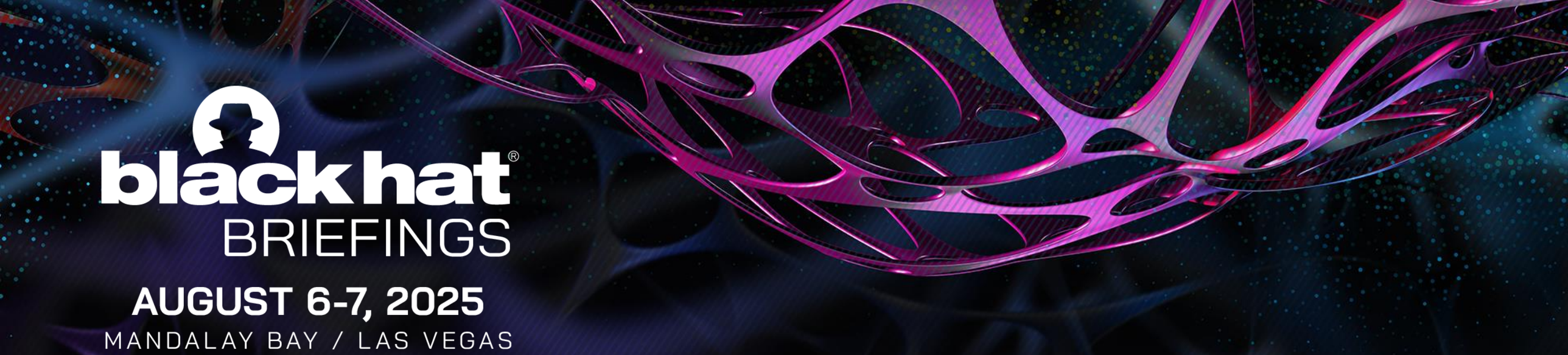


From Packet to Process: Disrupting DNS C2 with AI and eBPF in Linux Kernel for cloud environments

Speaker: Vedang Parasnis (Synarcs)

$whoami

Vedang Parasnis

Independent Researcher,

Master's Graduate @University Of Washington

Email: vedang.parasnis@outlook.com

Research Interests:

Kernel security, hardening, eBPF, cloud and system security

Agenda

- ❑ DNS a critical backdoor for enterprise networks
- ❑ DNS Exfiltration Attack Vectors
- ❑ DNS C2 Attack Infrastructure
- ❑ Existing Approaches and Challenges
- ❑ AI-Driven Linux Kernel Enforced Endpoint Security
- ❑ Cloud Deployment Architecture at scale to combat DNS C2 infrastructures
- ❑ Demo (disrupt Sliver, DNSCat2)
- ❑ Key Takeaways & Future Directions
- ❑ Q&A

They Breach Through DNS — Almost Every Time

Compromise Supply Chain:

- APT29 (Cozy Bear) — SolarWinds

Breach Cloud & Hyperscalers:

- UNC2452 (APT29)

Damage Critical Infrastructure:

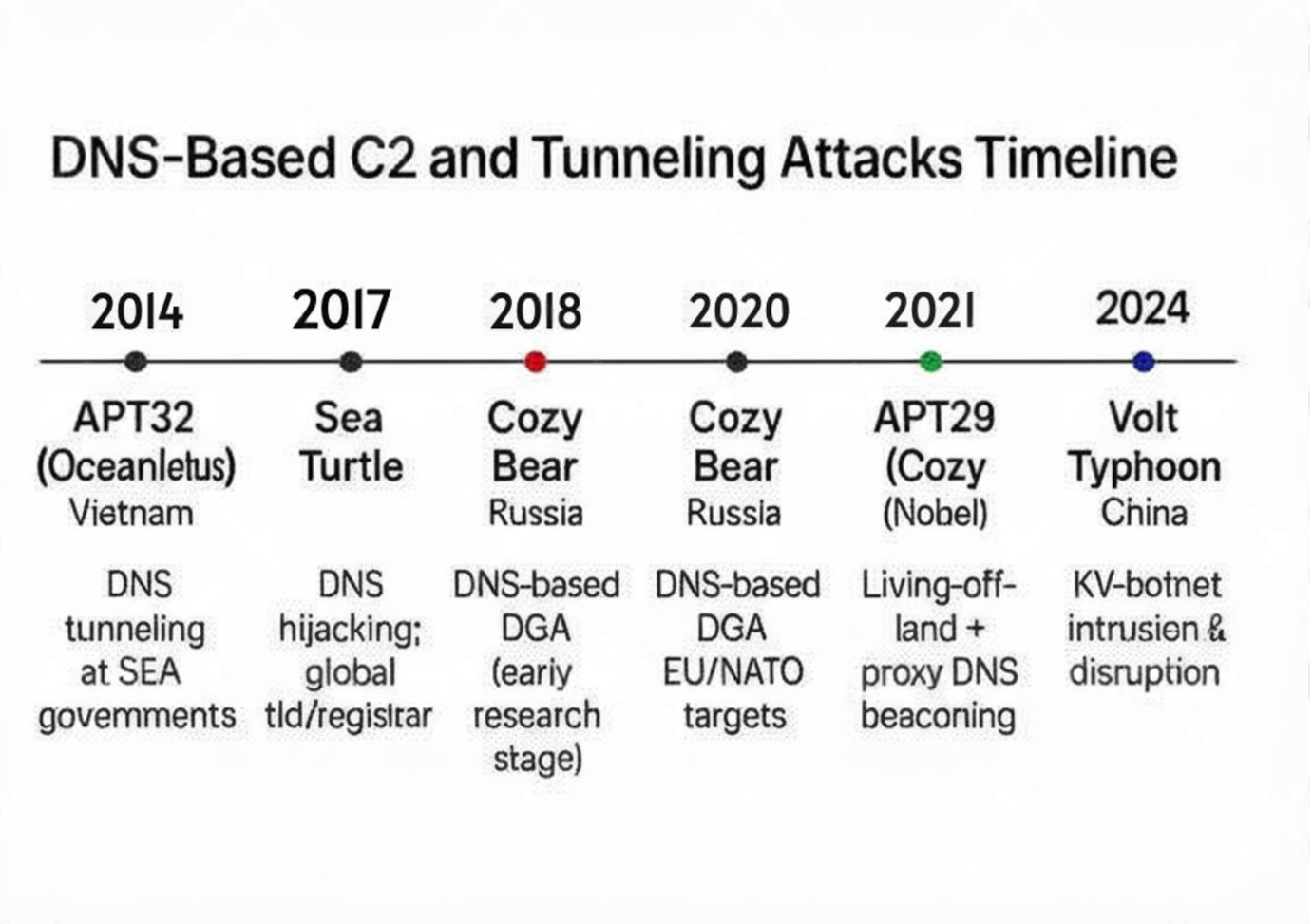
- Volt Typhoon

Harvest Credentials at Scale:

- APT28 (GRU), Sea Turtle

Exploit Shared Offensive Tools:

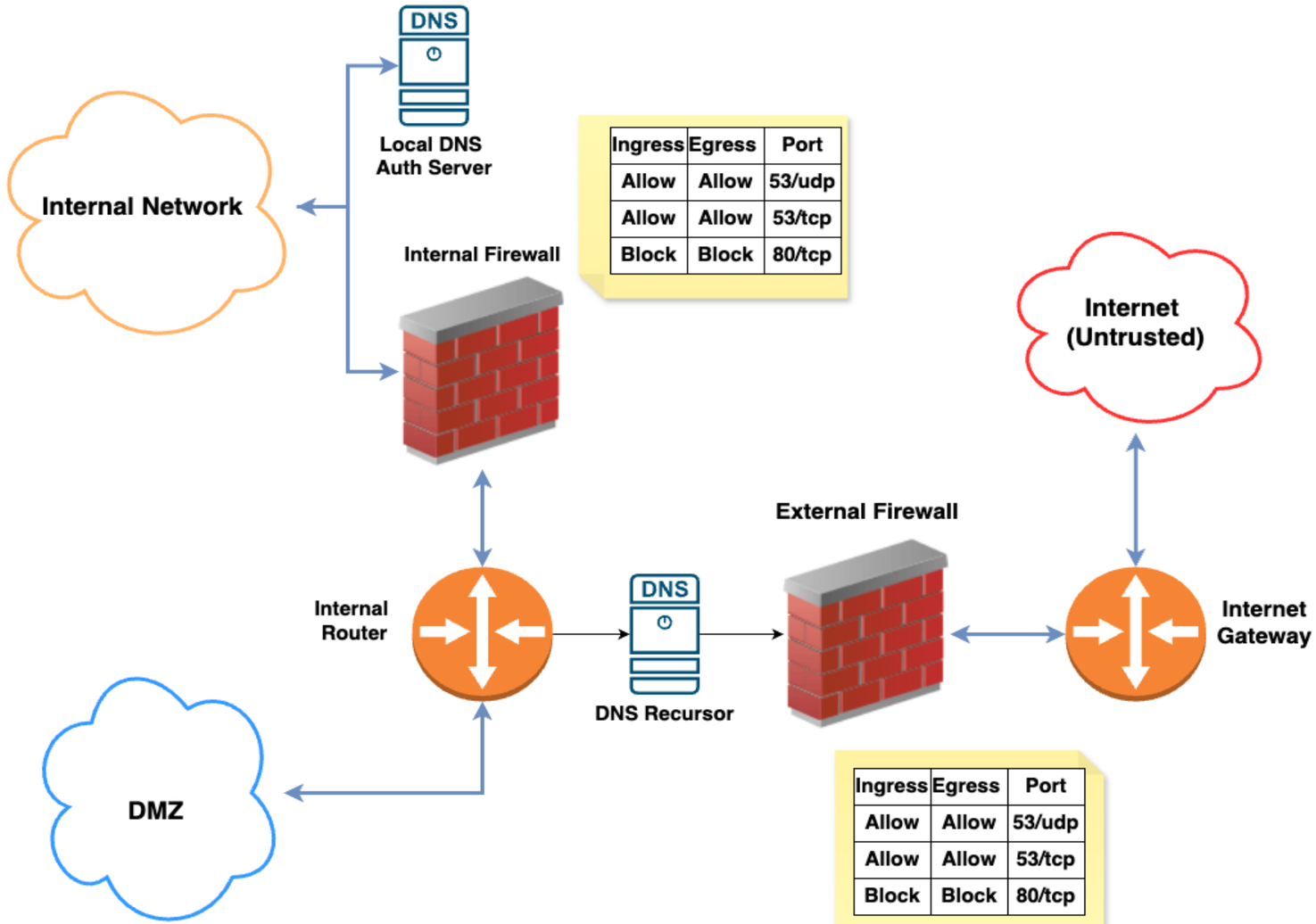
- APT41, FIN7



85%+ of APT's employ DNS for C2 and data breaches

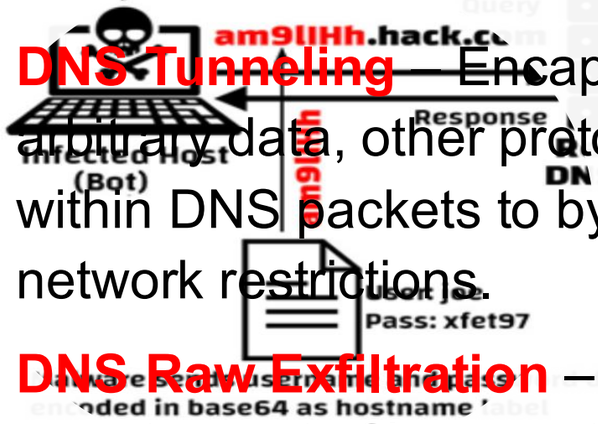
DNS a Blind spot to compromise networks

- Unencrypted by Default
- Logs Rarely Monitored
- Firewall Blindspot
- Stateless Protocol



DNS: Not Just For Name Resolution Anymore. Next channel deliver zero-day attacks.

- **DNS C2** – Uses DNS to embed commands, data in queries and responses to maintain covert communication with remote C2 attacker infrastructure.
- **DNS Tunneling** – Encapsulates arbitrary data, other protocols within DNS packets to bypass network restrictions.
- **DNS Raw Exfiltration** – Leaks sensitive data files directly in DNS queries.



RCE & Shellcode – Exploiting memory bugs, dropping payloads

Script & File Attacks – Scripted execution, file corruption

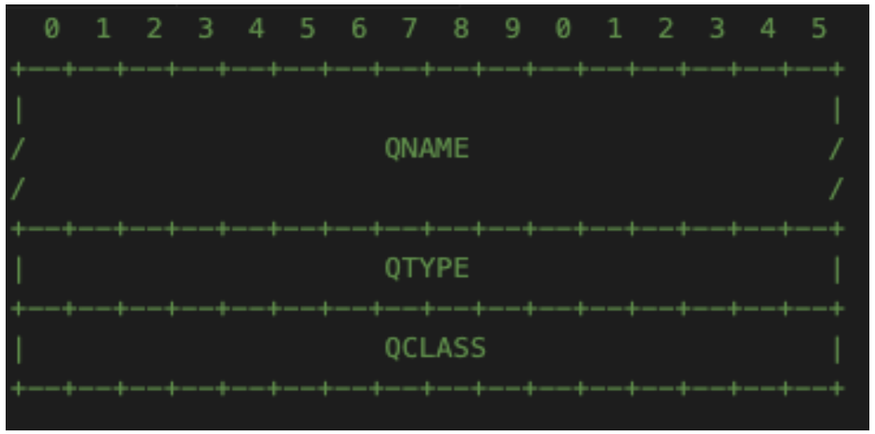
Side-Channel Process Abuse: Processing Injection Hallowing

Persistent Backdoors: Rootkits, ransomware stealth persistence.

Network Pivoting: Port Forwarding, reverse tunnels

DNS Protocol Specifications

DNS	Limit
UDP Packet Size	512 bytes (default) Up to 4096 bytes (with EDNS0)
Max Domain Question length	255
Max number of labels per query	127 labels
Max Label Length	63
Max Response Size	512 bytes, except 4096 for EDNS0
DNS Header Size	Limited by packet size
Query Section Size	Limited by packet size

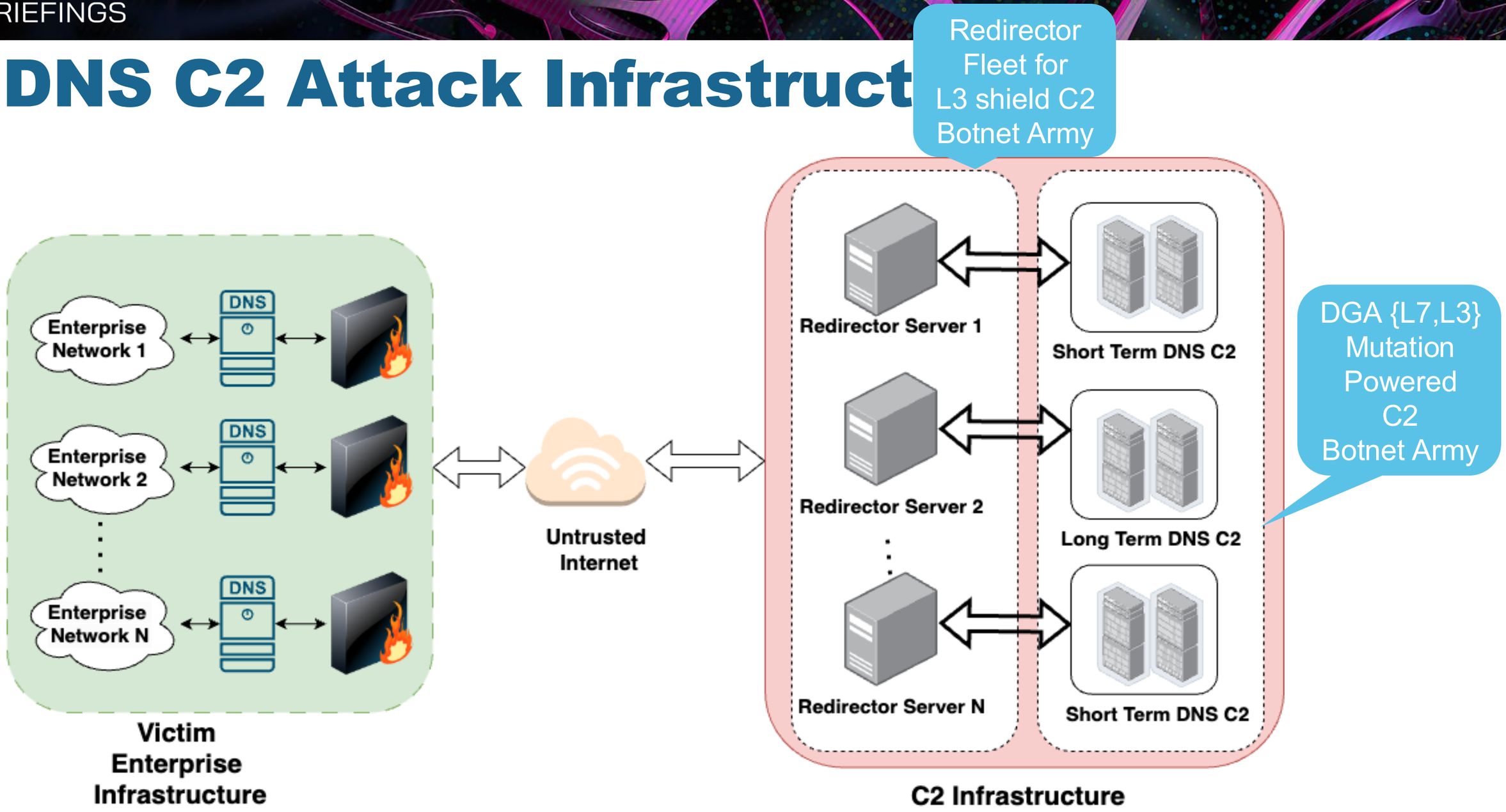


DNS Question Record

What Makes DNS Query contain C2 or exfiltrated data

- ❑ High Entropy QNAME
- ❑ Long or Excessive Labels
- ❑ No Dictionary Tokens
- ❑ DGA-style Patterns
- ❑ NXDOMAIN Abuse and Ghost domains flood

DNS C2 Attack Infrastruct



DGA (L7) and IP (L3) Mutation

- **Evade Detection** – Generates thousands of reflectors, IPS, domains to avoid static and policy blocklists.
- **Resilience** – If one domain is taken down, others remain reachable.
- **No Hardcoded domains** – Domains are algorithmically created on both attacker and implant sides.

Time-Based DGAs

Date + SystemClock
fkeo12jdn7z.com
sk9qpdmx43a.com

Seed-Based DGAs

Seed + shared math functions
bhack1.com
bhack2.com

Wordlist DGAs

Wordlist dictionary
catsun.net
reddog.org

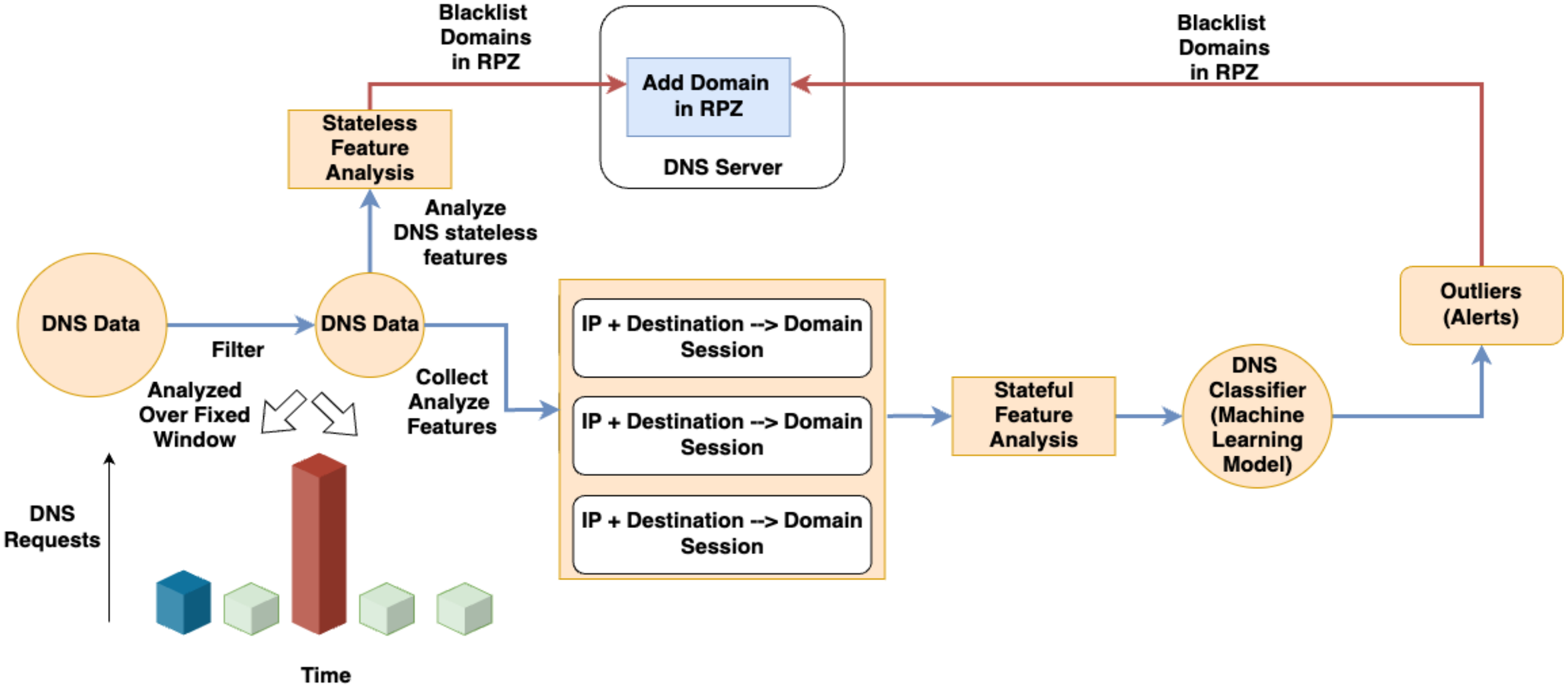
Character-Based or Randomized DGAs

Pseudo random chars
sdas232.bleed.io

Existing Approaches

- **Semi-Passive Analysis**
  - DNS Exfiltration Security as Middleware (DPI as middleware)
- **Passive Analysis**
  - Anomaly Detection (Traffic Timing / Volume)
  - Threat Signatures, Domain Reputation scoring

DNS Traffic Anomaly Detection and Prevention Pipeline



Challenges with current approaches

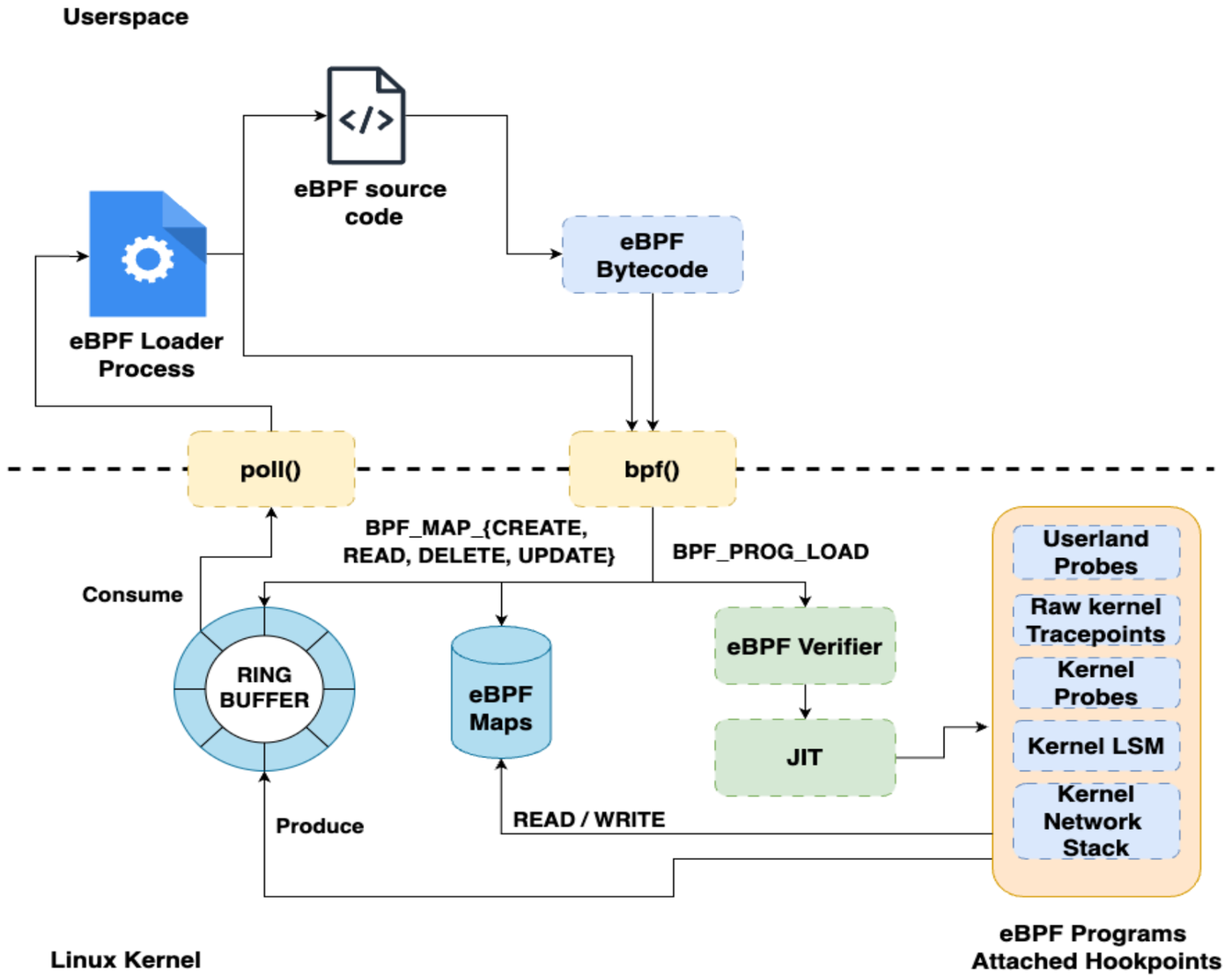
- ❑ Slow Detection, Slower Response: Stealthy mutable Implants survive
- ❑ Slow and easy bypass to Advanced DNS C2 Attacks
- ❑ Lack robust protection over Domain Generation Algorithms, IP mutation
- ❑ Unwanted latency for proxy-based DPI on benign traffic
- ❑ Dynamic Threat Patterns

Proposed Solution:

- ✓ Reactive Kernel EDR at Ring 0 — closest to the wire, at the implant source, beyond reach of userland evasion .

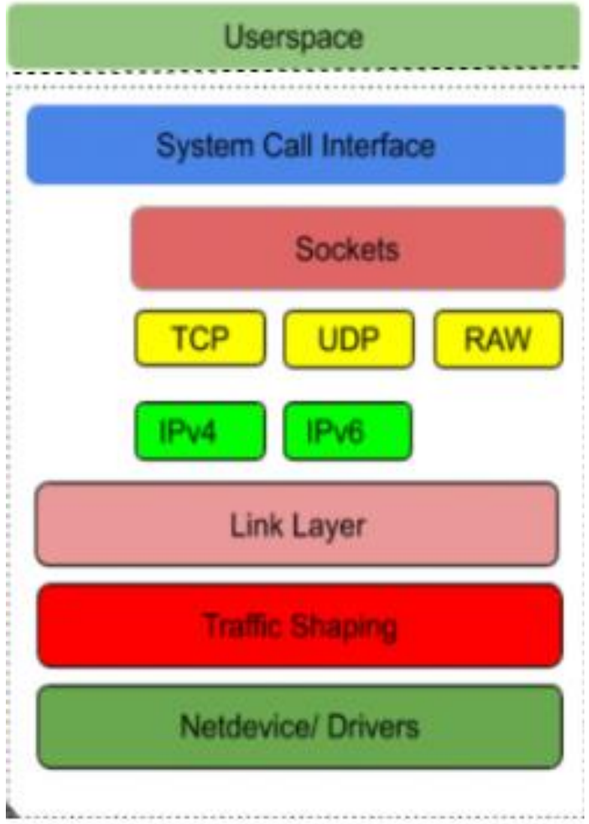
eBPF

- Reprogram the Linux kernel in safe way.
- Runs BPF virtual machine inside kernel
- Custom BPF bytecode
- CPU architecture and Linux kernel version agnostic (BTF)



EDR Agent Linux Kernel eBPF Hooks

Kernel Network Stack Attachr



Kernel Process scheduler

BPF Kprobes/ Tracepoints

BPF Cgroups/ Sockops

DNS Sockets Process

BPF Netfilter

Egress DPI of DNS from SKB

BPF TC

BPF XDP

Kernel MAC (Access Control) Attachments

Userspace

LSM (Linux Security Modules)

BPF LSM

Core Kernel Subsystems

Kernel Keyring, LSM Strong eBPF program integrity

Egress

Kernel Enforced Endpoint Security for DNS

Agent based Endpoint Security

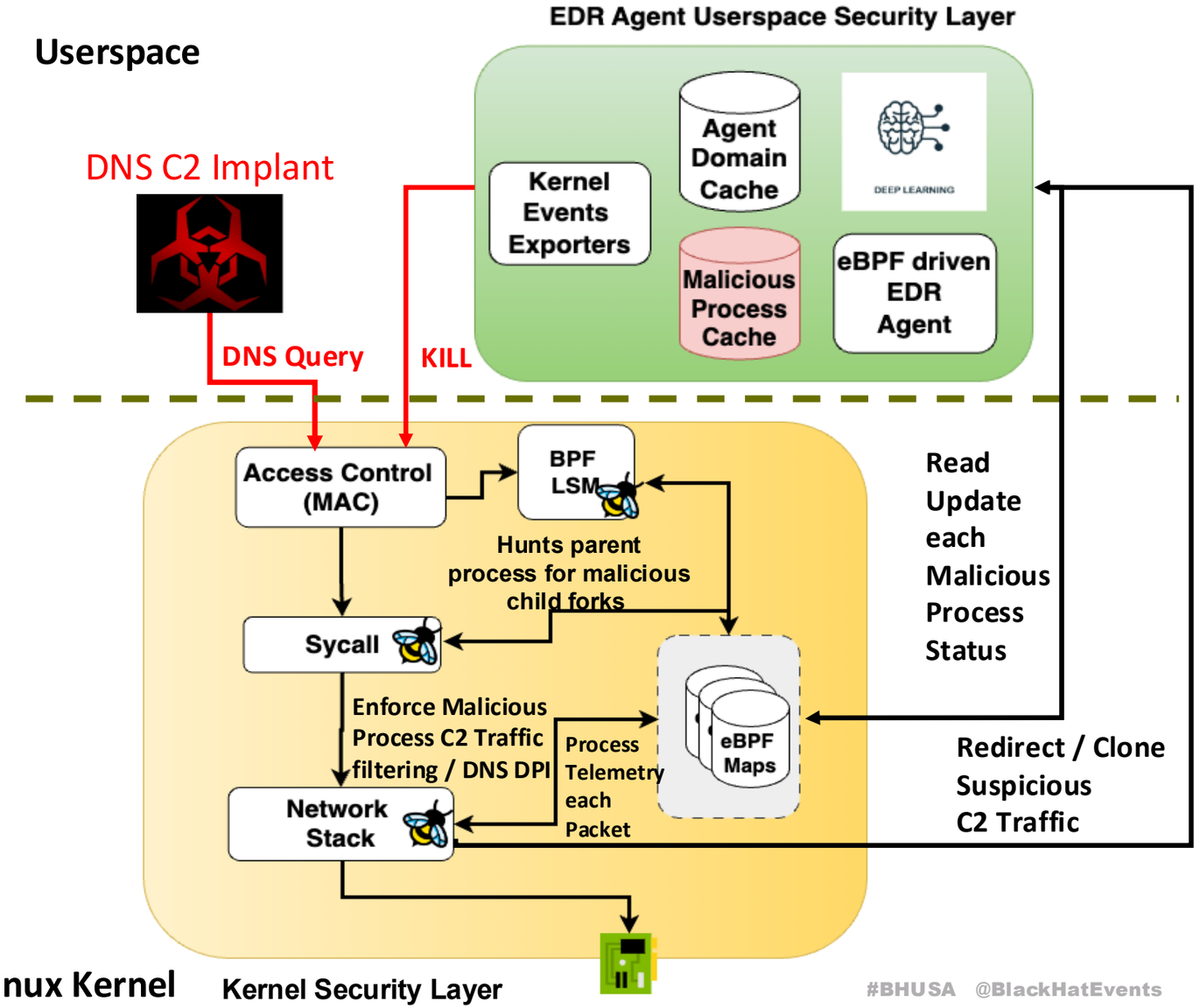
Continuous Security Enforcement Loop

Userspace

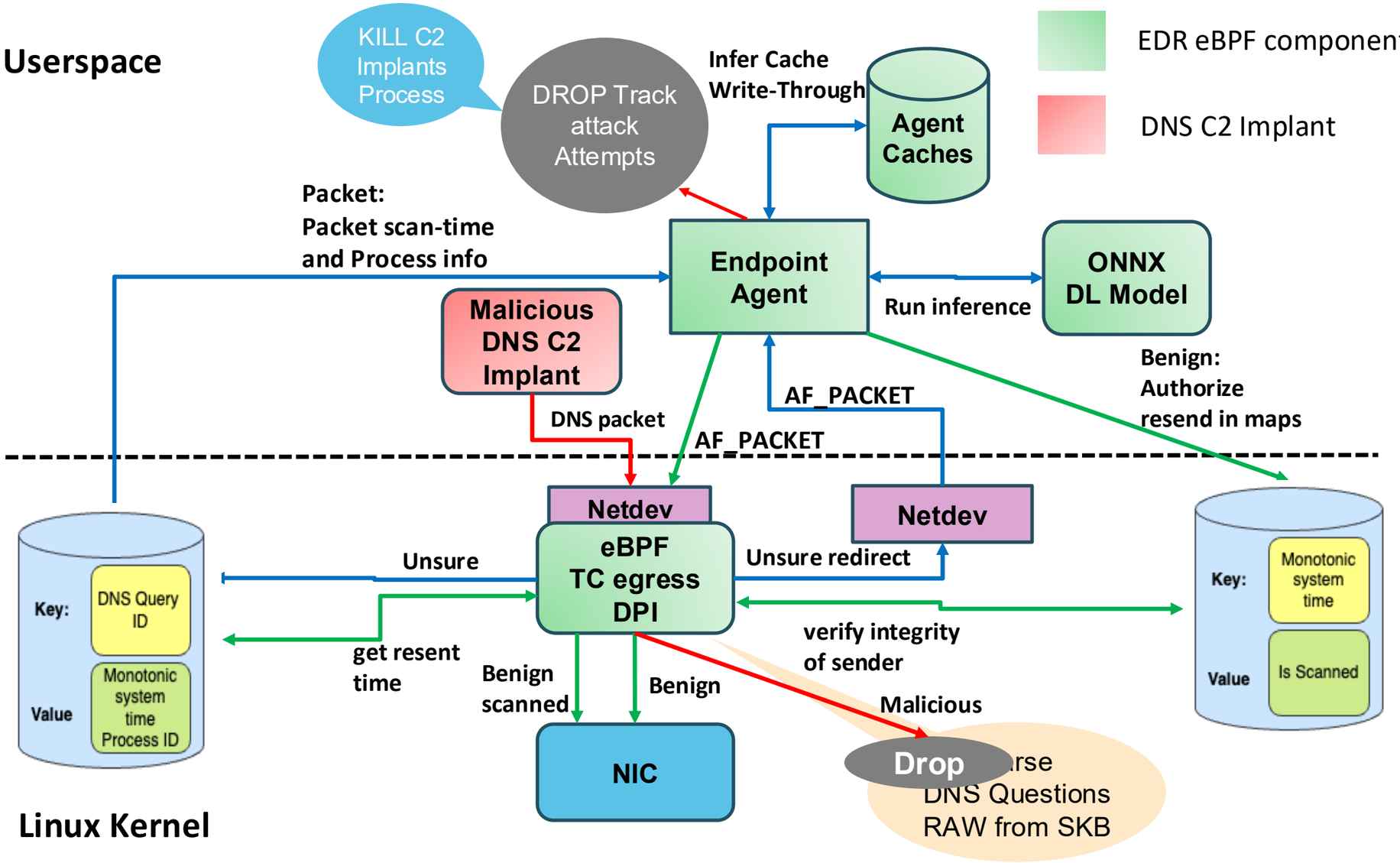
- eBPF Agent
- eBPF Agent Caches
- Quantized Deep Learning Model
- Events malicious metrics exporters

Linux Kernel

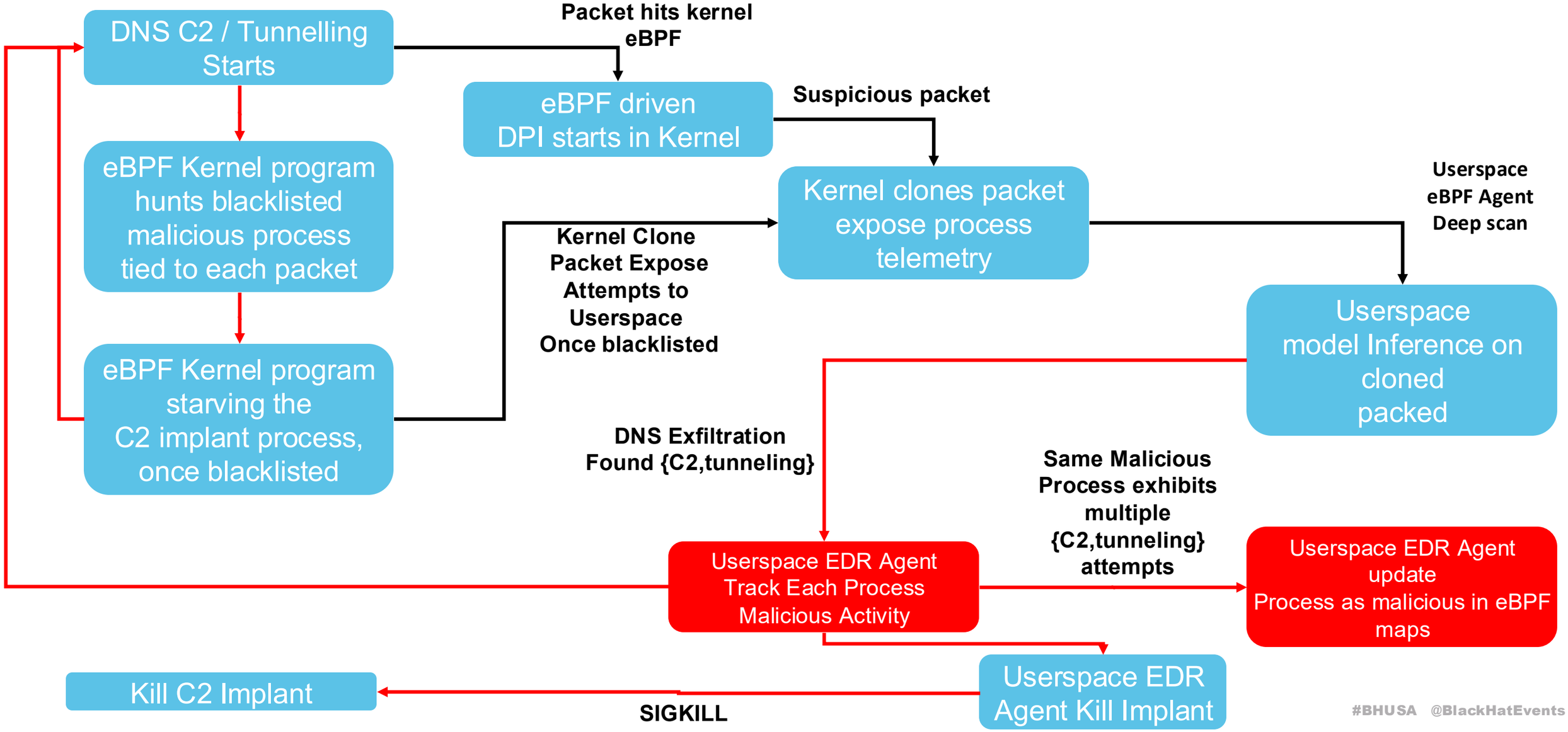
- eBPF Ring Buffers
- Access Control Layer (LSM)
- Syscall Layer (Tracepoints)
- Network Stack (TC, Sockets)



EDR Active Process Security Enforcement



EDR Agent Passive Process Security Enforcement



DNN based DNS Data Obfuscation Detection (Features)

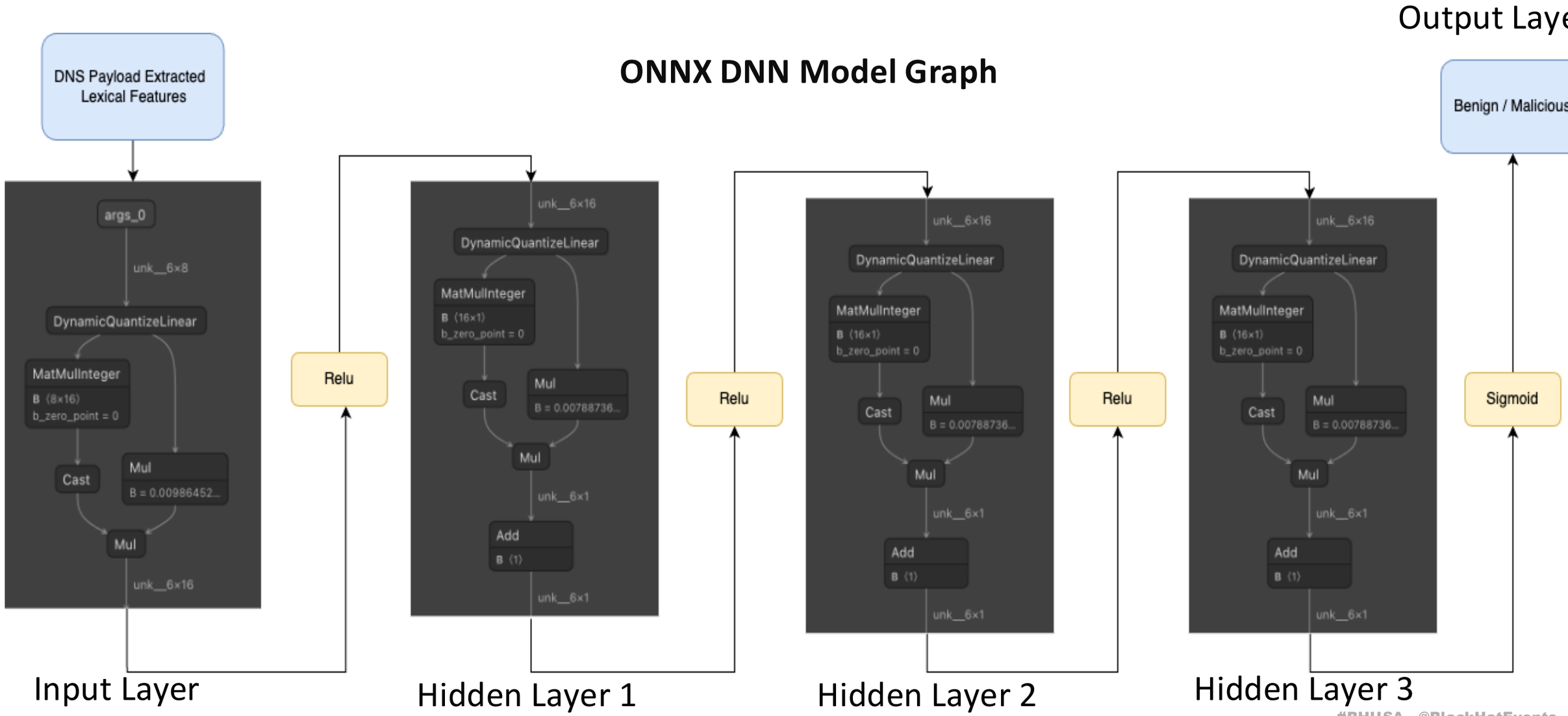
- ❑ Kernel Features
- ❑ Limits for DPI in Kernel

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).

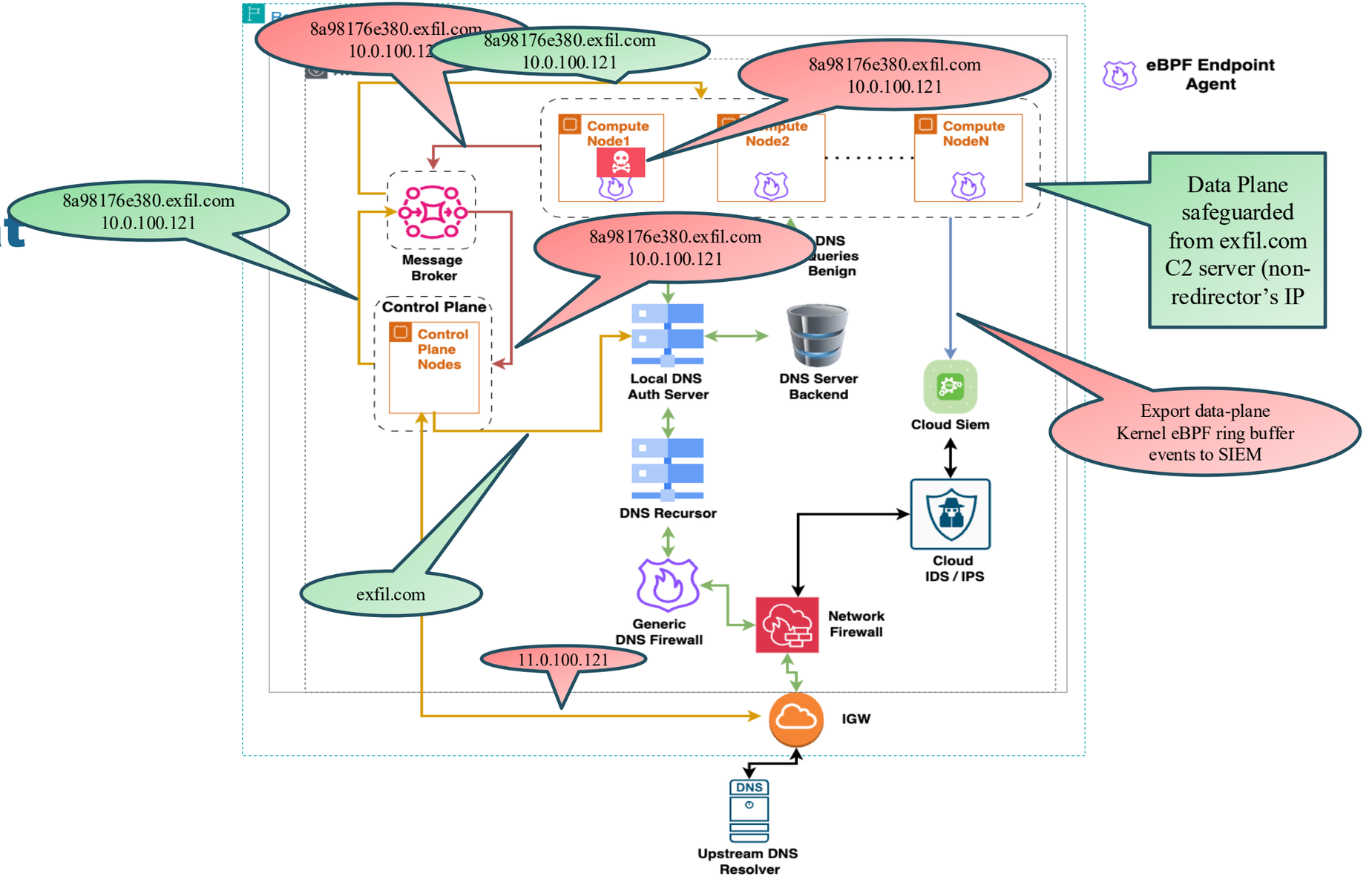
- ❑ Userspace Features
- ❑ Enhanced Lexical Features

Feature	Description
total_dots	Total number of dots (periods) in DNS query.
total_chars	Total number of characters in DNS query, excluding periods.
total_chars_subdomain	Number of characters in the subdomain portion only.
number	Count of numeric digits in DNS query.
upper	Count of uppercase letters in DNS query.
max_label_length	Maximum label (segment) length in DNS query.
labels_average	Average label length across the request.
entropy	Shannon entropy of the DNS query, indicating randomness.

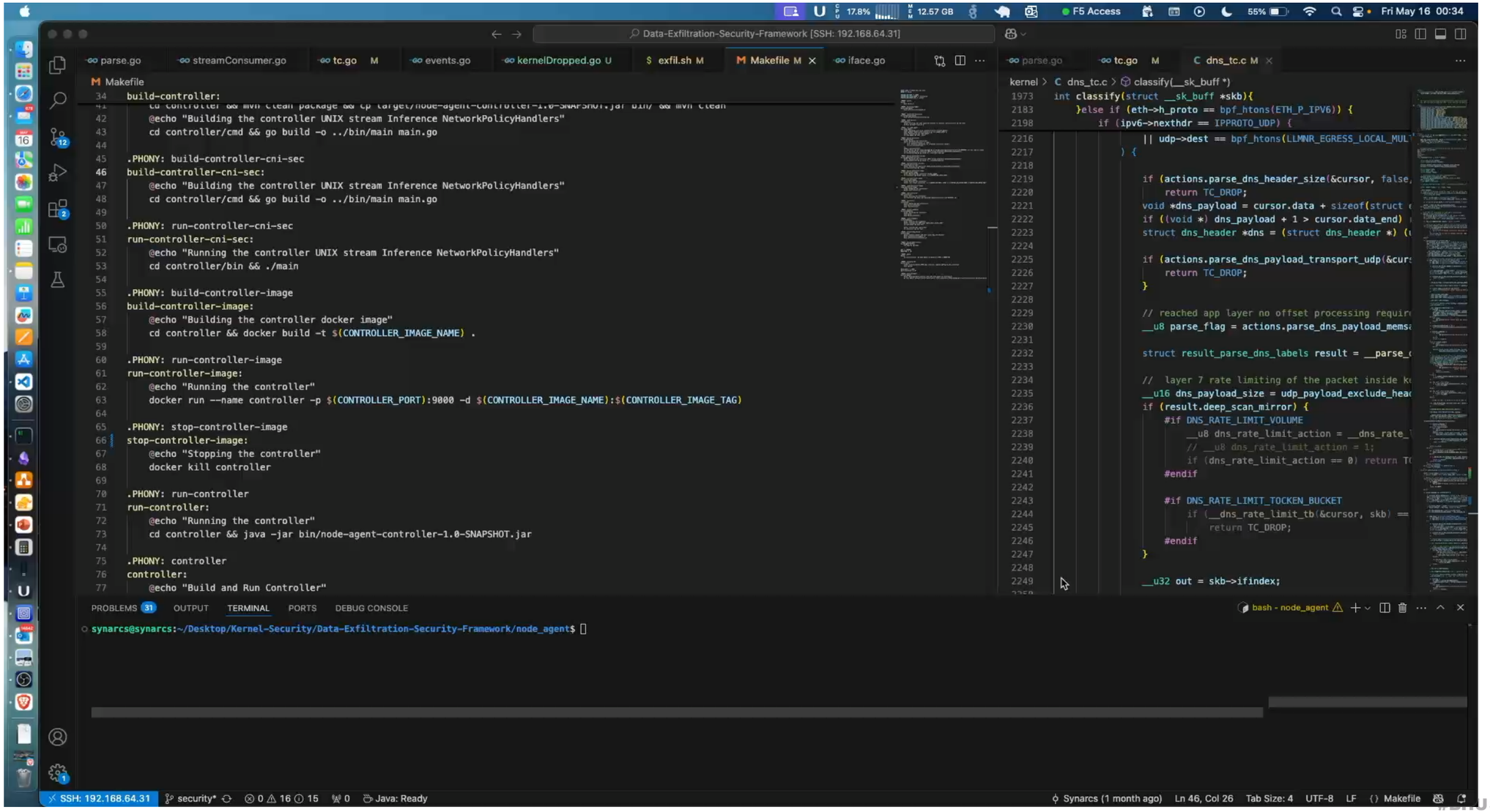
DNN fueled DNS Data Obfuscation Detection Model



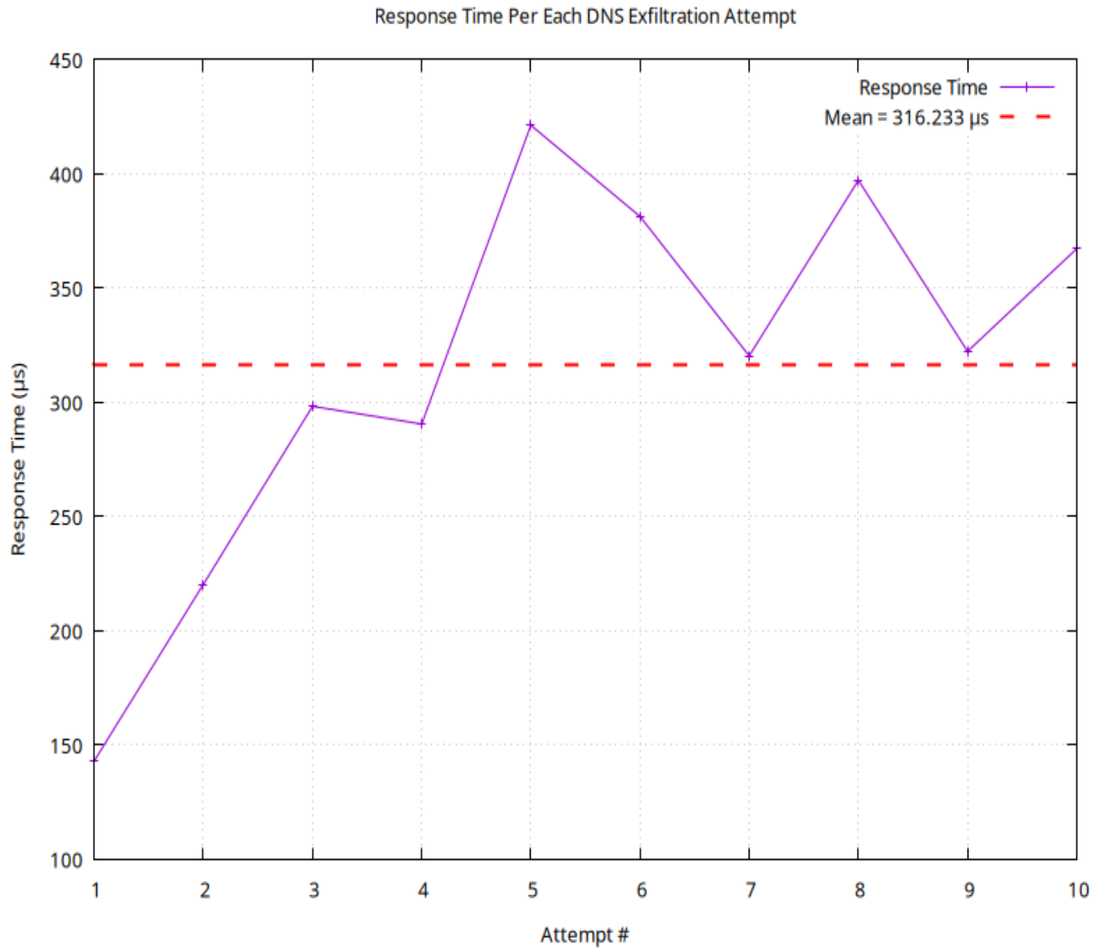
Framework Deployment in Cloud to Combat DNS C2 Infrastructure

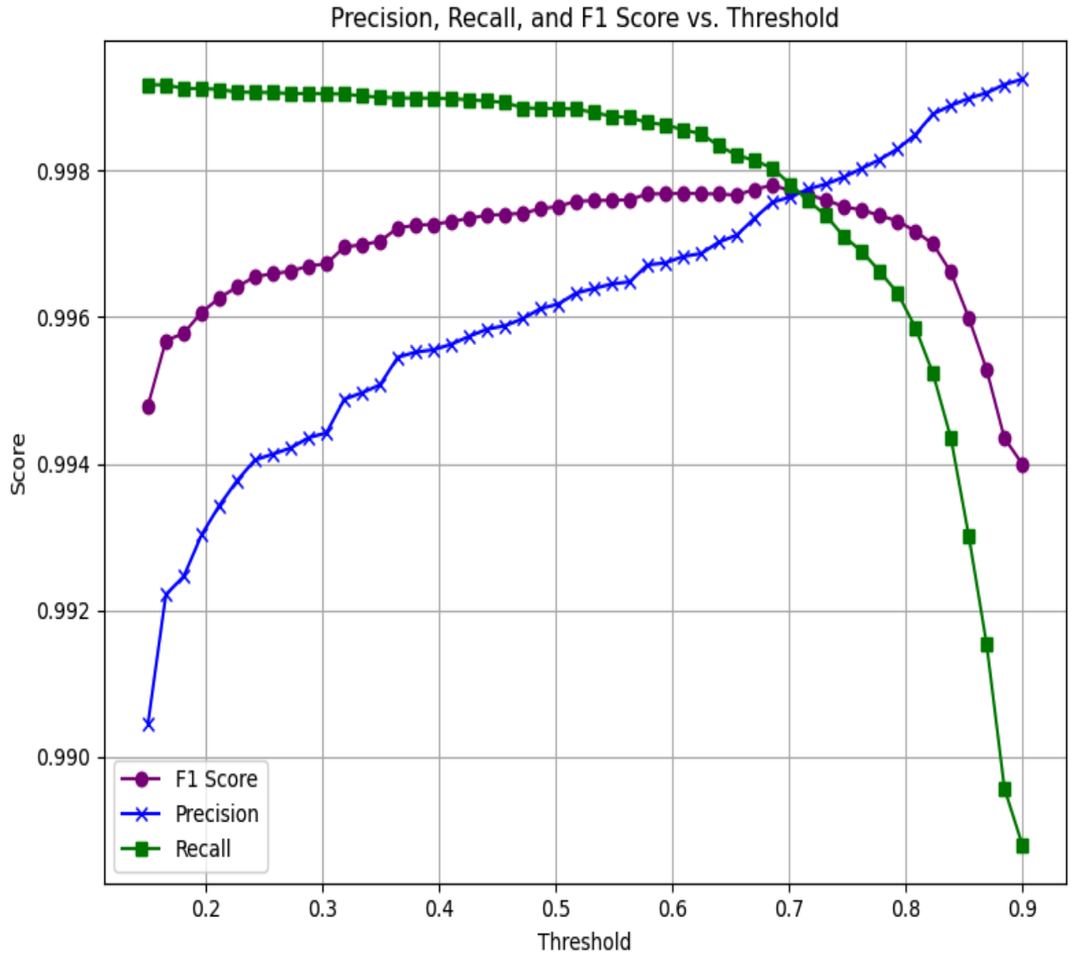


Demo


Data-Exfiltration-Security-Framework [SSH: 192.168.64.31]
parse.go
streamConsumer.go
tc.go M
events.go
kernelDropped.go U
exfil.sh M
Makefile M
iface.go
Makefile
build-controller:
@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"
parse.go
tc.go M
dns_tc.c M
kernel > dns_tc.c > classify(__sk_buff *)
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(&cur
return TC_DROP;
}
// reached app layer no offset processing requir
__u8 parse_flag = actions.parse_dns_payload_mems
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;
PROBLEMS 31
OUTPUT
TERMINAL
PORTS
DEBUG CONSOLE
bash - node_agent
synarcs@synarcs:~/Desktop/Kernel-Security/Data-Exfiltration-Security-Framework/node_agent$
SSH: 192.168.64.31
security*
Java: Ready
Synarcs (1 month ago)
Ln 46, Col 26
Tab Size: 4
UTF-8
LF
Makefile
F5 Access
Fri May 16 00:34

Response Speed with Precision





Next Steps

- **Kernel TLS Fingerprinting and Encrypted Tunnels**: eBPF for TLS fingerprinting to detect, hunt and kill exfiltration over TLS and kernel encapsulated traffic (wireguard).
- **Advanced Intelligence, Process Correlation:** eBPF kernel program and endpoint agent correlate cross-protocol exfiltration attempts to the originating process and block them.
- **AI-Driven Model Evolution**: Real-time drift detection, online learning, and confidence-based updates and deeper kernel behavior with GAN+LSTM emerging DNS obfuscation tactics.
- **eBPF Endpoint Agent a built-in guard for DNS DDoS attacks:** DNS NXDOMAIN flood at endpoint.

Black Hat Sound Bytes

- **Real-Time Kernel Threat Hunting & EDR Boost**: Hunt C2 implants dynamically in-kernel, accelerating user-space EDR with precise signals to stop C2 and breaches.
- **AI-Driven Kernel Enforcement**: Pair AI with eBPF to adaptively reprogram the kernel to combat mutating C2 implant activity.
- **Dynamic Kernel powered EDR fuels Cloud NACL's**: Enforce L3 filters at the endpoint and sync with cloud firewalls to disrupt DGA and evolving C2 infrastructure.
- **Deep OS Telemetry powers SIEM/SOAR**: Kernel-powered visibility feeds rich behavioral signals into upstream SIEM, SOAR.

Thank You

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

Framework WhitePaper: https://shorturl.at/42dVC