DNS Data Exfiltration Prevention: Kernel-Enforced Endpoint Security

Scalable Framework to Disrupt DNS C2 and Tunneling

Vedang Parasnis

University of Washington

Agenda

- Data Exfiltration / Data Breaches Phases
- DNS attack vectors to exfiltrate data
- Drawbacks of current approaches
- Security Framework Overview
- Kernel Enforced Endpoint Security Architecture (Kernel + Userspace)
- Results and Evaluation
- Protect Linux Kernel from malicious tampered endpoint security eBPF programs
  - Cloud PKI + kernel keyring + BPF LSM + Kernel Datapath

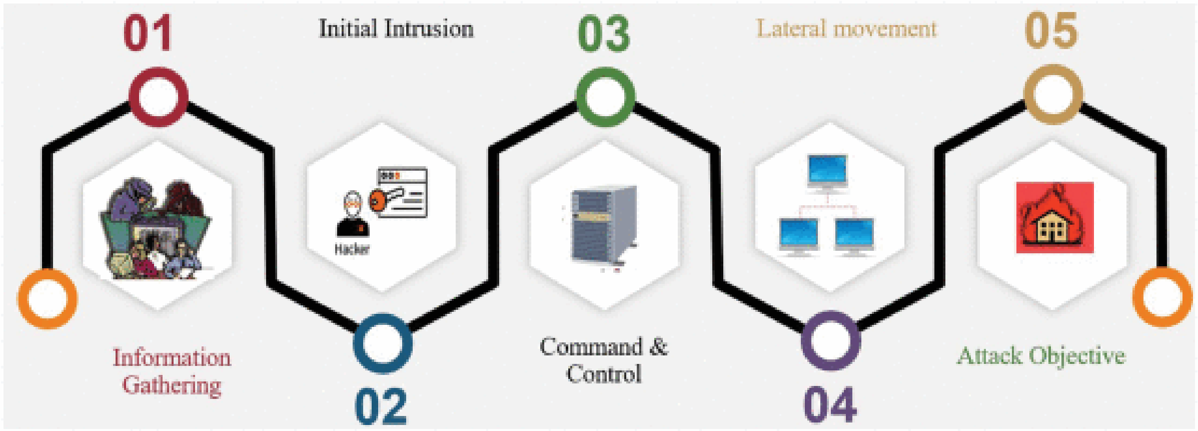
Data Exfiltration / Data Breaches

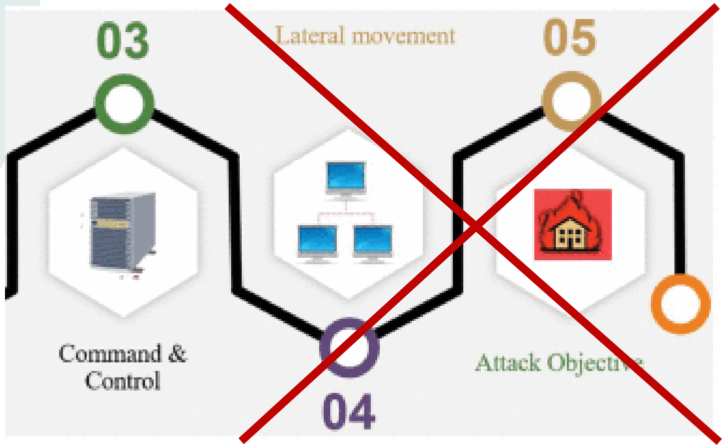
Definition: Unauthorized extraction or transmission of sensitive data from a system

Impact: Reputation, Financial damages

Attack Lifecycle

- 🕵 Information Reconnaissance
- 🔒 Initial Intrusion / Infiltration
- 📡 **Command and Control**
- 🔄 Lateral Movement
- 💣 Command Execution and Data Breaches





Core Defense Strategy

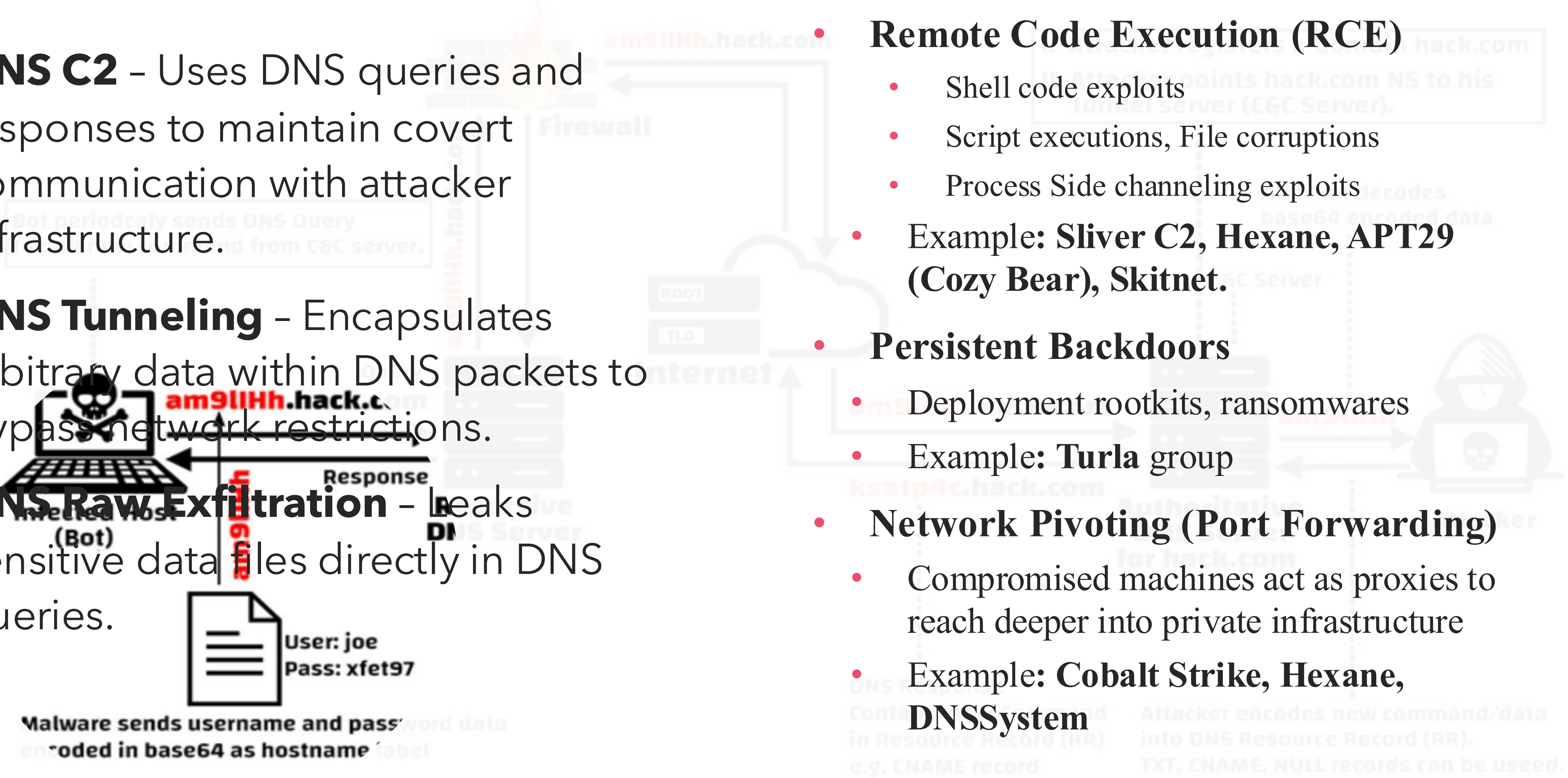
- 🛡 Endpoint Security (EDR / XDR)

DNS Data Exfiltration

DNS C2 – Uses DNS queries and responses to maintain covert communication with attacker infrastructure.

DNS Tunneling – Encapsulates arbitrary data within DNS packets to bypass network restrictions.

DNS Raw Exfiltration – Leaks sensitive data files directly in DNS queries.



- **Remote Code Execution (RCE)**
  - Shell code exploits
  - Script executions, File corruptions
  - Process Side channeling exploits
  - Example: **Sliver C2, Hexane, APT29 (Cozy Bear), Skitnet.**
- **Persistent Backdoors**
  - Deployment rootkits, ransomwares
  - Example: **Turla** group
- **Network Pivoting (Port Forwarding)**
  - Compromised machines act as proxies to reach deeper into private infrastructure
  - Example: **Cobalt Strike, Hexane, DNSSystem**

Existing Approaches

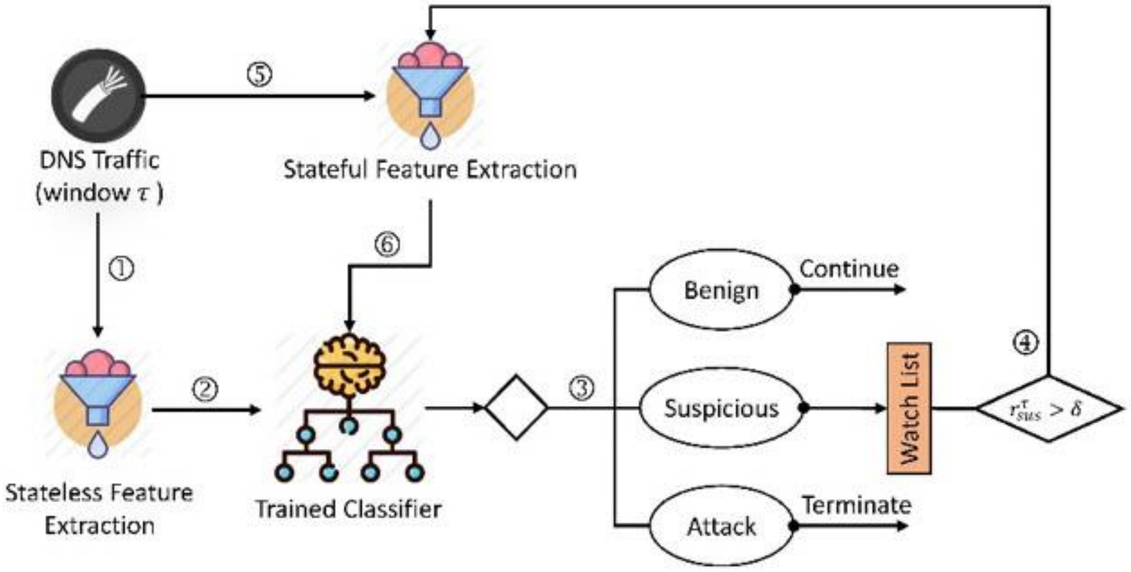
- **Semi-Passive Analysis**
  - DNS Exfiltration Security as Middleware
- **Passive Analysis**
  - Anomaly Detection
  - Threat Signatures, Domain Reputation scoring

Existing Approaches – Passive Analysis

- **Anomaly Detection:**
  - **Traffic Behavior Analysis**
    - DNS Passive Traffic Volume Analysis
    - DNS Passive Traffic timing Statistical Analysis
  - **Machine Learning-based Threat Intelligence**
    - Uses machine learning models to identify traffic anomalies.
- **Threat Signatures:**
  - DNS Domain Scoring
  - Malicious domain signature

Stateless Features – Lexical Analysis

Stateful Features – Statistical Analysis



Issues with current approaches

- **Slow Detection → High Dwell Time → More Damage**
- **Extremely slow to Advanced C2 Attacks**
  - More Damage if C2 infrastructure employs multiplayer mode (Botnet of C2 server exploiting scaled environments)
- **Dynamic Threat Patterns**:
  - Varying Throughput
  - Slow and Stealthy Rate
  - Kernel Encapsulated Traffic
  - Port Obfuscation
- **Centralized monitoring and analysis systems don’t scale**
- **Ineffective over IP Masquerading & Domain Generation Algorithms**

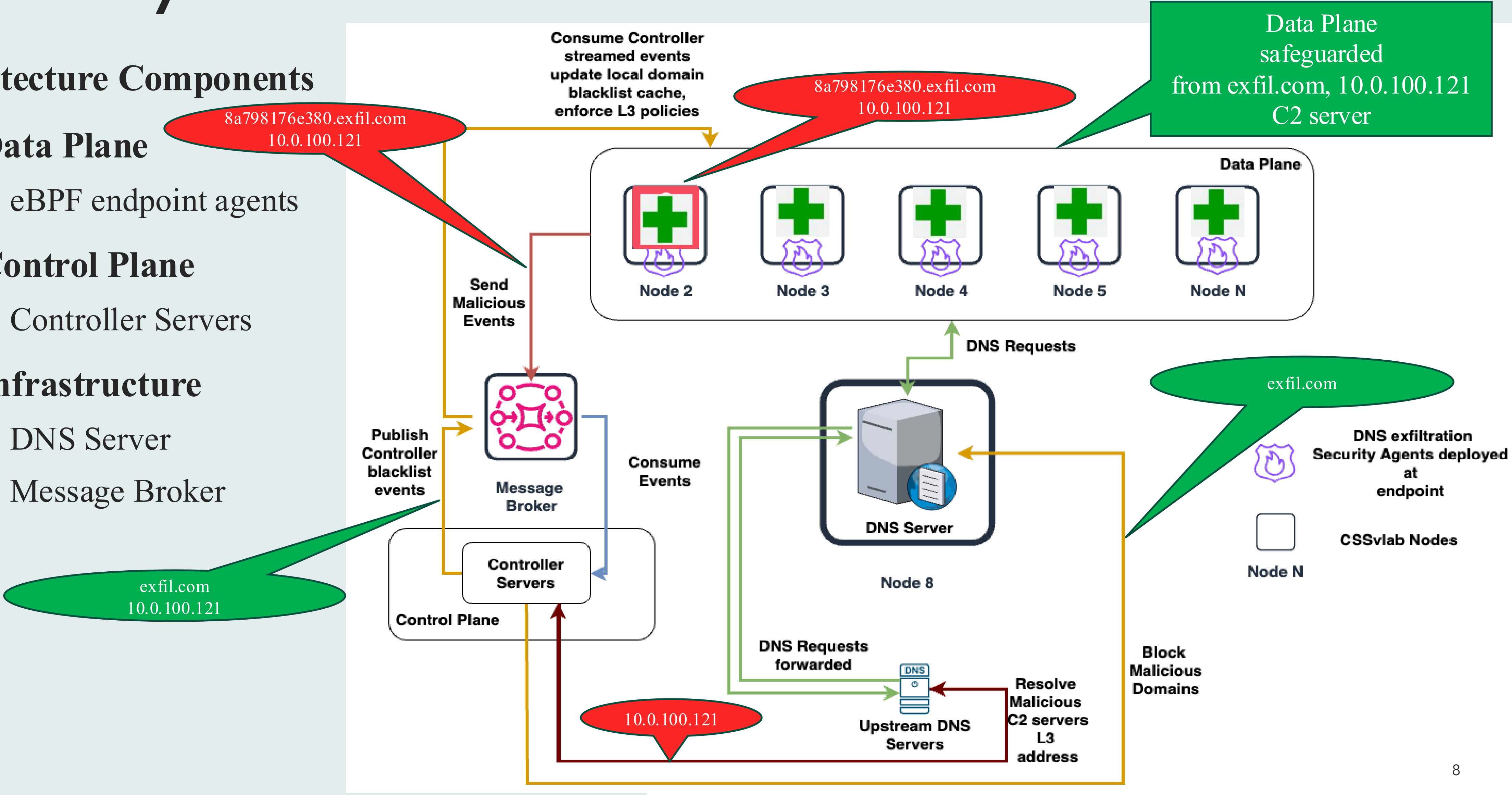
Solution:

Real-time, proactive enforcement at Ring 0 — inside the kernel, where no userland evasion can hide.

Security Framework Architecture Overview

Architecture Components

- **Data Plane**
  - eBPF endpoint agents
- **Control Plane**
  - Controller Servers
- **Infrastructure**
  - DNS Server
  - Message Broker



Data Plane safeguarded from exfil.com, 10.0.100.121 C2 server
Consume Controller streamed events update local domain blacklist cache, enforce L3 policies
8a798176e380.exfil.com 10.0.100.121
8a798176e380.exfil.com 10.0.100.121
Data Plane
Node 2
Node 3
Node 4
Node 5
Node N
Send Malicious Events
DNS Requests
exfil.com
Publish Controller blacklist events
Message Broker
Consume Events
DNS Server
DNS exfiltration Security Agents deployed at endpoint
CSSvlab Nodes
Node N
Node 8
exfil.com 10.0.100.121
Controller Servers
Control Plane
DNS Requests forwarded
Block Malicious Domains
Resolve Malicious C2 servers L3 address
10.0.100.121
DNS
Upstream DNS Servers

Security Framework Goals

Disrupt DNS covert C2 channel attacks, data exfiltration.

Implement in-kernel deep packet inspection and enforcement to block all forms of DNS exfiltration channels.

AI-Assisted Threat Detection

Use deep learning in userspace to detect advanced obfuscated exfiltration payloads with high accuracy aiding kernel network enforcements.

Malicious Process Aware Active Response (Threat-Hunt and Kill)

Link exfiltration attempt to parent process and kill implants processes, preventing lateral movement and further damage.

Dynamic Cross-Layer Policy Enforcement

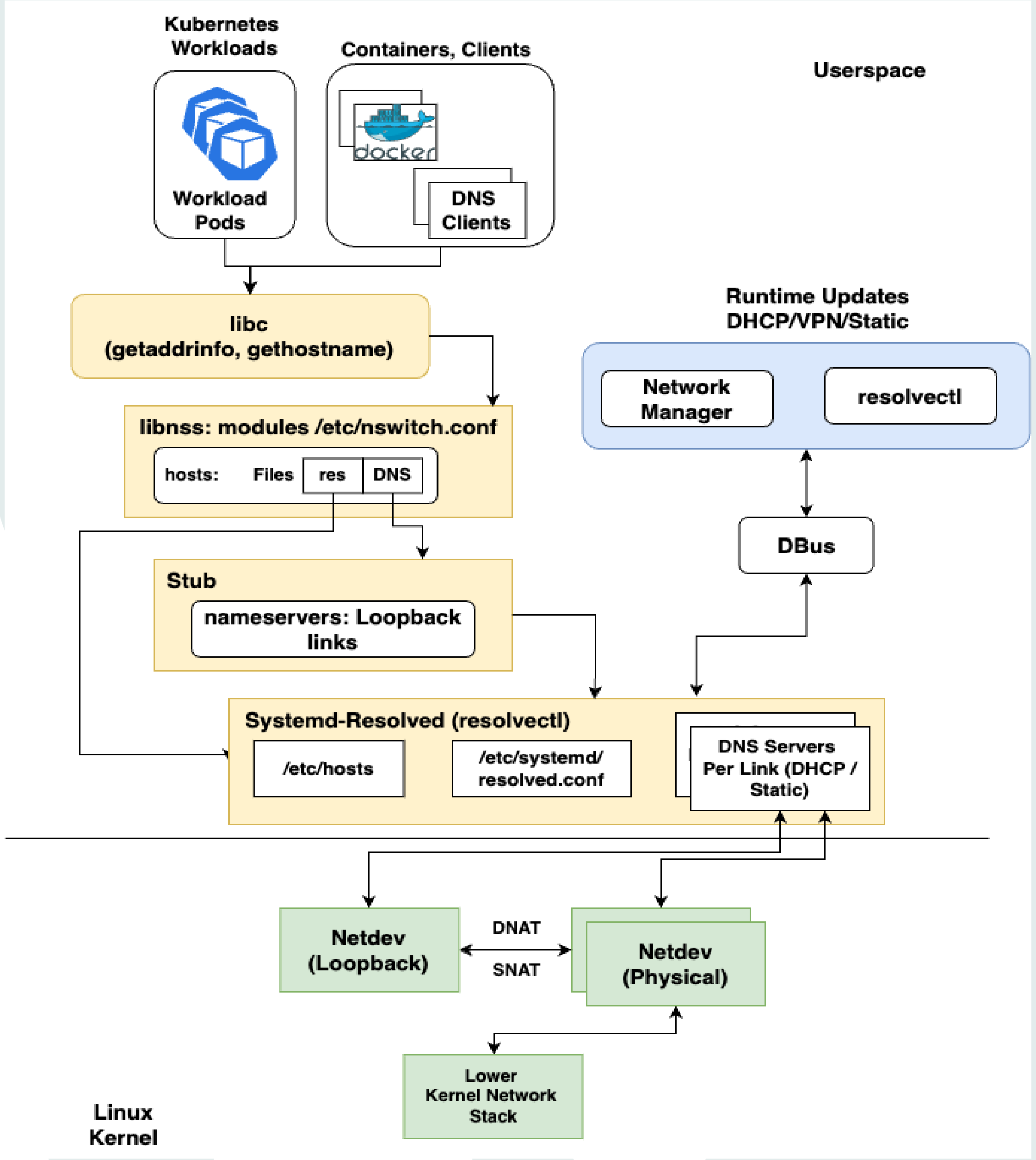
Enforce in-kernel L3 network policies adaptively and domain blacklisting on DNS server to combat DGA.

Scalable Multi-Cloud Deployment

Ensure framework's horizontal scales for real-world production cloud environments.

Systemd-Resolved

- **Userspace**
  - Libc (dns_utils)
  - Libnss (nss modules (nss-dns, nss-myhostname)
  - Systemd-resolved (resolvectl)
  - System Daemons
    - Network Manager (DHCP)
    - Dbus
- **Kernel**
  - Network Stack each netdev (east-west, north-south traffic)


Kubernetes Workloads
Workload Pods
Containers, Clients
docker
DNS Clients
Userspace
libc (getaddrinfo, gethostname)
Runtime Updates DHCP/VPN/Static
Network Manager
resolvectl
libnss: modules /etc/nswitch.conf
hosts: Files res DNS
DBus
Stub
nameservers: Loopback links
Systemd-Resolved (resolvectl)
/etc/hosts
/etc/systemd/ resolved.conf
DNS Servers Per Link (DHCP / Static)
Netdev (Loopback)
DNAT
SNAT
Netdev (Physical)
Lower Kernel Network Stack
Linux Kernel

Kernel Enforced Endpoint Security

Agent based Endpoint Security

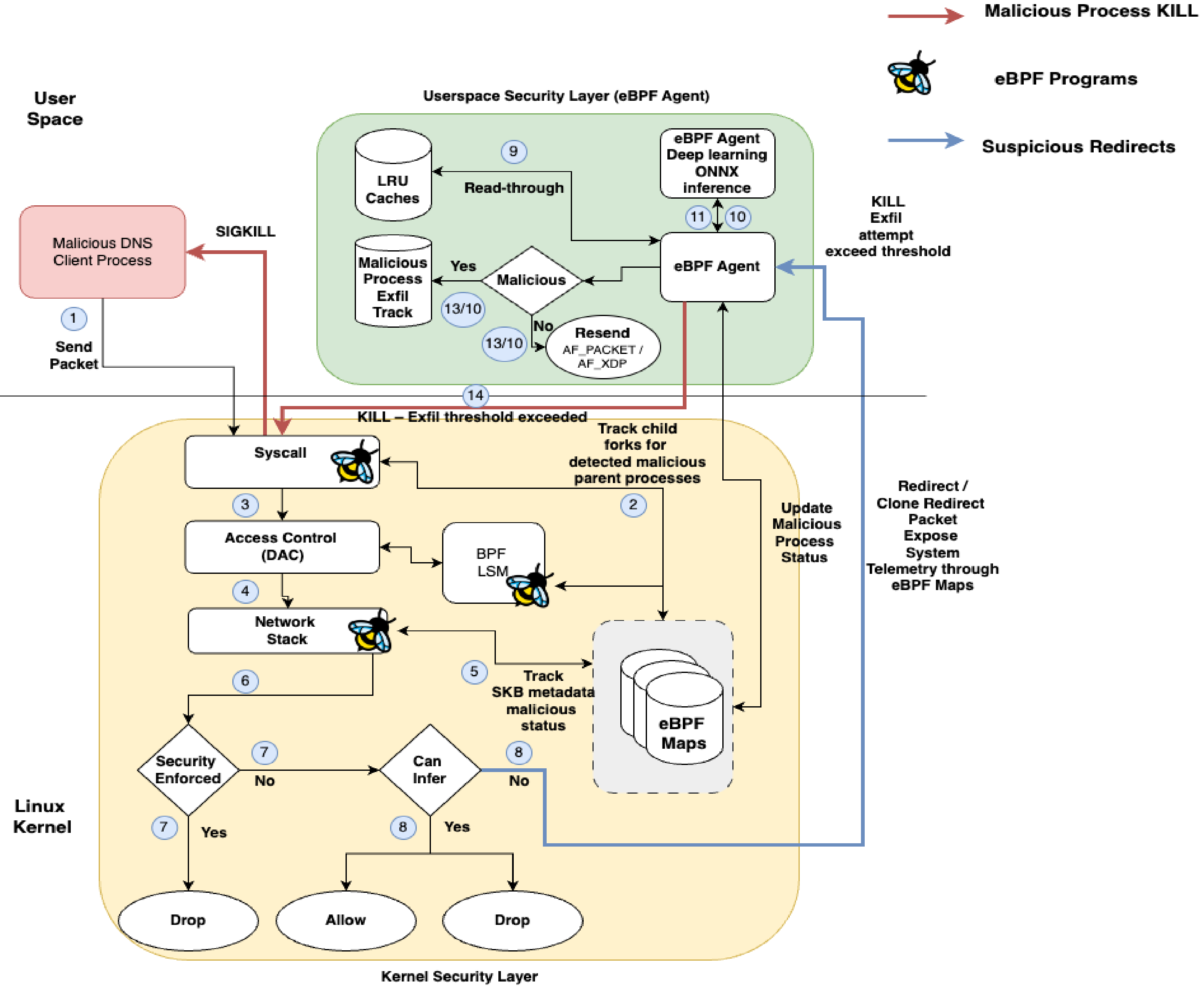
Continuous Security Enforcement Event Loop

Userspace

- eBPF Agent
- eBPF Agent LRU Caches
- ONNX Quantized Deep Learning Model
- Kernel malicious metrics export (Prometheus)

Linux Kernel

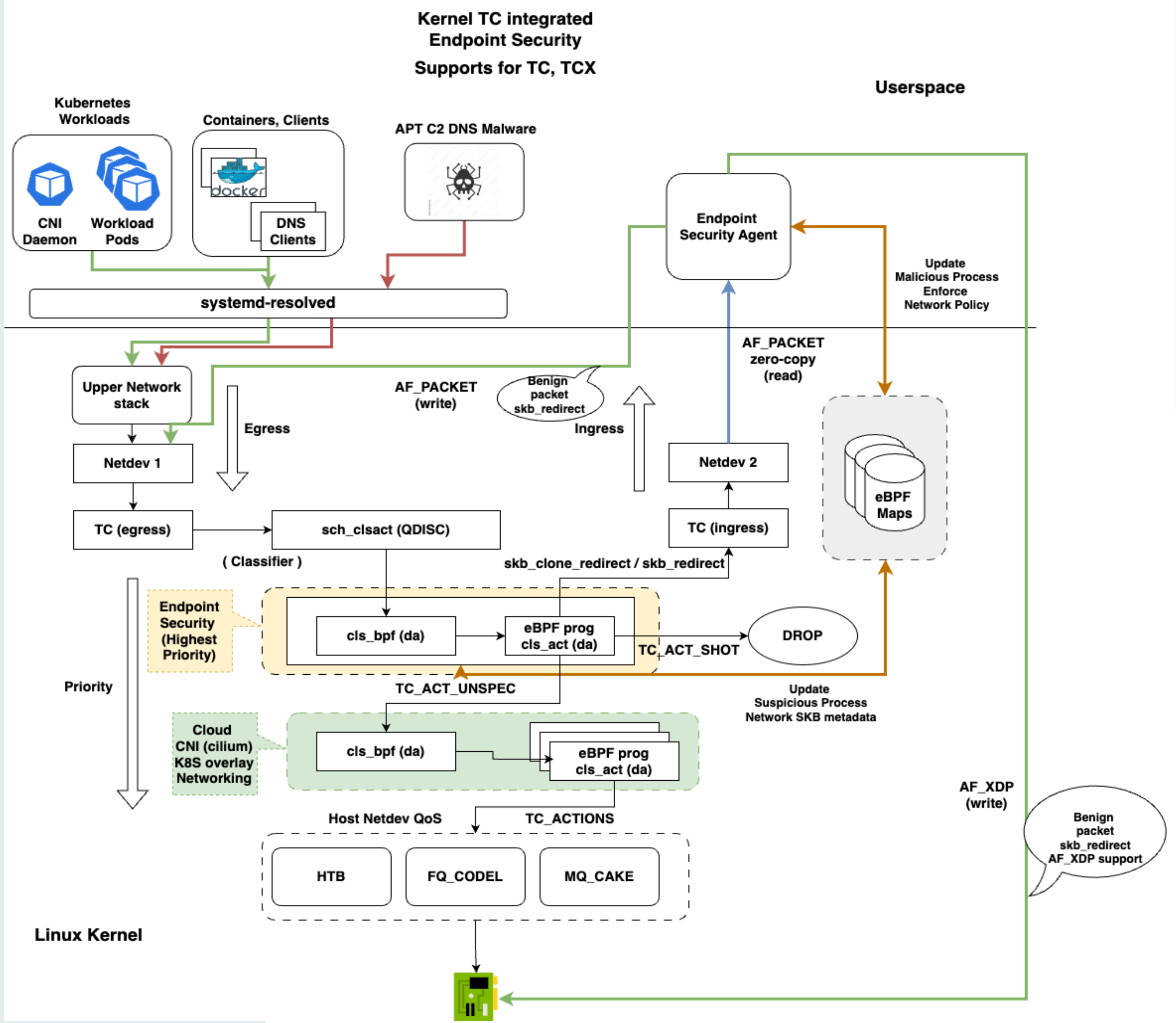
- **Inference Unix Domain Sockets**
- **eBPF Ring Buffers (malicious events)**
- **Network Stack (eBPF programs)**
  - Socket Layer
  - Traffic Control
- **Access Control Layer (eBPF programs)**
  - Security Modules (eBPF LSM)
  - Syscall (eBPF Tracepoints)



Malicious Process KILL
eBPF Programs
Suspicious Redirects
User Space
Userspace Security Layer (eBPF Agent)
LRU Caches
9
Read-through
eBPF Agent Deep learning ONNX inference
11
10
Malicious DNS Client Process
SIGKILL
Malicious Process Exfil Track
Yes
Malicious
eBPF Agent
KILL Exfil attempt exceed threshold
13/10
No
13/10
Resend AF_PACKET / AF_XDP
1
Send Packet
14
KILL – Exfil threshold exceeded
Syscall
Track child forks for detected malicious parent processes
2
3
Access Control (DAC)
BPF LSM
Update Malicious Process Status
Redirect / Clone Redirect Packet Expose System Telemetry through eBPF Maps
4
Network Stack
5
Track SKB metadata malicious status
6
eBPF Maps
Security Enforced
7
No
Can Infer
8
No
Linux Kernel
7
Yes
8
Yes
Drop
Allow
Drop
Kernel Security Layer

Kernel Datapath Enforcement Layer

- Sockets
- TCP/IP Stack
- Netfilter
- Traffic Control (QoS)
- Network Drivers (XDP)



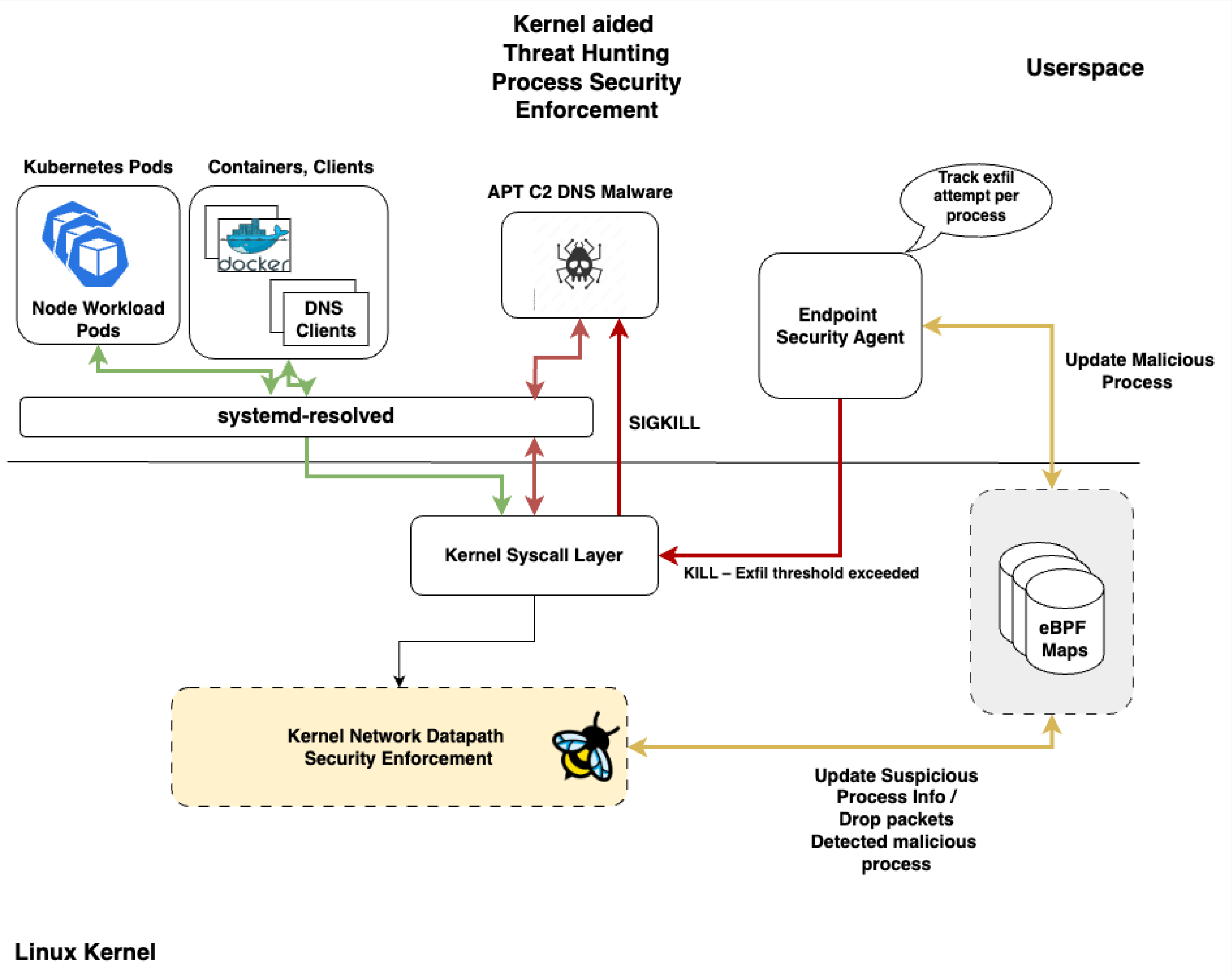
Process Enforcement Layer

Userspace

- Send SIGKILL to malicious process

Linux Kernel

- Kills the malicious implant instructed by userspace endpoint security agent.



Kernel aided Threat Hunting Process Security Enforcement
Userspace
Kubernetes Pods
Node Workload Pods
Containers, Clients
docker
DNS Clients
APT C2 DNS Malware
Track exfil attempt per process
Endpoint Security Agent
Update Malicious Process
systemd-resolved
SIGKILL
Kernel Syscall Layer
KILL – Exfil threshold exceeded
eBPF Maps
Kernel Network Datapath Security Enforcement
Update Suspicious Process Info / Drop packets Detected malicious process
Linux Kernel

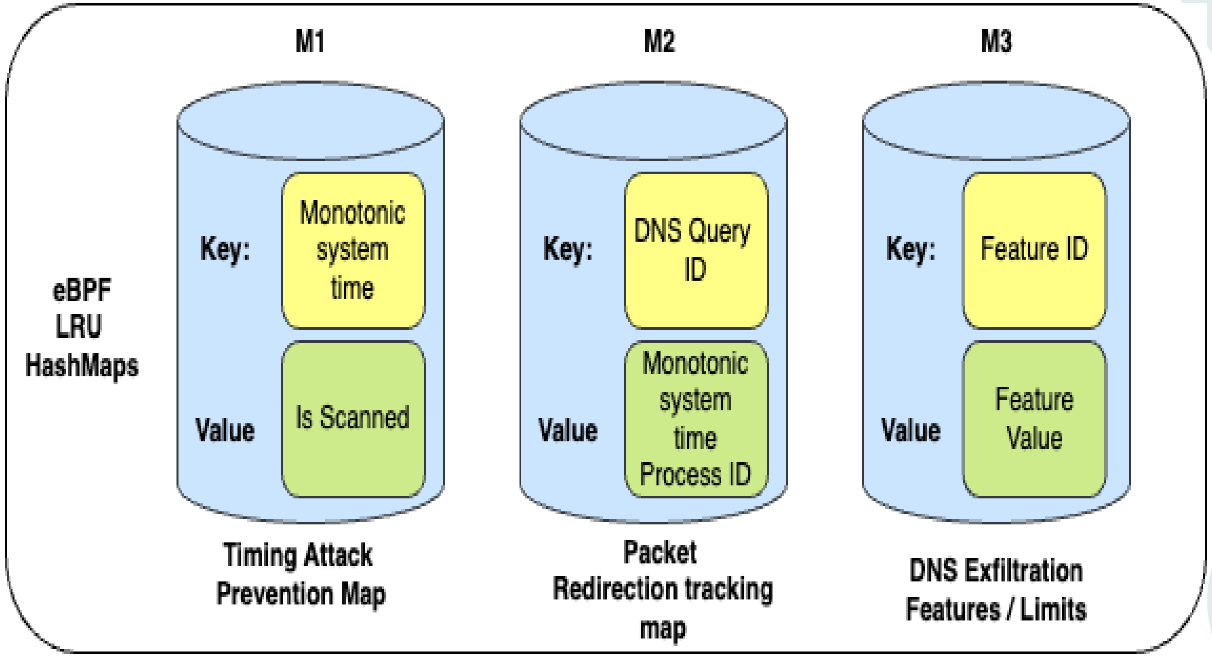
eBPF Agent Operations Modes

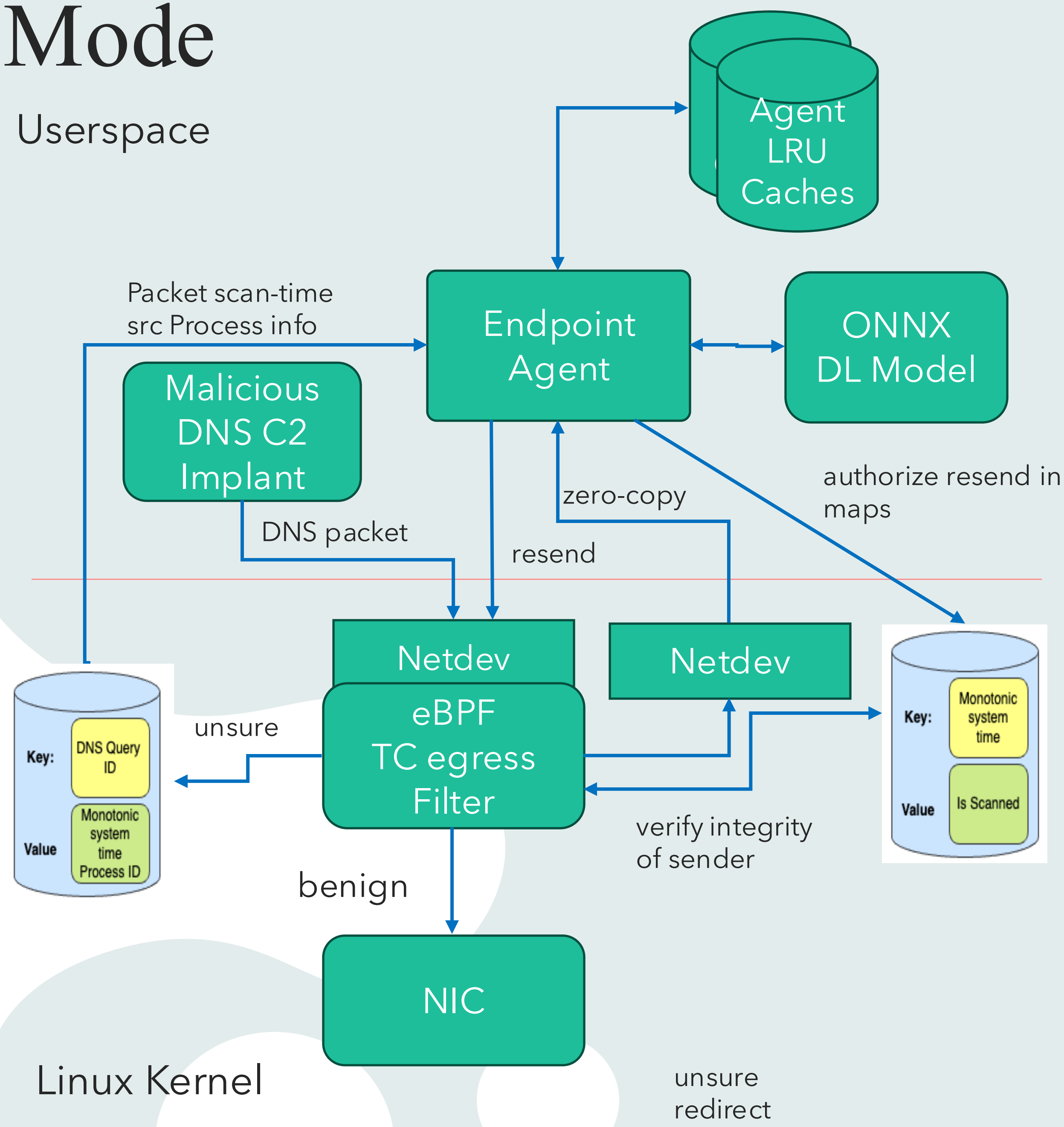
eBPF Agents in Data Plane handle DNS exfiltration over UDP

Mode	Goal	Requirement	Security Enforcement Process
Strict Enforcement Active Mode	Kill C2 Implants, ensure zero data loss and C2 command execution.	DNS Traffic over UDP ports (53, 5353,5355), for encapsulated and non-encapsulated traffic.	• **Kernel**: Live Redirects suspicious DNS packets to userspace. • **Userspace** Trace malicious process exfiltration count and terminates it, resend benign packets.
Process-Aware Passive Threat Hunt Mode	Kill C2 Implants, ensure negligible data loss and minimal C2 command execution.	DNS Traffic over random UDP ports.	• **Kernel:** Allow suspicious traffic passthrough. In Kernel start threat hunting process tied to malicious DNS packets. • **Userspace:** Trace malicious process and terminates it.

Strict Enforcement Active Mode

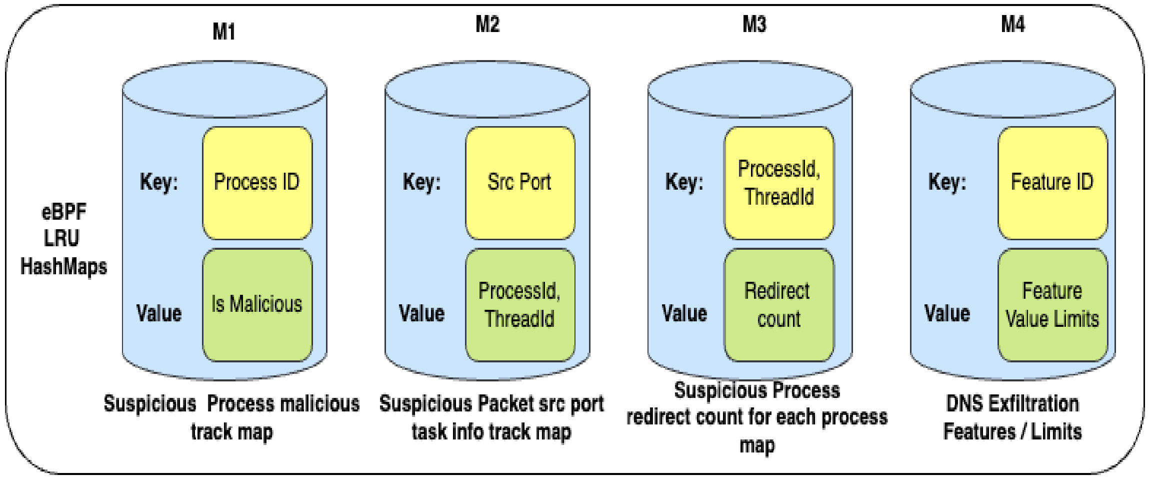
- **eBPF program deep parse of suspicious DNS packets from SKB**
- **Real-time verdict from kernel DPI eBPF program**
- **Userspace DL model aids classification**
- **eBPF zero-trust checks on resend timing**
- **Per-process exfil attempts tracked in userspace**


eBPF
LRU
HashMaps
M1
Key:
Monotonic system time
Value
Is Scanned
Timing Attack Prevention Map
M2
Key:
DNS Query ID
Value
Monotonic system time Process ID
Packet Redirection tracking map
M3
Key:
Feature ID
Value
Feature Value
DNS Exfiltration Features / Limits

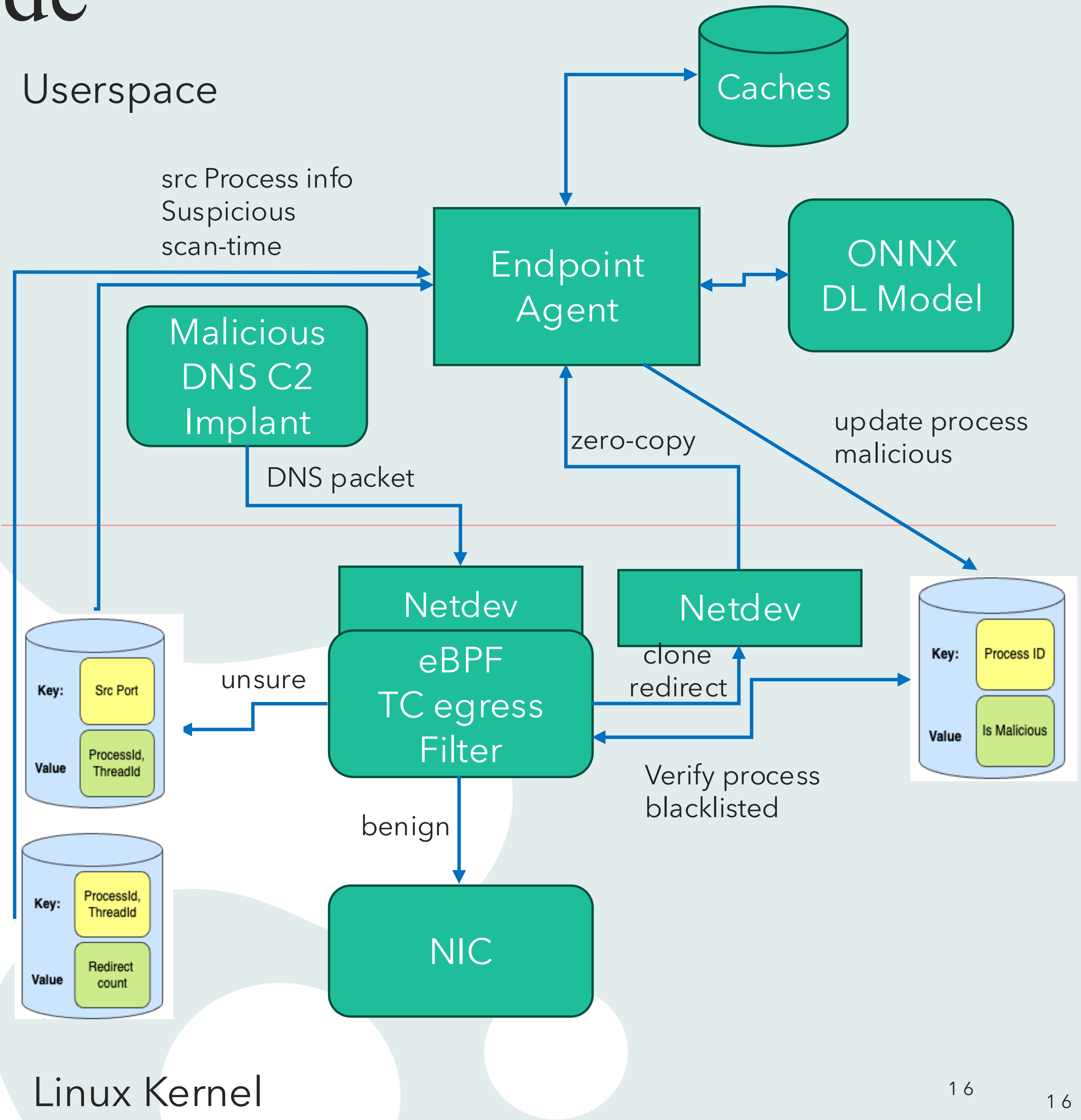


Process-Aware Passive Mode

- **eBPF kernel program deep parses DNS from skb**
- **Suspicious packets cloned to userspace**
- **DL model classifies and tags process**
- **Malicious process flagged and eBPF maps updated**
- **eBPF begins drop packet from malicious process + clone for exfiltration attempt telemetry**



eBPF LRU HashMaps
M1
Key: Process ID
Value Is Malicious
Suspicious Process malicious track map
M2
Key: Src Port
Value ProcessId, ThreadId
Suspicious Packet src port task info track map
M3
Key: ProcessId, ThreadId
Value Redirect count
Suspicious Process redirect count for each process map
M4
Key: Feature ID
Value Feature Value Limits
DNS Exfiltration Features / Limits



DNN based DNS Data Obfuscation Detection

- Model Architecture ONNX
- Sample Malicious Data
- Graph
- Input Features
- Features

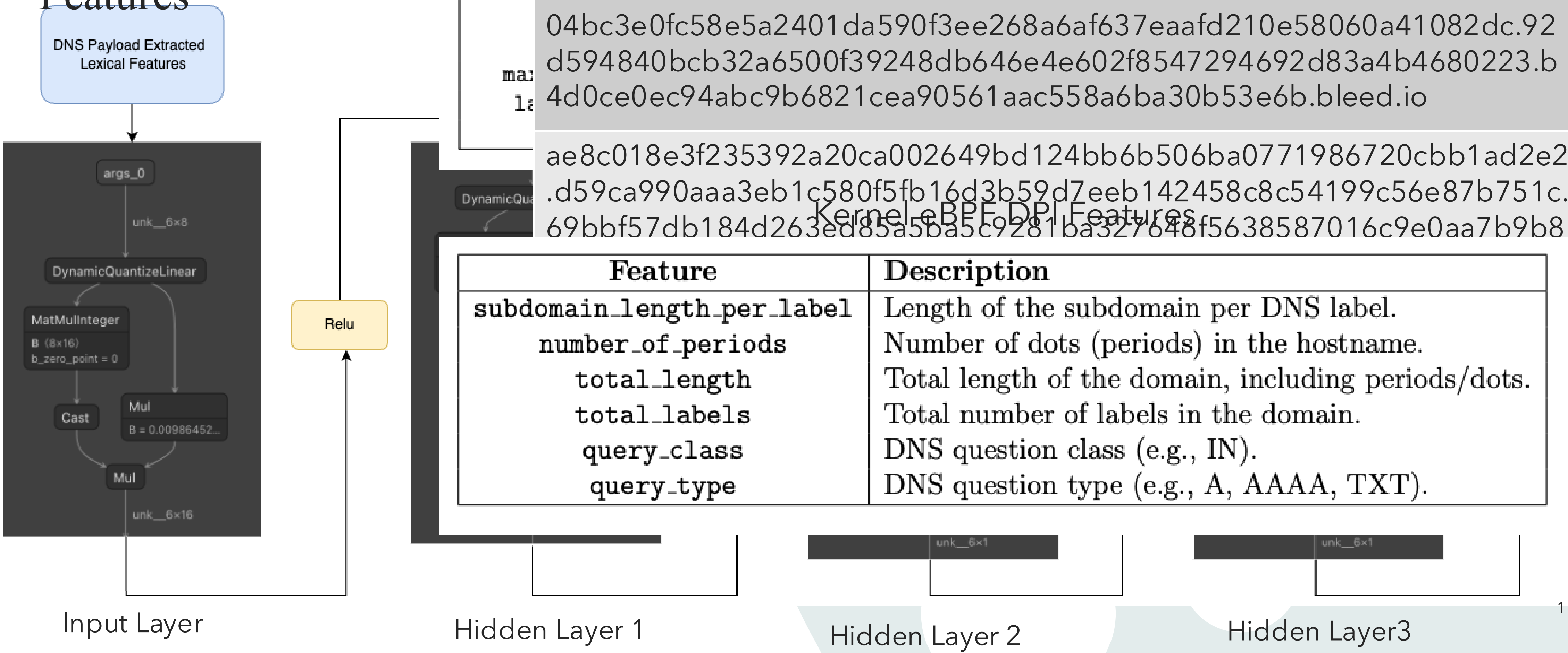
DNN Features

Feature	Description
total	
ma	
la	

Malicious Exfiltrated data DNS queries
381c018e3f5d05b78e3f6a026381e0f3476c066e8017be6ba9f5a9d758ef.d04bc3e0fc58e5a2401da590f3ee268a6af637eaafd210e58060a41082dc.92d594840bcb32a6500f39248db646e4e602f8547294692d83a4b4680223.b4d0ce0ec94abc9b6821cea90561aac558a6ba30b53e6b.bleed.io
ae8c018e3f235392a20ca002649bd124bb6b506ba0771986720cbb1ad2e2.d59ca990aaa3eb1c580f5fb16d3b59d7eeb142458c8c54199c56e87b751c.69bbf57db184d263ed85a5ba5c9281ba327648f5638587016c9e0aa7b9b8.

Kernel eBPF DPI Features

Feature	Description
subdomain_length_per_label	Length of the subdomain per DNS label.
number_of_periods	Number of dots (periods) in the hostname.
total_length	Total length of the domain, including periods/dots.
total_labels	Total number of labels in the domain.
query_class	DNS question class (e.g., IN).
query_type	DNS question type (e.g., A, AAAA, TXT).



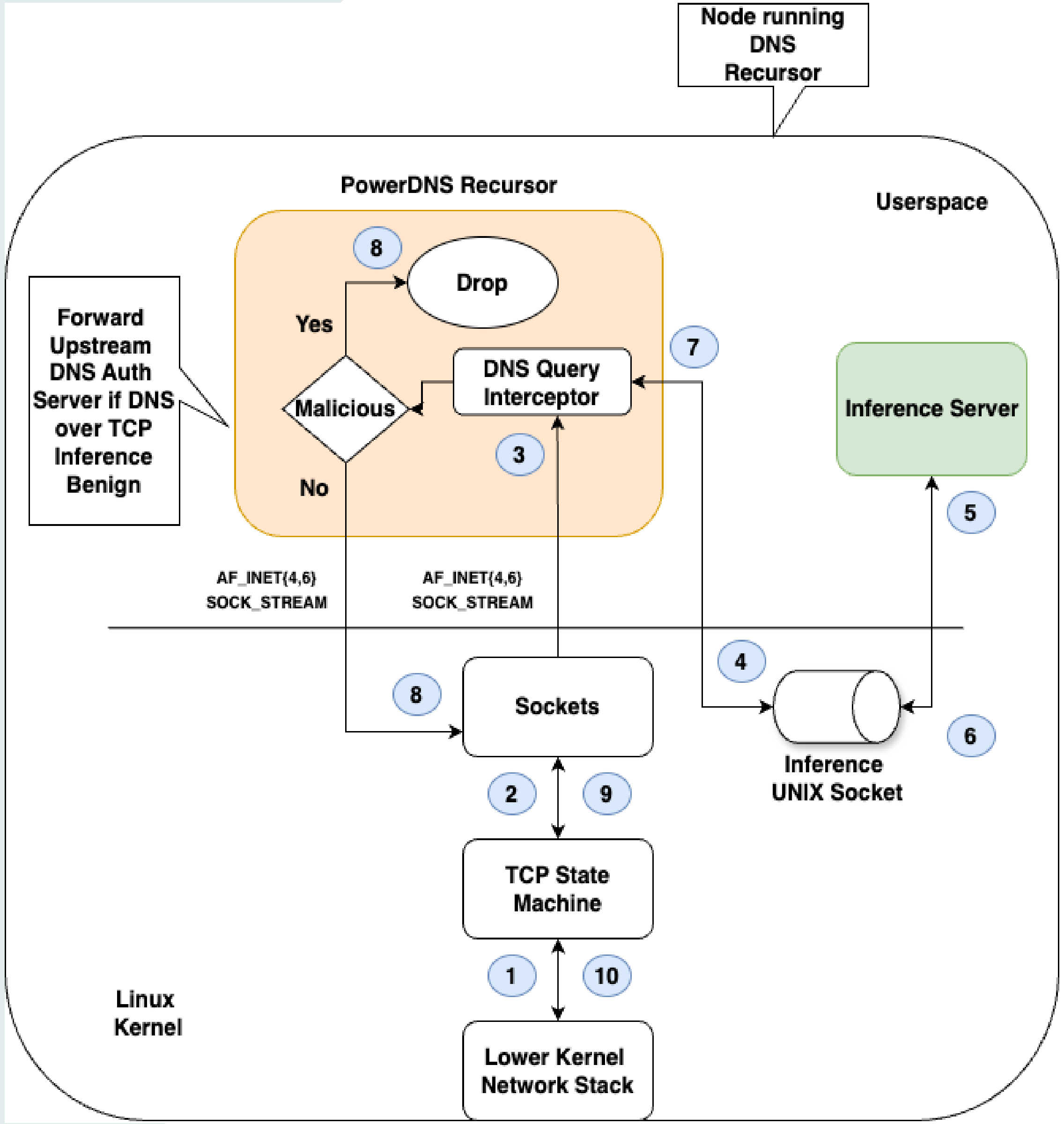
Datasets

Dataset Type	Source / Characteristics	Size	Primary Goal
Trusted Benign Cache	Top 1M Cisco Second-Level Domains (SLDs)	1 Million	Reduce inference on known-good traffic.
ISP-Captured DNS	Live-sniffed ISP DNS traffic [Ziza et al.]	50 Million	Provide real-world benign & malicious baseline.
Synthetic Exfiltration	Custom-generated (DET, DNSCat2, Sliver, Nuages, Custom Scripts, etc.);	2.4 Million	Malicious samples use varied obfuscation across file formats
Final Combined Dataset	Synthetically formed	3.8 millions	Balanced dataset w/ obfuscated payloads across file formats

DNS Exfiltration over TCP

Prevent DNS Exfiltration over TCP

- Runs on
  - PowerDNS Recursor
- Relies on
  - PowerDNS recursor Query Interceptors
  - Inference UNIX domain sockets

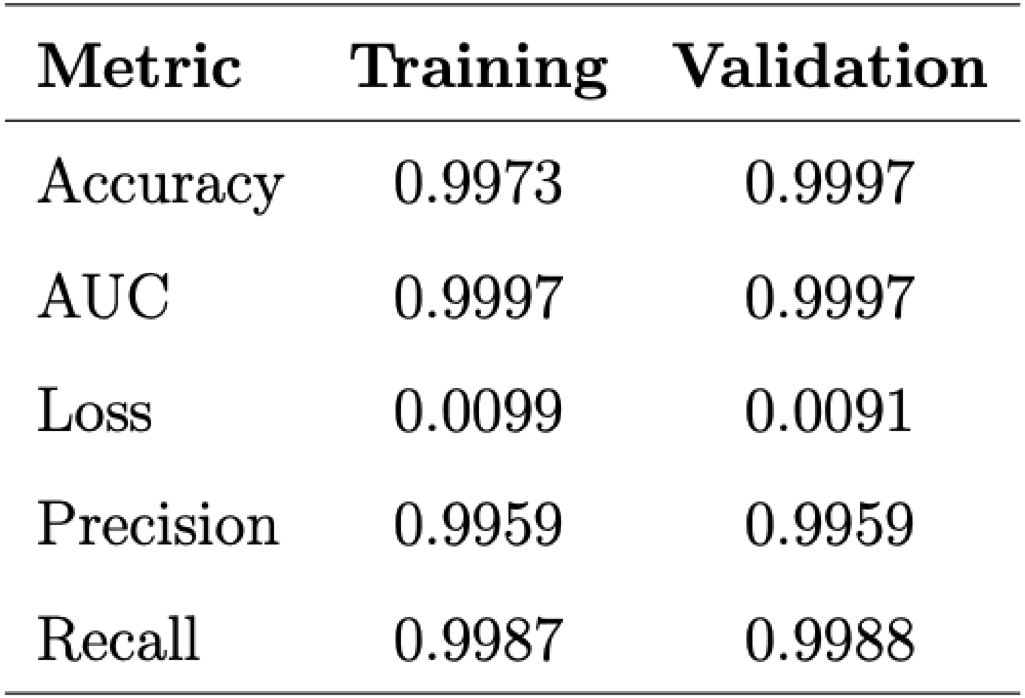


Results and Evaluation

- Model Metrics
- Throughput comparisons (Active mode)
- Response Time per Exfiltration attempt
- Kernel DPI time (raw parse DNS protocol from SKB)
- Resources
  - Memory Usage
    - Security Agent memory usage at endpoints in data plane
  - Endpoint Agent Flame Graph

Test Bench
CPU: Intel Xeon 6130
Memory: 8 GB
Linux Kernel: 6.12.4
Network Driver: netvsc
Bandwidth: 100 Gb/sec
Root QDISC: FQ_Codel
Queues: 8 RX / TX

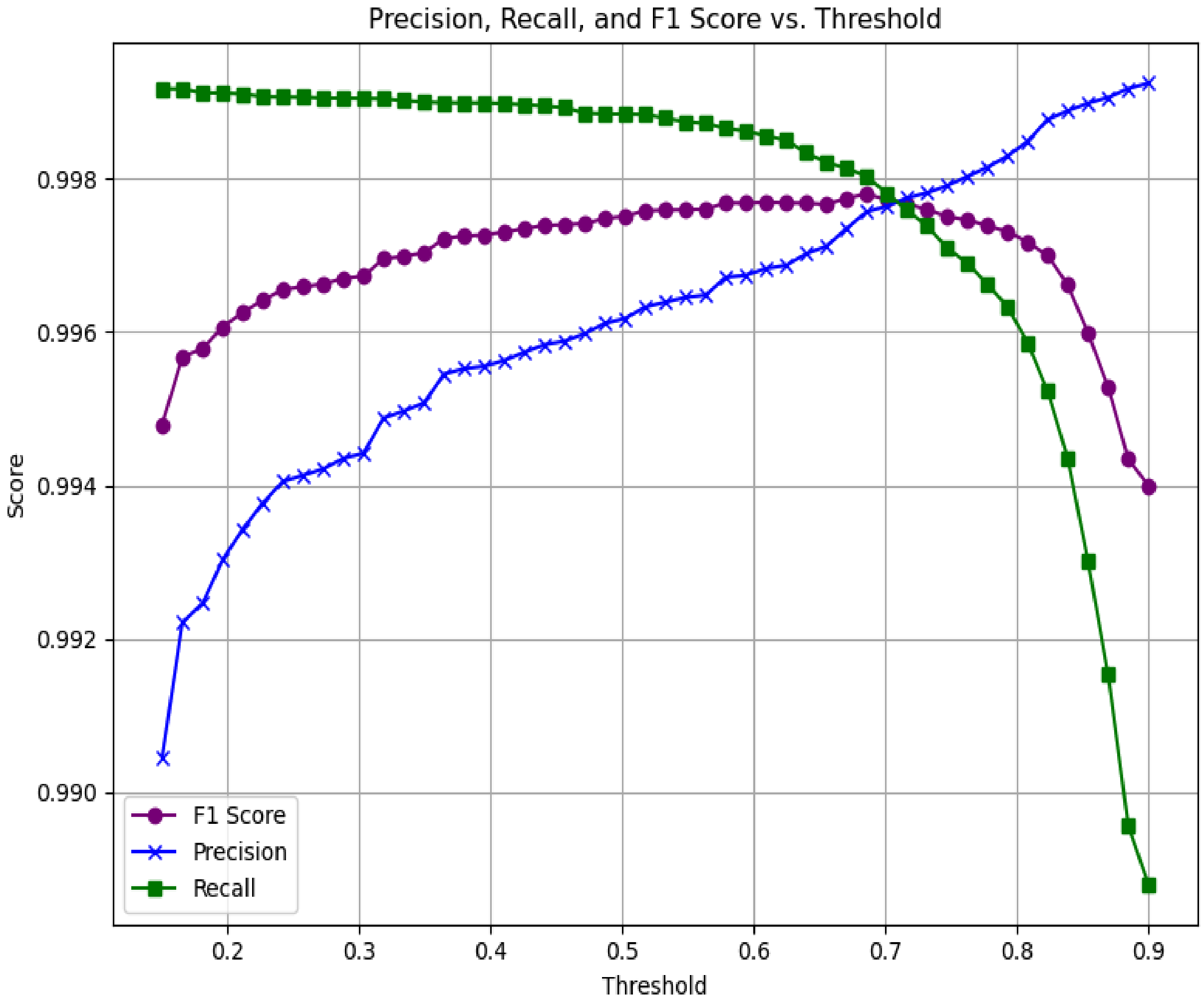
DNN Model Metrics



Metric	Training	Validation
Accuracy	0.9973	0.9997
AUC	0.9997	0.9997
Loss	0.0099	0.0091
Precision	0.9959	0.9959
Recall	0.9987	0.9988

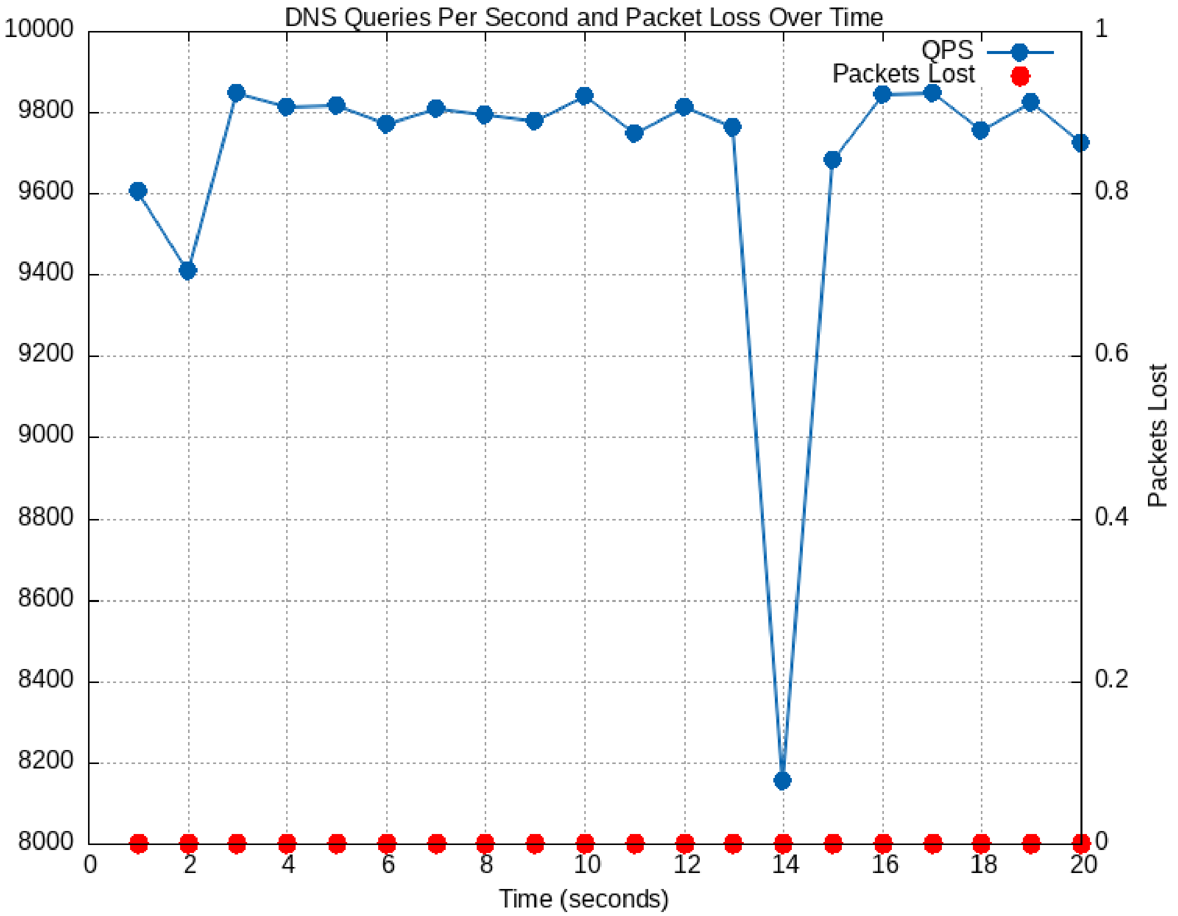
Table 5.1: Model Evaluation Metrics

Model Performance

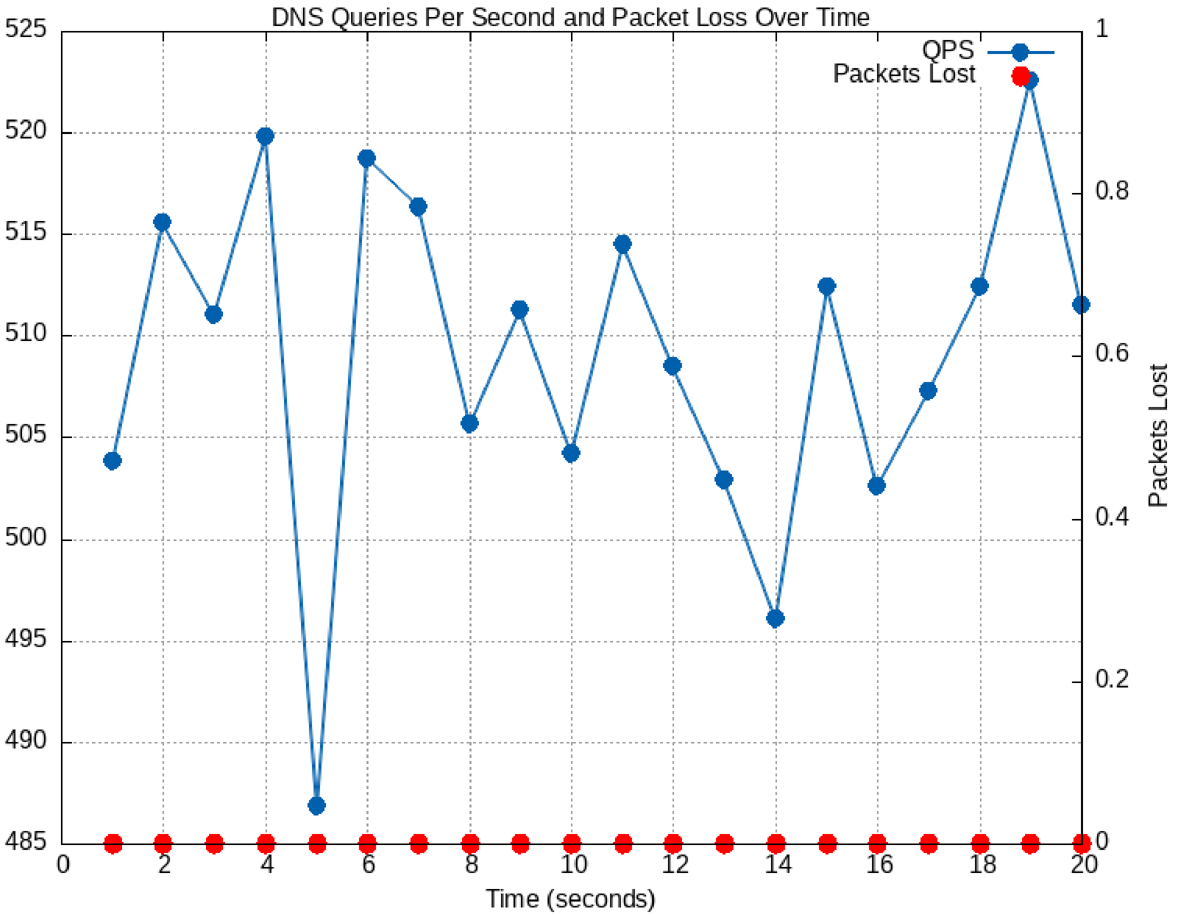


Model Scores

Throughput comparisons – Active Mode

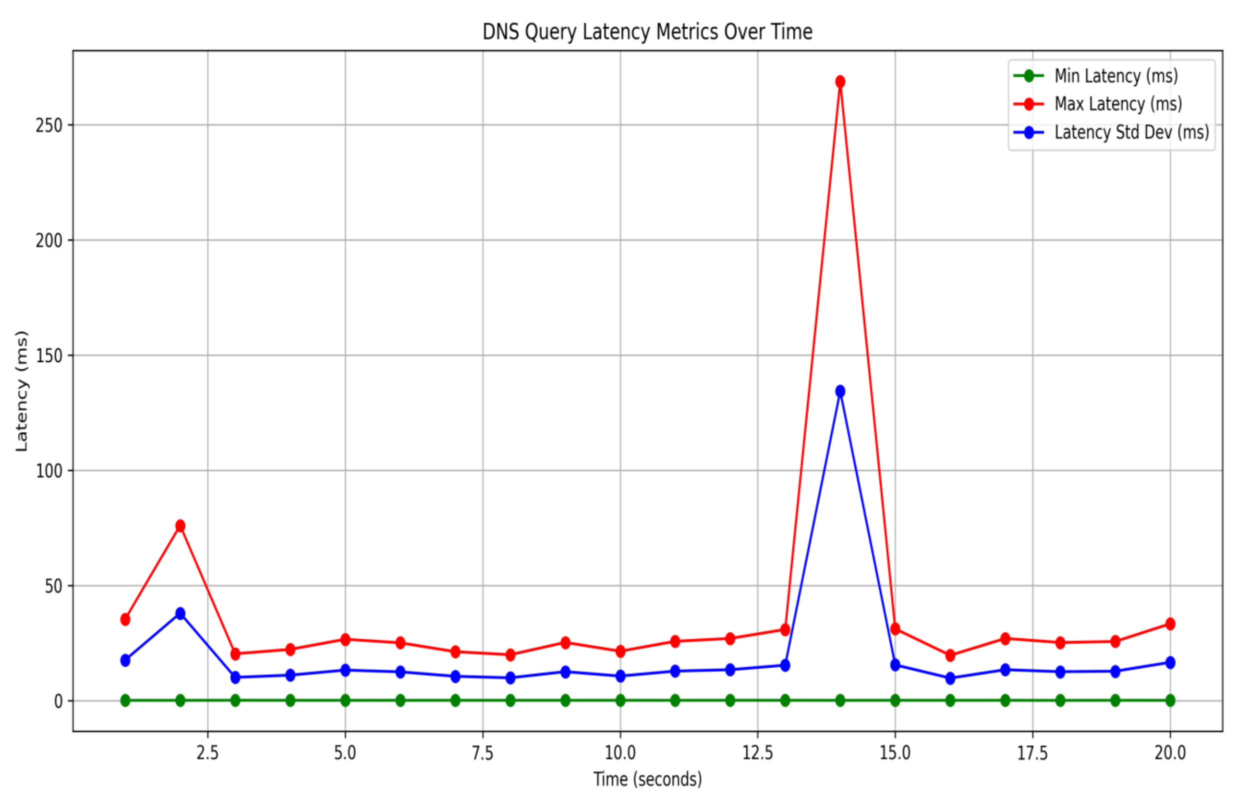


Agent LRU Cache Read-Through Hit 10k DNS req/sec

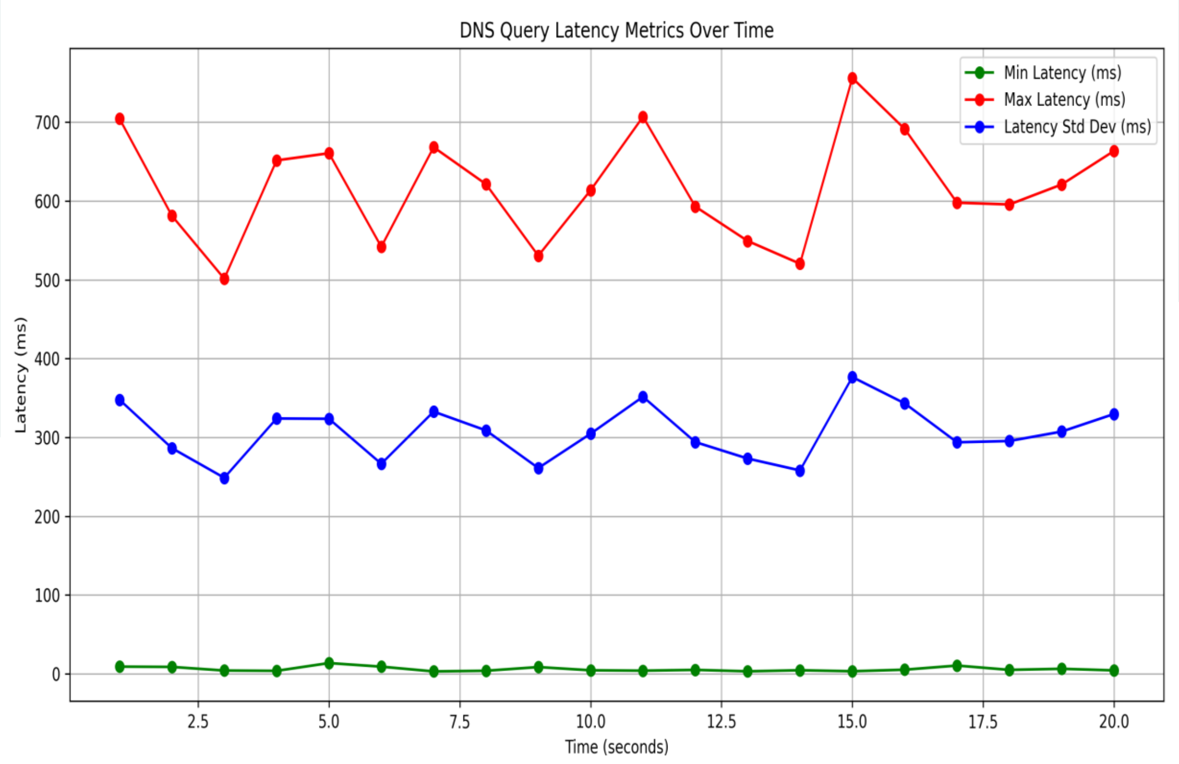


ONNX Live Inferencing 10k DNS req/sec

Throughput comparisons – Active Mode (continued)



Agent LRU Cache Read-Through Hit

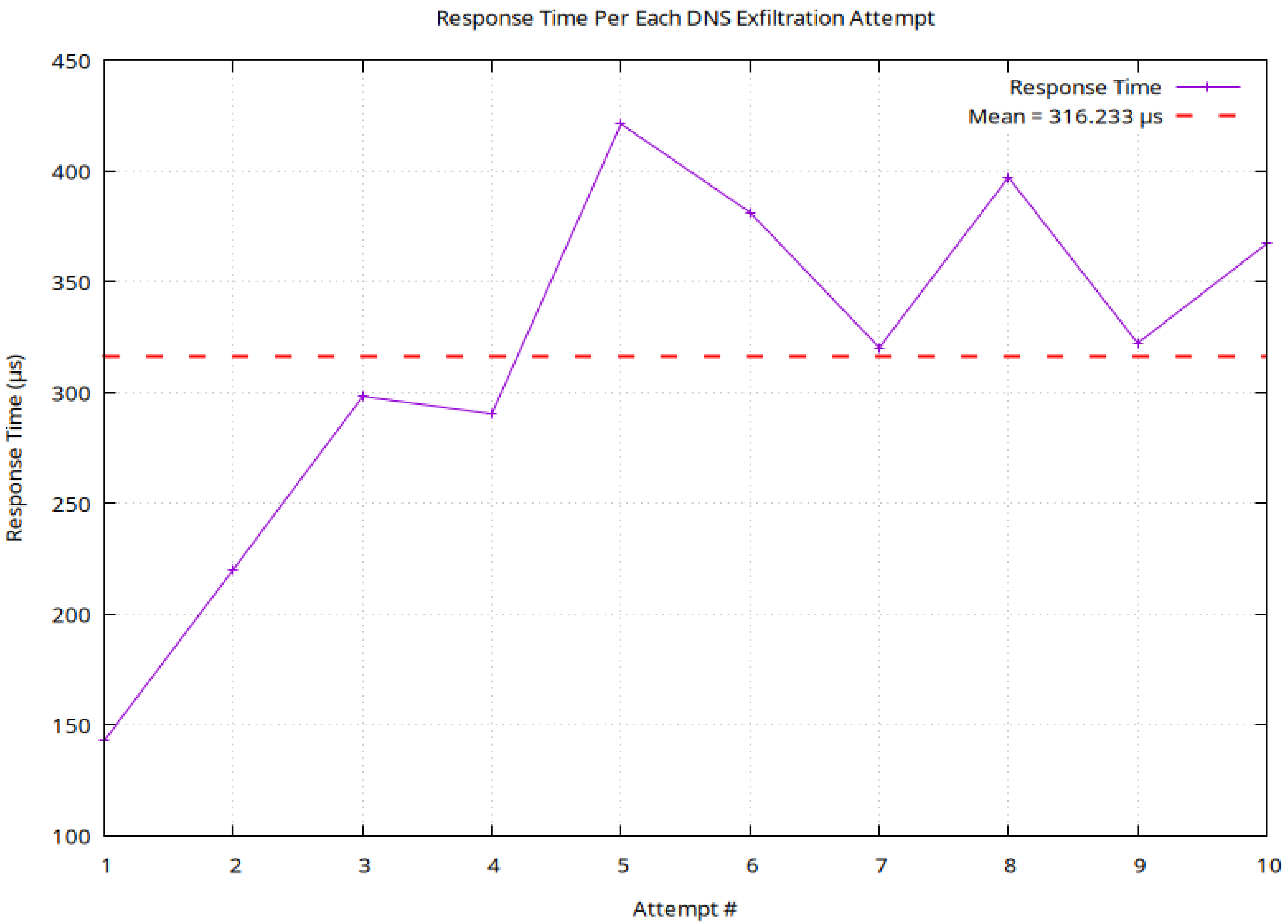


ONNX Live Inferencing

Response Speed - Active Mode

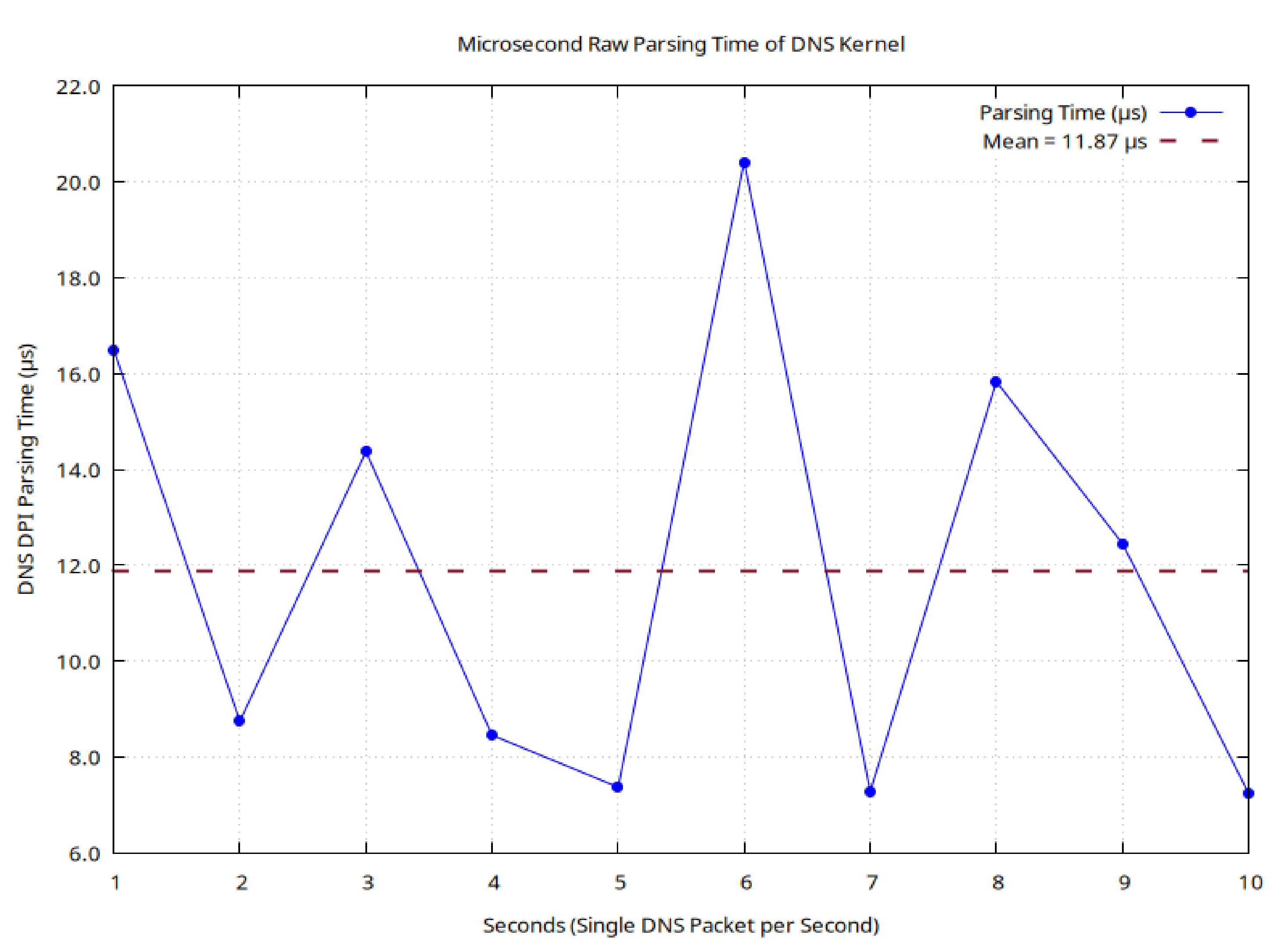
Response Speed Before Implant Eventually Killed

- Each Exfiltration Attempt



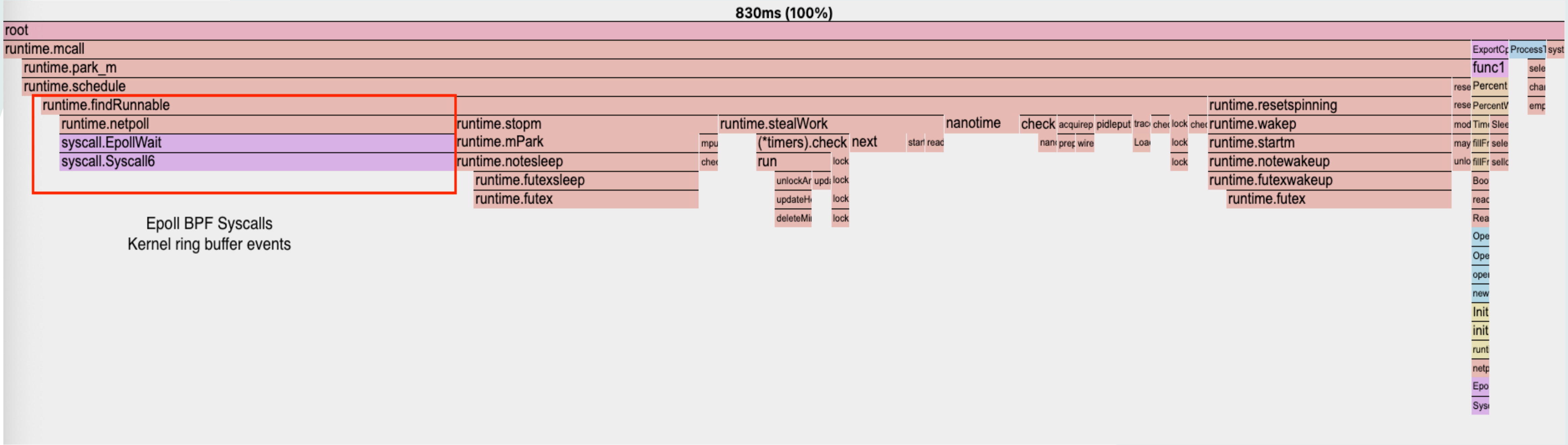
Kernel eBPF Program DPI Time

Kernel DNS Deep Parsing Time



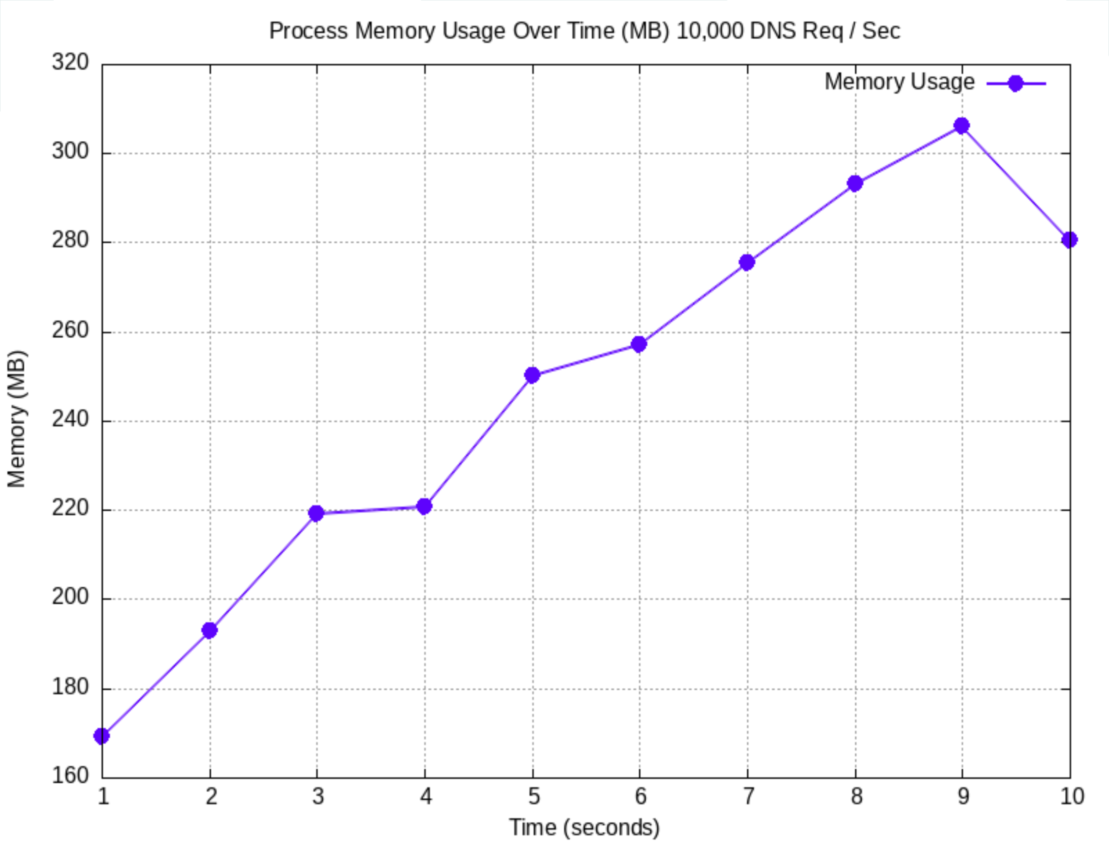
Resource Usage – eBPF Agent Flame Graph

Kernel Epoll asynchronous I/O agent performance boost

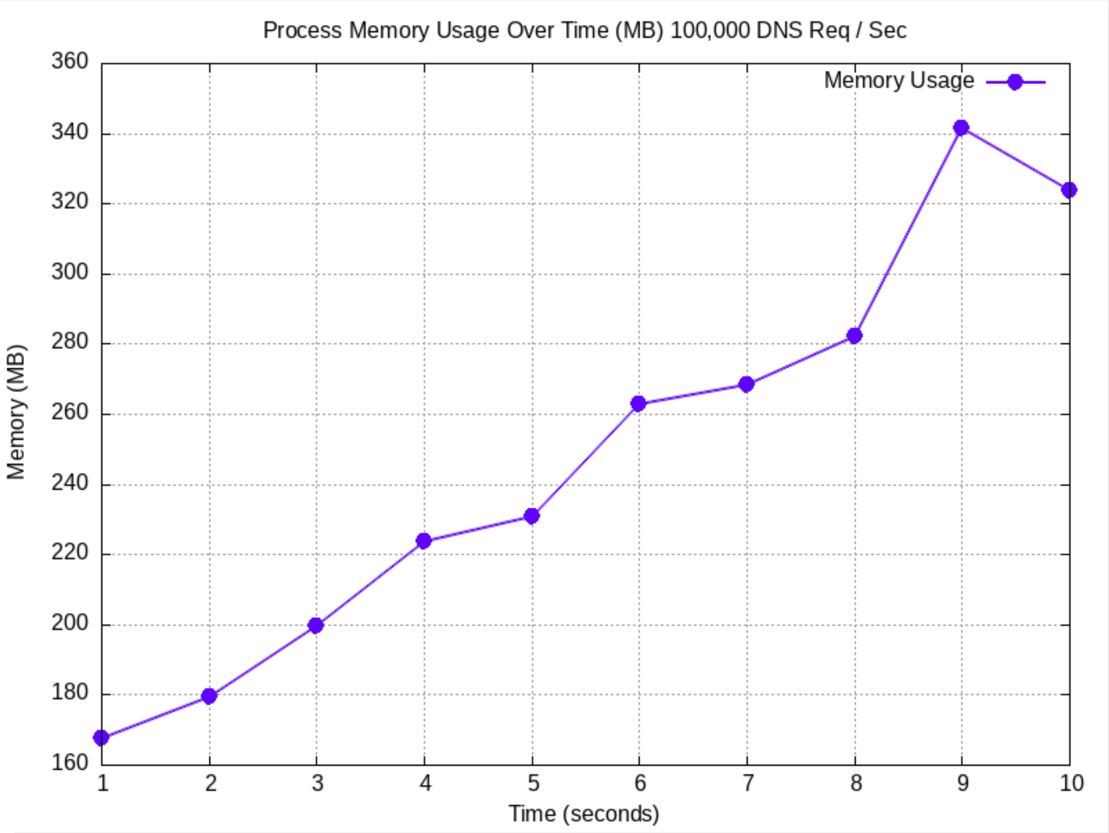


Epoll BPF Syscalls
Kernel ring buffer events

Resource Usage - Memory

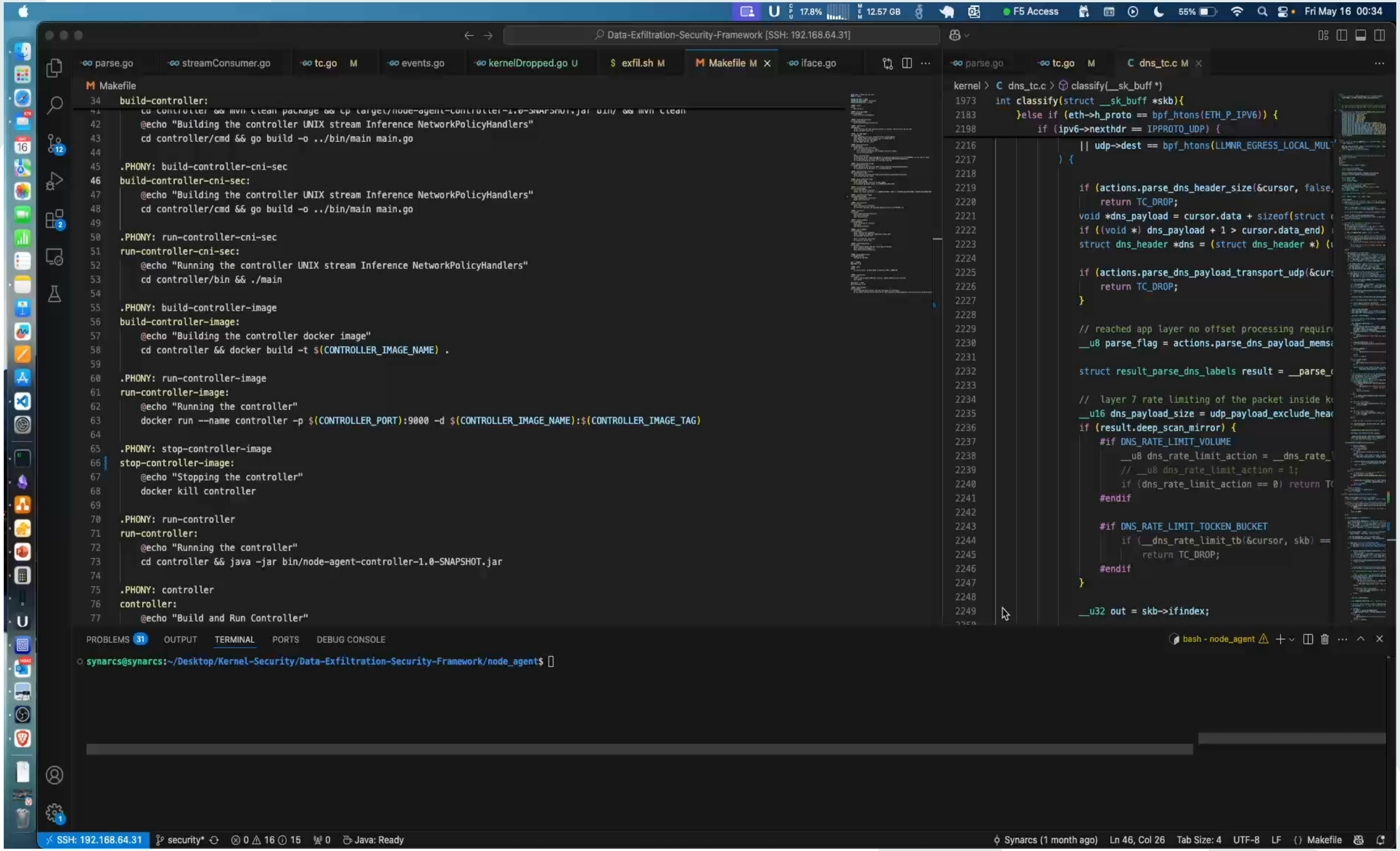


10,000 DNS Req / Sec



100,000 DNS Req / Sec

Framework Security Strength



```makefile
build-controller:
    cd controller && mvn clean package && cp target/node-agent-controller-1.0-SNAPSHOT.jar bin/ && mvn clean
    @echo "Building the controller UNIX stream Inference NetworkPolicyHandlers"
    cd controller/cmd && go build -o ../bin/main main.go

.PHONY: build-controller-cni-sec
build-controller-cni-sec:
    @echo "Building the controller UNIX stream Inference NetworkPolicyHandlers"
    cd controller/cmd && go build -o ../bin/main main.go

.PHONY: run-controller-cni-sec
run-controller-cni-sec:
    @echo "Running the controller UNIX stream Inference NetworkPolicyHandlers"
    cd controller/bin && ./main

.PHONY: build-controller-image
build-controller-image:
    @echo "Building the controller docker image"
    cd controller && docker build -t $(CONTROLLER_IMAGE_NAME) .

.PHONY: run-controller-image
run-controller-image:
    @echo "Running the controller"
    docker run --name controller -p $(CONTROLLER_PORT):9000 -d $(CONTROLLER_IMAGE_NAME):$(CONTROLLER_IMAGE_TAG)

.PHONY: stop-controller-image
stop-controller-image:
    @echo "Stopping the controller"
    docker kill controller

.PHONY: run-controller
run-controller:
    @echo "Running the controller"
    cd controller && java -jar bin/node-agent-controller-1.0-SNAPSHOT.jar

.PHONY: controller
controller:
    @echo "Build and Run Controller"
```

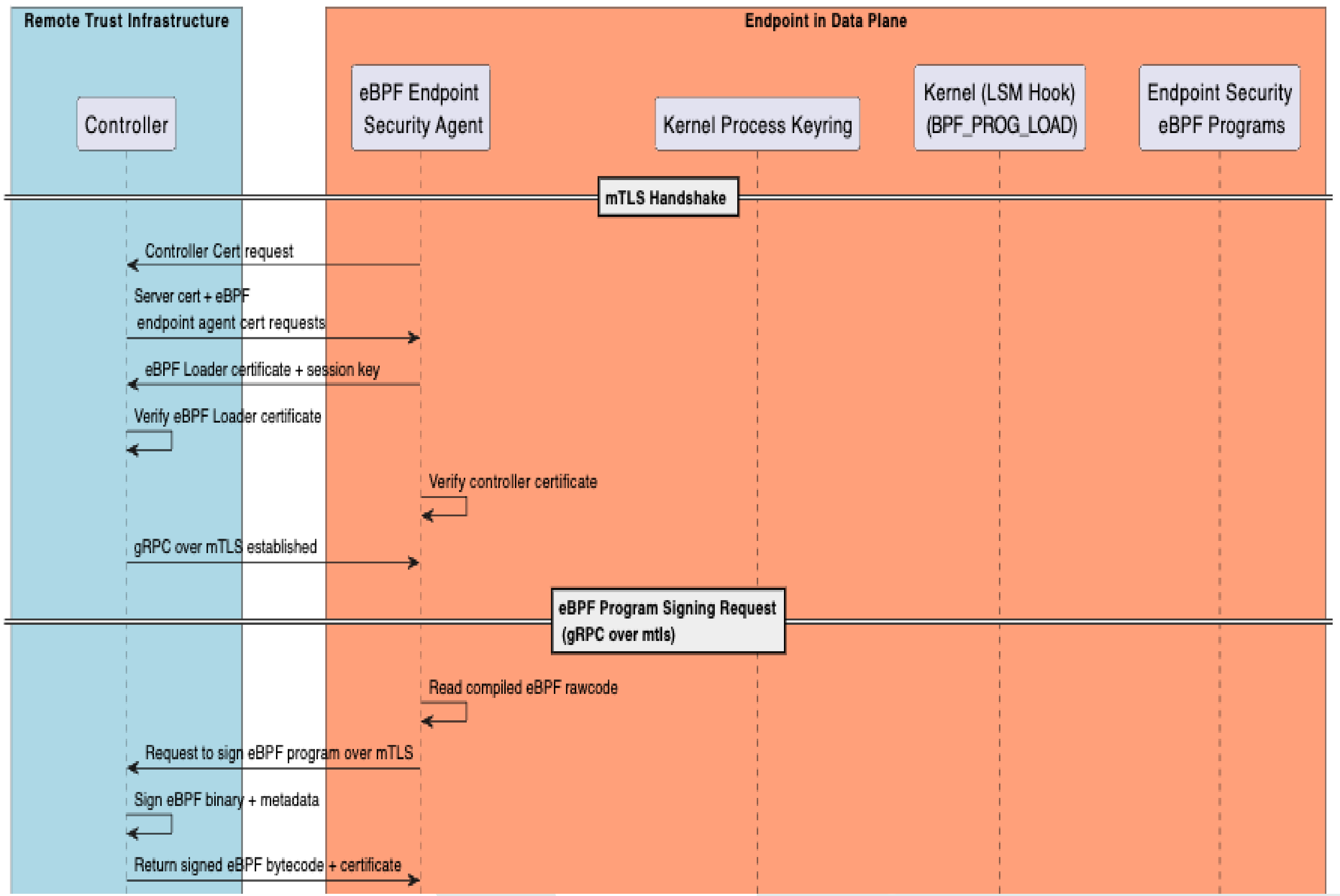
```c
int classify(struct __sk_buff *skb){
    }else if (eth->h_proto == bpf_htons(ETH_P_IPV6)) {
        if (ipv6->nexthdr == IPPROTO_UDP) {
                    || udp->dest == bpf_htons(LLMNR_EGRESS_LOCAL_MUL
                ) {

                    if (actions.parse_dns_header_size(&cursor, false,
                        return TC_DROP;
                    void *dns_payload = cursor.data + sizeof(struct
                    if ((void *) dns_payload + 1 > cursor.data_end)
                    struct dns_header *dns = (struct dns_header *) (

                    if (actions.parse_dns_payload_transport_udp(&curs
                        return TC_DROP;
                    }

                    // reached app layer no offset processing requir
                    __u8 parse_flag = actions.parse_dns_payload_memsa

                    struct result_parse_dns_labels result = __parse_

                    //  layer 7 rate limiting of the packet inside k
                    __u16 dns_payload_size = udp_payload_exclude_hea
                    if (result.deep_scan_mirror) {
                        #if DNS_RATE_LIMIT_VOLUME
                            __u8 dns_rate_limit_action = __dns_rate_
                            // __u8 dns_rate_limit_action = 1;
                            if (dns_rate_limit_action == 0) return T
                        #endif

                        #if DNS_RATE_LIMIT_TOCKEN_BUCKET
                            if (__dns_rate_limit_tb(&cursor, skb) ==
                                return TC_DROP;
                        #endif
                    }

                    __u32 out = skb->ifindex;
```

```
synarcs@synarcs:~/Desktop/Kernel-Security/Data-Exfiltration-Security-Framework/node_agent$
```

First Trust Chain (Endpoint-Agents – Cloud Trust Infrastructure)

- **Endpoint Agents**
- **Controller (Remote PKI)**

- **mTLS-based identity handshake**
- **Remote-signed eBPF raw bytecode**



Remote Trust Infrastructure

Endpoint in Data Plane

Controller

eBPF Endpoint Security Agent

Kernel Process Keyring

Kernel (LSM Hook) (BPF_PROG_LOAD)

Endpoint Security eBPF Programs

mTLS Handshake

Controller Cert request

Server cert + eBPF endpoint agent cert requests

eBPF Loader certificate + session key

Verify eBPF Loader certificate

Verify controller certificate

gRPC over mTLS established

eBPF Program Signing Request (gRPC over mtls)

Read compiled eBPF rawcode

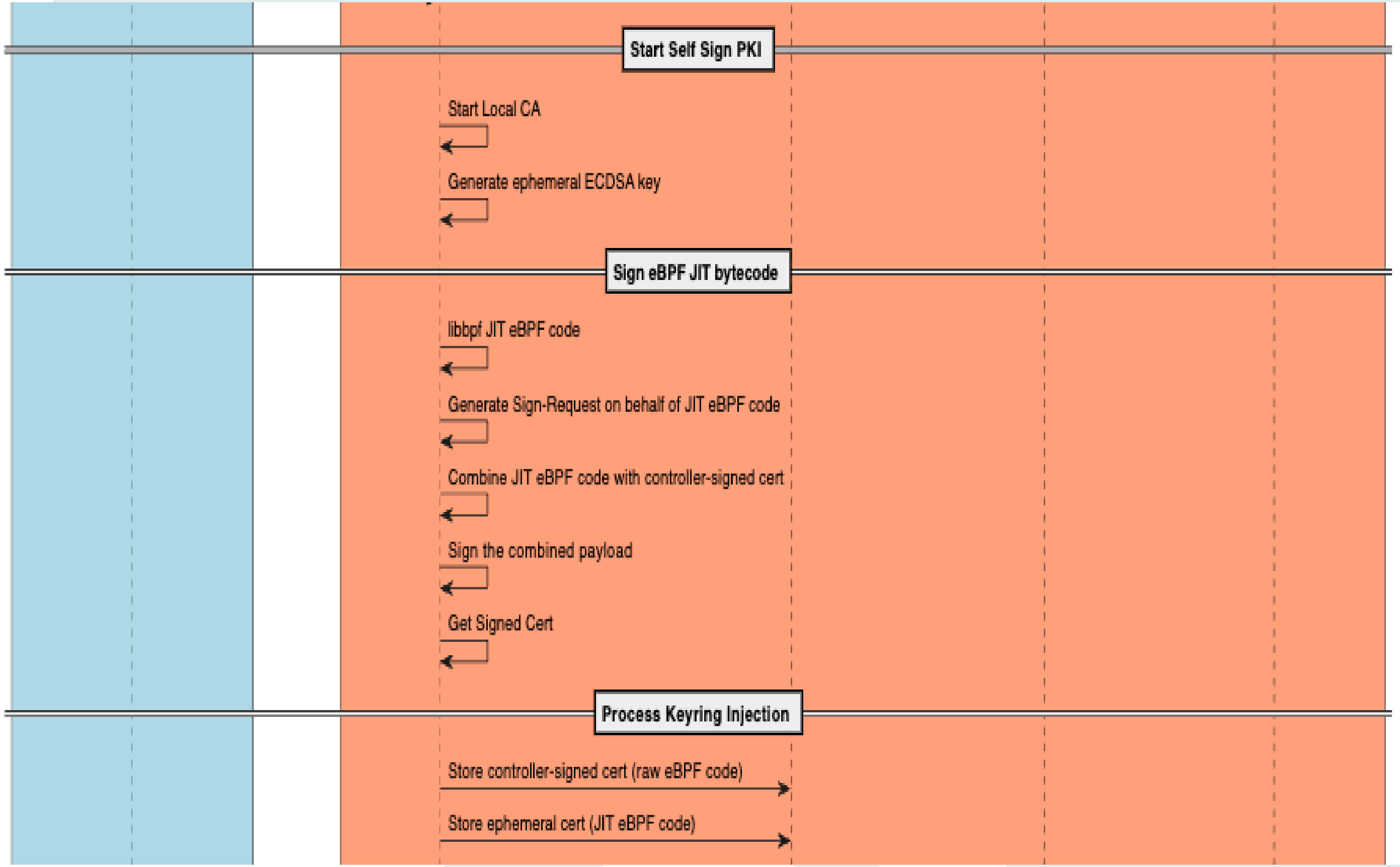
Request to sign eBPF program over mTLS

Sign eBPF binary + metadata

Return signed eBPF bytecode + certificate

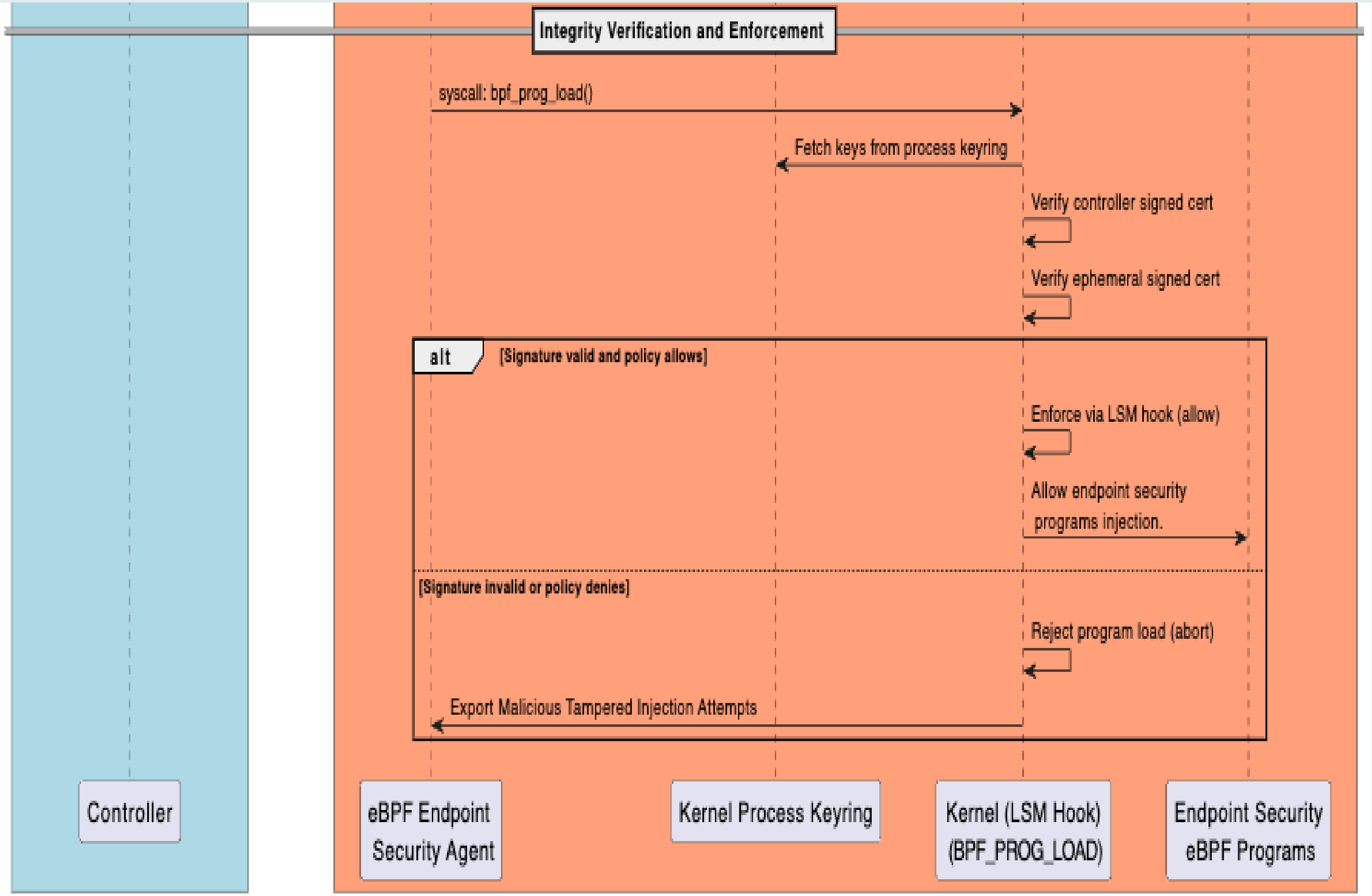
Ephemeral Runtime Signing Bootstrap PKI (JIT bytecode integrity)

- **Ephemeral PKI bootstrapping**
- **JIT eBPF bytecode signing**
- **Controller-bound cert injection**
- **Keyring-based program identity**


Start Self Sign PKI
Start Local CA
Generate ephemeral ECDSA key
Sign eBPF JIT bytecode
libbpf JIT eBPF code
Generate Sign-Request on behalf of JIT eBPF code
Combine JIT eBPF code with controller-signed cert
Sign the combined payload
Get Signed Cert
Process Keyring Injection
Store controller-signed cert (raw eBPF code)
Store ephemeral cert (JIT eBPF code)

Second Trust Chain (Endpoint-Agents – Kernel Keyring)

- Verify controller signed cert (raw eBPF bytecode)
- Verify ephemeral signed cert (JIT bytecode)
- Ensure 3-way strong integrity prevent eBPF program tampering.


Integrity Verification and Enforcement
syscall: bpf_prog_load()
Fetch keys from process keyring
Verify controller signed cert
Verify ephemeral signed cert
alt
[Signature valid and policy allows]
Enforce via LSM hook (allow)
Allow endpoint security programs injection.
[Signature invalid or policy denies]
Reject program load (abort)
Export Malicious Tampered Injection Attempts
Controller
eBPF Endpoint Security Agent
Kernel Process Keyring
Kernel (LSM Hook) (BPF_PROG_LOAD)
Endpoint Security eBPF Programs

Summary and Future Work

- **Extend Support for DNS-over-TCP and Encrypted Tunnels:** Implement in-kernel eBPF-based detection for DNS-over-TCP replicating TCP state machine over kernel socket layer, paired with userspace DPI via Envoy proxy.
- **Add In-Kernel TLS Fingerprinting**: Use eBPF for TLS fingerprinting (e.g., JA3/JA4) to detect DNS exfiltration over TLS (DOH), DNS over mTLS, WireGuard.
- **Continuous Model Evolution:** Drift detection, online learning, and confidence-based live updates to maintain precision against emerging DNS obfuscation tactics.
- **Cloud Native Security:**
  - Dynamic L3/L7 security enforcement over cloud Vnet’s / VPC via dynamic blacklist’s NACL’s.

Discussion and QA

Codebase:
https://github.com/Synarcs/DNSObelisk

WhitePaper:
https://github.com/Synarcs/DNSObelisk_Report