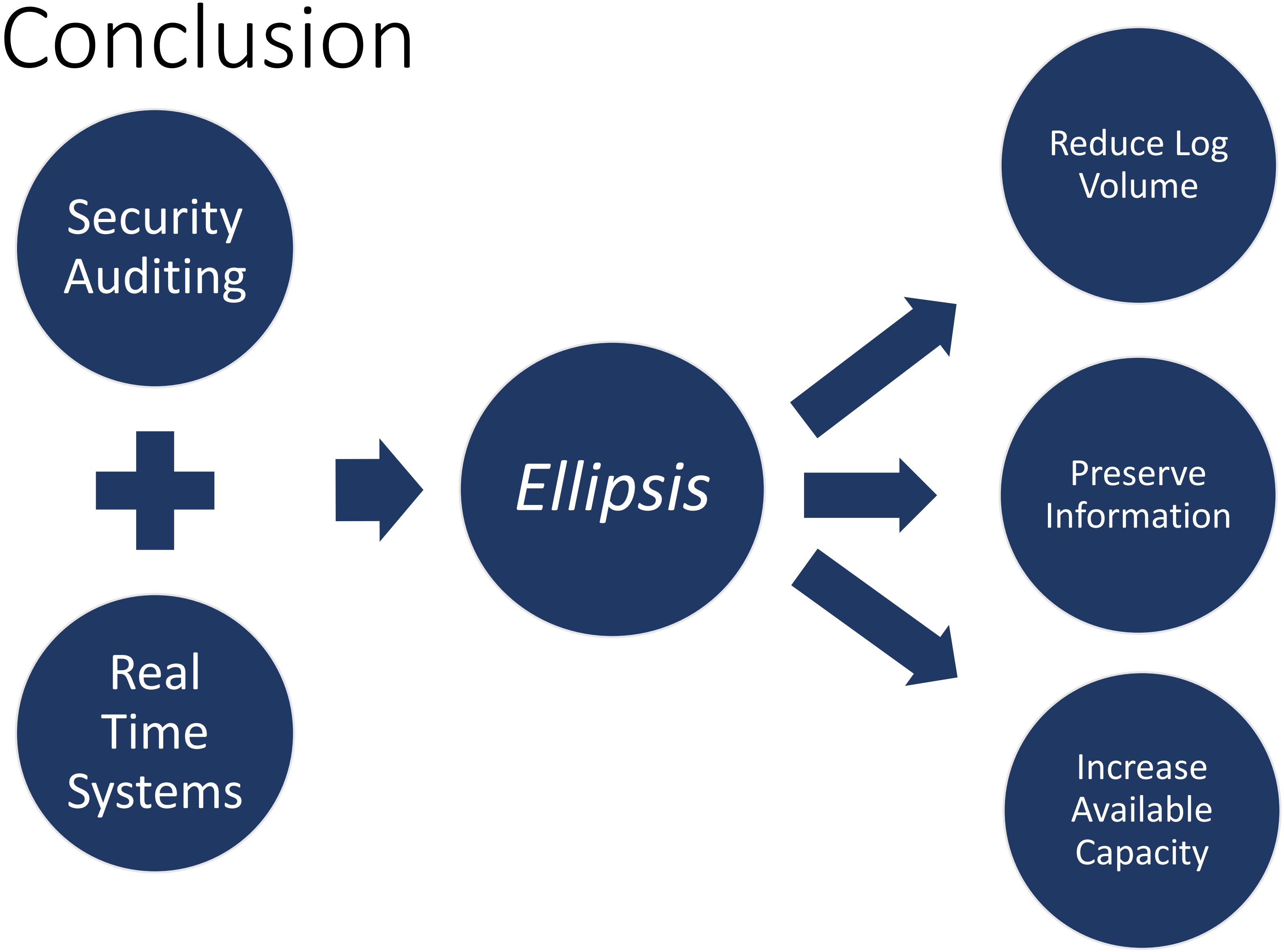
- Relevance different for RTS
  - *e.g.,* the reads and writes to the same files will be reduced by existing techniques
  - the **timing** and **count** are relevant to RTS.



[1] Chen *et.al.,* “Distributed provenance compression,” in ACM SIGMOD 2017.
[2] Ben *et.al.,* “T-tracker: Compressing system audit log by taint tracking,” in ICPADS 2018.
[3] Ma *et.al.,* “ProTracer: Towards Practical Provenance Tracing by Alternating Between Logging and Tainting,” in NDSS 2016.
[4] Ma *et.al.,* “Kernel-supported cost-effective audit logging for causality tracking,” in USENIX ATC 2018.

Thanks: ONR [N00014-17-1-2889] and NSF [CNS 1750024, CNS 1932529, CNS 1955228, CNS 2055127, CNS 2145787, CNS 2152768]

Conclusion



Ellipsis leverages predictability in RTS design and execution

Large reductions in audit event records generated

Information and threat detection same as Linux Audit

More available capacity when malicious events happen