Bringing Situational Awareness into Zeek

Leveraging the Input Framework for Enriched Intrusion Detection

Aashish Sharma / Partha Banerjee

Cyber Security • Lawrence Berkeley National Laboratory

Zeek Workshop @ CERN, Geneva • March 2026

U.S. DEPARTMENT OF ENERGY

UNIVERSITY OF CALIFORNIA

About Me & The Berkeley Lab

- Berkeley Lab: Established in 1932, a DoE national laboratory, UC managed, open-science mission
- ~7,000 employees, ~20,000 devices, ESnet connected, 3x100G+ backbone, Wi-Fi, VPN, Cloud, …
- Birthplace of tcpdump, libpcap, trace route, Zeek …. Cyclotron ;)
- Home of the 1986 Cliff Stoll/Hanover hacker incident — **network security is in our DNA**
- The challenge: researchers need open access, intrusions are reality of life
- LBNL Cybersecurity team — running Zeek since 1995

The Network Reality

Multiple buildings, labs, guest networks
Researchers come and go — high churn
Collaborators from 100's of institutions
Cloud workloads in AWS
We can't just block things (be it services, devices, people or network)

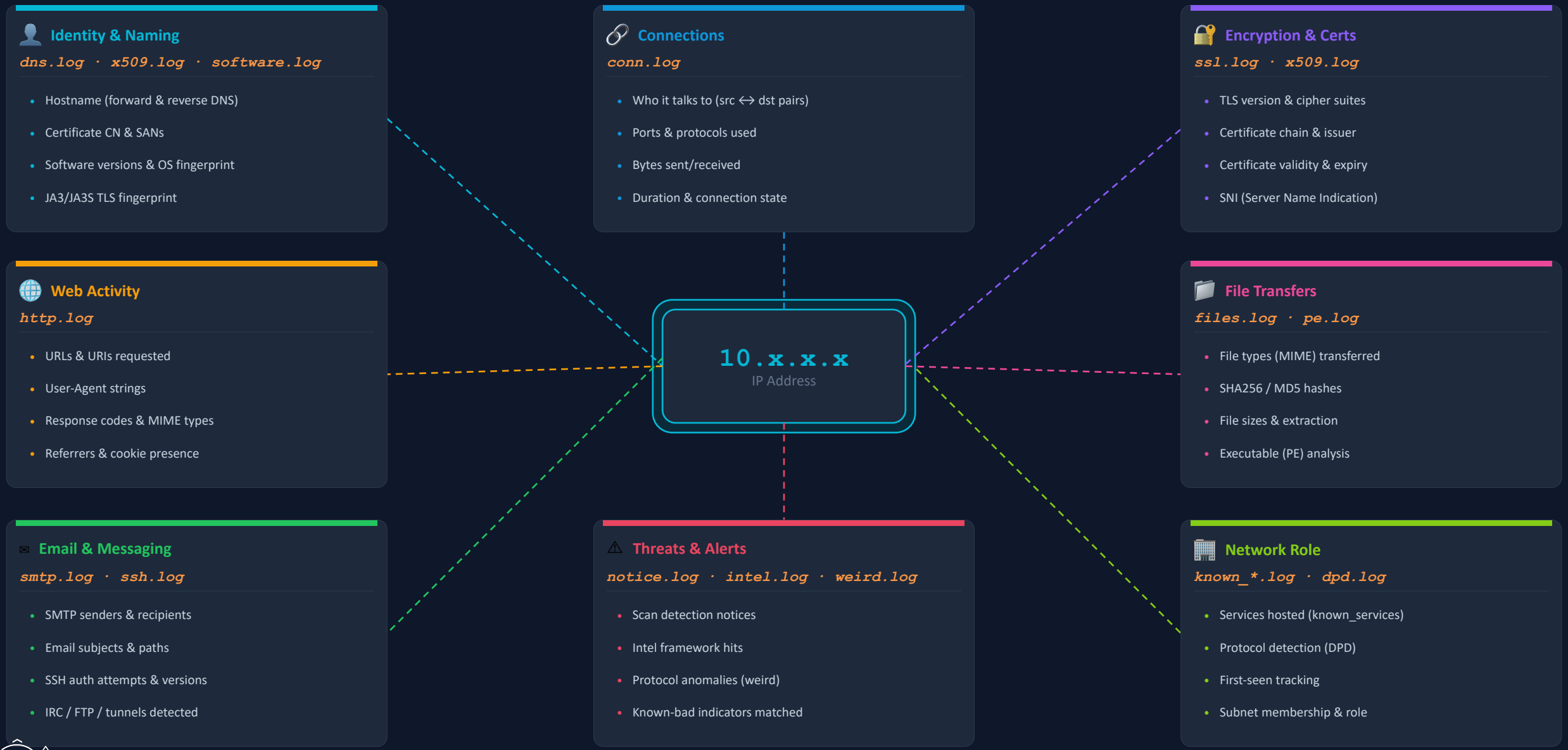
The Zeek Deployment

Zeek cluster: Many clusters put in strategic locations for greater visibility
50+ external feeds via Input Framework
Many custom security Zeek-packages
200+ notice types, escalation policies

Running in production for 30+ years now

Lawrence Berkeley National Laboratory

Zeek is a powerful tool - it can tell things about an IP*



BERKELEY LAB
Bringing Science Solutions to the World

**Every connection, every protocol, every file — Zeek turns raw packets into structured, queryable intelligence.*

The Problem: Zeek's Blind Spots

Zeek generates incredible data — conn.log, dns.log, http.log, ssl.log, smtp.log, ntp.log, ldap.log, ssh.log … … … …

But raw traffic analysis alone don't directly answer the questions that matter in operations:

- Is this an employee laptop or a visitor's device or something rogue ?
- Who owns this IP ?
- Is this subnet an allocated subnet, unallocated one, or a honeypot?
- Is this remote IP a known TOR exit node or on a blacklist?
- Is this a legitimate scanner or an attacker?

ZEEK

conn.log says:
"10.1.2.3 → 131.243.x.y:445"

No user context, no subnet role,
no asset info, no threat context ..

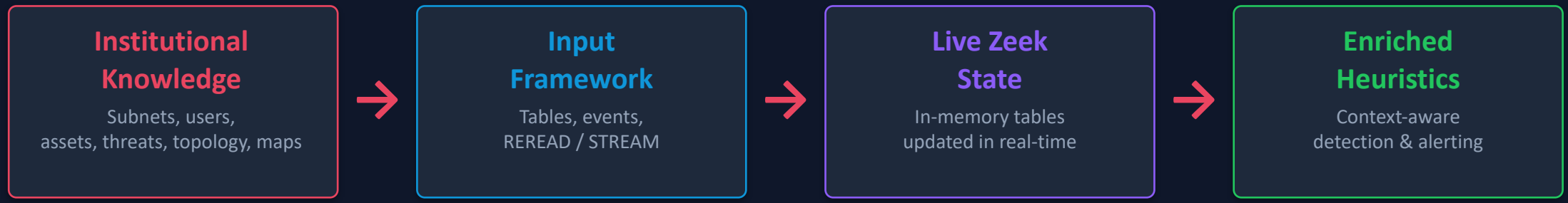
SITUATIONAL AWARE ZEEK

Same flow, enriched:
"TOR exit node → John Smith's workstation
(Physics Dept, Building 110, CrowdStrike agent missing)
SMB/445 – PrintNightmare indicator"

Actionable. Contextual. Prioritized.

The Solution: Feed Knowledge into Zeek

Every organization has deep institutional knowledge* about its network, assets, users, and threats. This context turns Zeek from a logging engine into an analyst.



*make Zeek familiar with institutional knowledge

Building Blocks for a Living Network Map

For any IP on the network, you can know:

User Identity/Mapping

User identity
Groups/Teams/Department
Physical locations
office allocations

Asset inventory

Software & Version Intelligence
Network Services & Infrastructure
DNS & Name Resolution
Vulnerability Context
EDR Enrollment Check

Network Topology

Subnet allocations & Addressing
Security Zones & Enforcement Points
Institutional Segments & Special Zones

Threat Intelligence

Known malicious IPs
C2 server addresses
Malware hashes (files, TLS certs)
Suspicious domains
Threat actor infrastructure

Geographic contexts

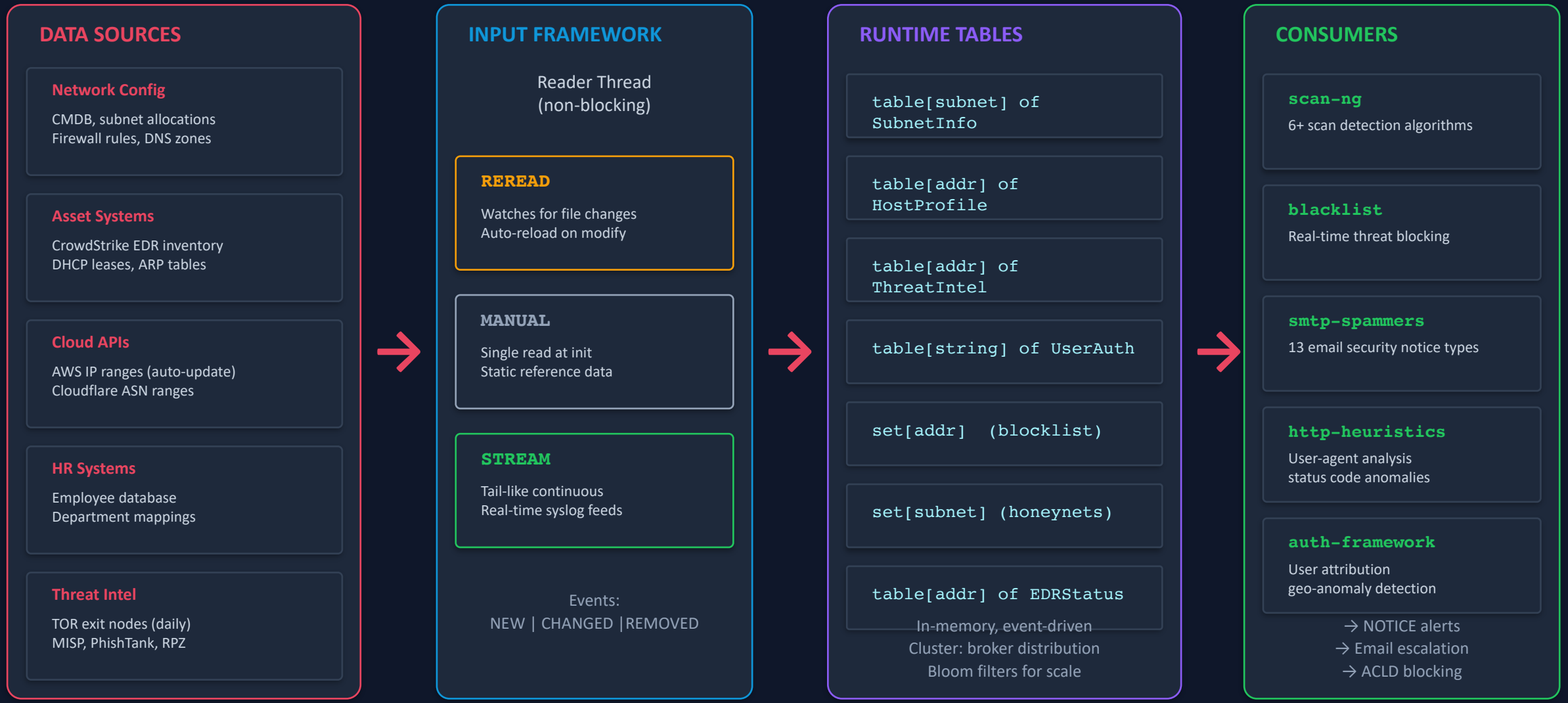
Spread of your user base
Travel / hub airports
Some ASN's which are just bad - IP2ASN

Beyond Perimeter

Cloud Infrastructure
External Accessors
Neighbor Nets

Sub-modules: dhcp.zeek • host-profiling.zeek • mobile-device.zeek • read-awquery • weird-conn.zeek • conn-remote-users.zeek

The machinery: Input Framework



Know Your Network: Your Uniform Resource Tree (YURT)™

14 auto-generated feeds define our network reality — every connection gets classified

YURT::ALLOCATED_NETS	Your address space breakdown / subnets.txt
YURT::BOGON	Non routable/reserved IPs
YURT::HONEY_NET	Honeypot/decoy ranges
YURT::DHCP_HOSTS	Dynamic IP pools
YURT::EMP_WIFI	Employee WiFi networks
YURT::AWS	Cloud infrastructure (auto-updated)
YURT::CLOUDFLARE	CDN/proxy ranges

YURT::NAMESERVERS	Internal DNS servers
YURT::NEIGHBOR_NETS	Peering institution networks
YURT::NEVER_DROP_NETS	Critical infrastructure whitelist
YURT::RFC1918	Private address space
YURT::LBL_VPN	VPN infrastructure
YURT::SCAN_NETS	Authorized scanner subnets
YURT::CrowdStrike	EDR asset inventory

Classification Output

Internal (ALLOCATED) | Cloud (AWS, CF) | VPN | Honeypot | Darknet (BOGON)

Some examples of what these feeds enable.

Let's zoom out and look at the full picture.

Zeek now knows your network.

What can it actually do with that knowledge?

"Who is behind that IP?"

Identity enrichment

"Is this scan real?"

Scan detection through feeds

"Block it — now, no restart"

Living blacklists

"Is this known bad?"

Threat intelligence

"Is this email a threat?"

Email security pipeline

"Is this machine compromised?"

Beaconing & honeynets

Know Your Network: Leveraging Your Uniform Resource Tree (YURT)

Auto-generated feeds define our network reality — every Site::local_nets() IP address gets classified

ARPwatch | CrowdStrike | Network Topology & Subnet info | Honeypot | DNS Zone Transfers | MAC_OUI

```
$ column -t -s $'\t' /Users/aashish/_work/zeek-feeds/anonymized-log-sample.tsv

ts                mac                mac_random  ip_type          host           network          cs_installed  dns          vendorName          use
1765311940.751512  c8:4b:d6:7a:e2:91  F           Site::DHCP       172.24.87.78   172.24.86.0/23   F             host-7094    Dell Inc.           Building 12: Floor 3
1765311940.751512  50:7c:6f:3d:c8:f4  F           Site::STATIC     198.51.44.100  198.51.44.64/26  F             gw-100       Intel Corporate     ScienceDMZ NAT Pool
1765311940.751524  00:50:56:6b:a3:17  F           Site::VPN        203.0.172.94   203.0.160.0/20   T             jdoe-w51     VMware, Inc.        VPN Service
1765311940.751524  8c:ec:4b:2e:d7:43  F           Site::DHCP       192.0.55.170   192.0.52.0/22    F             pluto        Dell Inc.           Building 7,8,9
1765311940.751524  00:50:56:c4:52:e9  F           Site::STATIC     172.30.18.171  172.30.18.0/23   T             svc-vendor   VMware, Inc.        Building 3A-201: CoLo (B)
1765311940.751524  a6:f1:4b:9c:72:e3  T           Site::DHCP       198.18.207.172 198.18.204.0/22  T             asmith-w49   (randomized)        Building 14,15
1765311940.751526  b4:96:91:8c:41:a7  F           Site::DHCP       172.22.91.120  172.22.91.0/24   T             bjones-w57   Intel Corporate     Building 11,11A,5
```

Know Your Network: Leveraging Your Uniform Resource Tree (YURT)

Auto-generated feeds define our network reality — every Site::local_nets() IP address gets classified

ARPwatch | CrowdStrike | Network Topology & Subnet info | Honeypot | DNS Zone Transfers | MAC_OUI

```
$ column -t -s $'\t' /Users/aashish/_work/zeek-feeds/anonymized-log-sample.tsv

ts              mac               mac_random  ip_type       host            network          cs_installed  dns          vendorName          use
1765311940.751512  c8:4b:d6:7a:e2:91  F           Site::DHCP    172.24.87.78    172.24.86.0/23   F             host-7094    Dell Inc.           Building 12: Floor 3
1765311940.751512  50:7c:6f:3d:c8:f4  F           Site::STATIC  198.51.44.100   198.51.44.64/26  F             gw-100       Intel Corporate     ScienceDMZ NAT Pool
1765311940.751524  00:50:56:6b:a3:17  F           Site::VPN     203.0.172.94    203.0.160.0/20   T             jdoe-w51     VMware, Inc.        VPN Service
1765311940.751524  8c:ec:4b:2e:d7:43  F           Site::DHCP    192.0.55.170    192.0.52.0/22    F             pluto        Dell Inc.           Building 7,8,9
1765311940.751524  00:50:56:c4:52:e9  F           Site::STATIC  172.30.18.171   172.30.18.0/23   T             svc-vendor   VMware, Inc.        Building 3A-201: CoLo (B)
1765311940.751524  a6:f1:4b:9c:72:e3  T           Site::DHCP    198.18.207.172  198.18.204.0/22  T             asmith-w49   (randomized)        Building 14,15
1765311940.751526  b4:96:91:8c:41:a7  F           Site::DHCP    172.22.91.120   172.22.91.0/24   T             bjones-w57   Intel Corporate     Building 11,11A,5
```

Heuristics

Notice Type	Trigger	Severity
EDR_Disappeared	cs_installed T → F	Critical — endpoint agent removed/killed
MAC_Changed_For_IP	Same IP, different MAC	High — ARP spoofing or IP conflict
MAC_Changed_For_Hostname	Same hostname, different MAC	High — device impersonation
Vendor_Changed_For_MAC	MAC changed AND vendor OUI changed	Critical — definitive MAC spoofing
IPType_Changed	ip_type transitions (STATIC→DHCP, etc.)	Varies — with severity annotation per transition
RandomMAC_On_Static	Privacy MAC on a STATIC host	High — servers should never have random MACs
MAC_On_Multiple_Subnets	Same MAC on different subnets	Medium — dual-homed, bridging, or NAC bypass
VM_On_WiFi	VMware vendor on EMP_WiFi	Medium — unauthorized VM on wireless
DHCP_In_Datacenter	DHCP device in CoLo/server-room subnet	Medium — rogue device in datacenter
Managed_Host_Hit_Honeypot	cs_installed=T + ip_type=HONEYNET	Critical — compromised managed endpoint
Rogue_Device	Active IP not in ARP, DHCP, or DNS feeds	High — unknown device on network

Who Is Behind That IP?

Inside Zeek identity enrichment turns anonymous IPs into named, attributed entities

Identity Feeds

- auth.log — STREAM mode, real-time auth events from AuthFramework
- IPs_Users_Authenticated_from — VPN/SSH user ↔ IP mapping
- LBL-Employee-info — HR database: name, department, role
- LBL-Crowdq-Report — CrowdStrike EDR enrollment status
- MAC_VENDOR_OUI — Device manufacturer identification
- syslog-dhcp-sec.feed — DHCP lease tracking

Notice can Include

- User name & department
- Device hostname & MAC vendor
- CrowdStrike enrollment status
- DHCP vs. static assignment
- Authentication history
- Geographic location (GeoIP)

Auth Framework Architecture

Kerberos	kerberos-auth-data.zeek	**RADIUS**	radius-auth-data.zeek	**RDP**	rdp-auth-data.zeek
Google Auth	Google Authentication data	**SSH**	SSH auth from syslog	**VPN**	VPN + other MISC auth
Step-Stone	Multi-hop attack detection	**GeoIP**	unusual travels	**Watch Users**	Targeted user monitoring

Addressing Operational Issues: Smarter Scan Detection Through Feeds

THE CHALLENGE

Why raw scan detection breaks

- Scan detection thrives on state — but maintaining state is fragile in a dynamic system
- Dynamic components (NATs, DHCP, ISPs) cause occasional false positives but need time bound response
- Restarts required but disruptive — kills in-flight detections
- Too many knobs need insider knowledge to tune

6 INPUT FEEDS

REREAD · no restarts

Hot-reloadable configuration — changes propagate in real-time

- **scan-portexclude** — Known services to skip
- **ip-whitelist.scan** — Authorized scanners (Nessus, Qualys, cloud scanner, pen-testing teams
- **subnet-whitelist.scan** — Institutional scanner subnets
- **knockknock.exceptions** — Port knock exclusions
- **WIRED.blocknet** — Blocked network ranges
- **LBL-subnets.csv** — Local subnet definitions & roles

DETECTION ENGINES

6 algorithms produce 7 notice types

- **TRW** — Threshold Random Walk
- **Address Scan** — Destination IP sweep detection
- **Port Scan** — Port flux per source
- **Landmine** — Responses from darknet addresses
- **Backscatter** — Reflection attack detection
- **Knock-Knock** — Sequential port probing patterns

NOTICE TYPES

`PortScan` `AddressScan` `KnockKnock` `Landmine` `Backscatter` `PasswordGuessing` `LikelyScanner`

Without feeds

"This IP is incorrectly blocked — it's someone's home IP"

- ✗ Manually remove the block from each Zeek instance
- ✗ Whitelist the IP by editing config files
- ✗ Restart Zeek across the board — killing in-flight state
- ✗ Only senior engineers can do this safely

With feeds

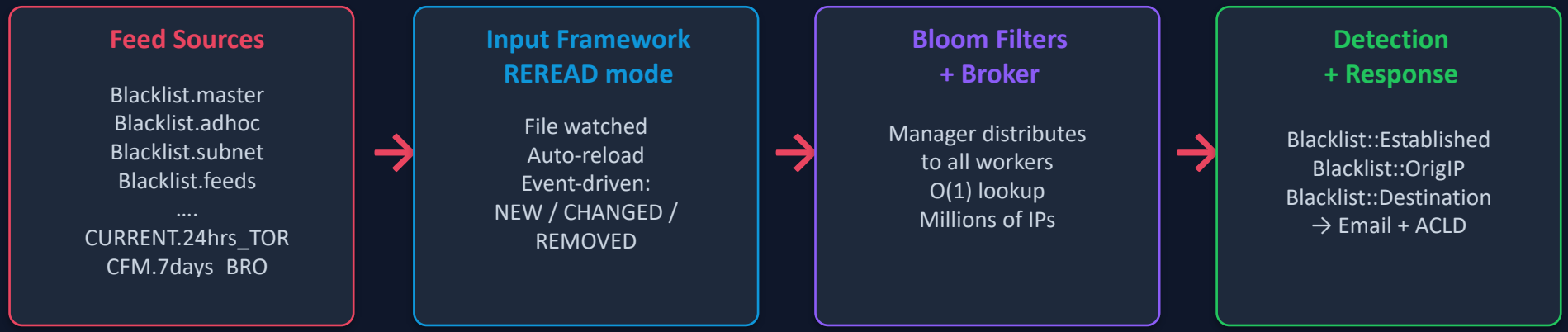
```
$ zeek-add-whitelist <IP>
```

- ✓ Anyone on the team can run it — no Zeek expertise needed
- ✓ Changes propagate across all deployments in real-time
- ✓ No restarts — zero impact on in-flight detections
- ✓ Audit trail of every whitelist change

Before we needed to restart zeek for every &redef change and one had to have Zeek background to do so.

Living Blacklists — Not Static Anymore

Dynamic threat blocking without Zeek restart. Add an IP to the feed, detection happens within seconds across the entire cluster.




```
event read_ip(desc: Input::TableDescription,
              tpe: Input::Event,
              left: BlacklistIPAddrIdx,
              right: BlacklistIPAddrVal) {
    if (tpe == Input::EVENT_NEW) { /* track */ }
    if (tpe == Input::EVENT_REMOVED) {
        delete blacklist[ip];
        NOTICE([$note=Removed, ...]);
    }
}
```

Why Dynamic Matters

- Analyst adds an IP → detection in seconds, no restart
- Feed removes an IP → suppression stops automatically
- Cluster-wide via Broker: manager → all workers
- Scales to millions via bloom filters

ts	ltype	duration	total	bl_ips	active_bl_ips	inactive_bl_ips	bl_nets	active_nets	active_nets_ips	inactive_nets
1773990552.872041	Blacklist::Summary	02-07:34:00	1926322	110994	576	110418	6715	3907	96342	2808
1773991152.872121	Blacklist::Summary	02-07:44:00	1926322	110994	576	110418	6715	3913	96454	2802
1773991752.872147	Blacklist::Summary	02-07:54:00	1926322	110994	576	110418	6715	3914	96565	2801
1773992352.872248	Blacklist::Summary	02-08:04:00	1926322	110994	576	110418	6715	3916	96688	2799
1773992952.872353	Blacklist::Summary	02-08:14:00	1926322	110994	576	110418	6715	3916	96825	2799
1773993552.872437	Blacklist::Summary	02-08:24:00	1926322	110994	577	110417	6715	3921	96931	2794

When It Mattered Most: Log4j

CVE-2021-44228 — a real-world test of the feed architecture

Friday Night

⚠ **Log4j zero-day drops**

Public disclosure of CVE-2021-44228.
Every Java app with Log4j is potentially vulnerable.

Saturday 2 AM

IOCs added to feed files

Known exploit callback domains, IPs, and
JNDI lookup patterns pushed to TSV feeds.

Saturday 3 AM

Zeek REREADs — detection live

All clusters pick up the new feeds automatically.
No code changes. No restarts. No deploys.

Saturday Morning

✓ **First detections fire**

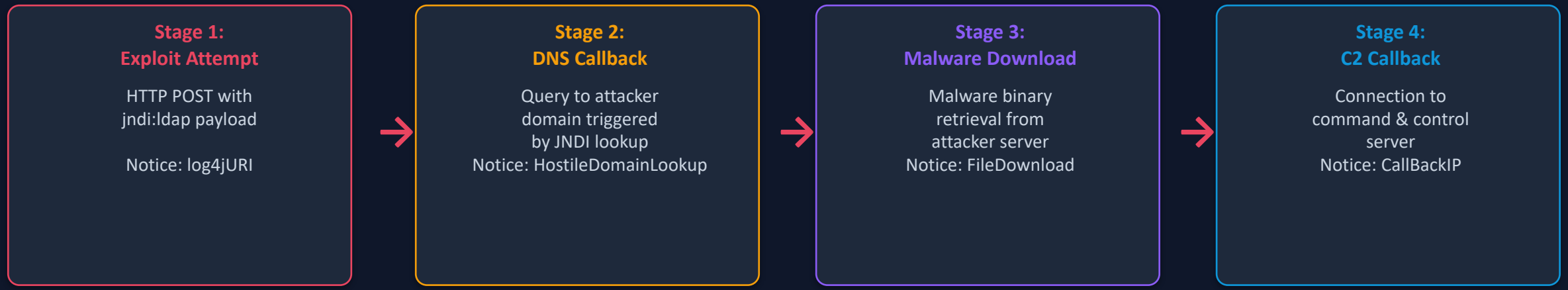
Zeek matches Log4j callback patterns in HTTP
headers and DNS queries across the network.

Total time from zero-day to full detection: **< 6 hours** • Code changes: **0** • Restarts: **0** • People needed: **1**

"When Log4j dropped on a Friday night, we had detection running in hours — just push IOCs into the feed files."

Response: When Speed Matters

Log4Shell (CVE-2021-44228) — Multi-Stage Detection



Active CVE Detection Packages

PrintNightmare	CVE-2021-1675	DCE-RPC spoolss operations
Bad Neighbor	CVE-2020-16898	Malformed ICMPv6 Router Advertisement
Struts	CVE-2017-5638	Apache Struts content-type injection
WinPHP	CVE-2024-4577	PHP CGI argument injection on Windows
Shellshock	CVE-2014-6271	Bash function injection in HTTP headers
CUPS RCE	CVE-2024	CUPS printing system remote code exec
SIGRed	CVE-2020-1350	Windows DNS Server RCE

The Big Picture: What Situational Awareness Gives You

"An alert without context is noise. An alert with identity, device info, and role — that's actionable."

BEFORE: Plain Zeek

```
conn.log entry:
10.1.2.3 → 131.243.X.Y:445
```

That's it. An IP. A port. A timestamp.
No context. No priority. No action.

Analysts drown in alerts they
can't triage effectively.

AFTER: Situationally Aware Zeek

Source → TOR exit node 1.2.3.4 [added to the feed 3 days ago]

Host - PHYSICS-LAB-PC01

Subject John Smith

Location → Building 50, Physics Dept

Posture → CrowdStrike: agent MISSING

Detection → SMB/445 — PrintNightmare DCE-RPC indicator on spool5

Actionable. Contextual. Prioritized.

The Input Framework runs in a separate thread. Zero impact on packet processing.

Lessons from Years of Feed Integration with Input Framework

Start small, validate before adding more

Don't try to build 50 feeds overnight. Start with subnet awareness, add one feed at a time, and validate each one produces meaningful detection before moving on.

REREAD mode is your best friend

Most feeds benefit from REREAD. Your threat landscape changes constantly — your Zeek tables should too, without requiring restarts.

Automate feed generation

Use scripts and cron to keep feeds fresh. Manual feeds go stale. Our feeds are generated by scripts pulling from DHCP, DNS, CrowdStrike, HR systems, and other SoA's

Always pair detections with exclusion feeds

Every detection heuristic needs a corresponding whitelist/exception feed. This is how you achieve low false-positive rates.

Cluster awareness matters

Input Framework is per-node. Use @if(is_manager) for large feeds and Broker for distribution. Bloom filters handle scale.

Test with real traffic before deployment

What looks good in dev may behave differently at 100Gbps. Validate feed sizes, lookup performance, and memory consumption.

Operational Challenges & War Stories

What they don't tell you in the documentation

The Empty Feed File Incident

A cron job glitched and truncated our blacklist feed to zero bytes. Zeek dutifully re-read it — identified that this is less than expected data so did not update its tables internally. Lesson: always validate feed file size before publishing.

Monitoring the Monitors

Who watches the feed pipeline? We wrote a meta-feed that tracks last-modified timestamps of every other feed. If a feed goes stale for >24h, we get a NOTICE. Because the worst feeds bug is the one where nothing happens.

Cluster Distribution

Workers don't share state. A feed loaded on the manager isn't visible to workers by default. We built a pattern: manager reads feeds, generates Broker events, workers populate local tables. Sounds simple — took three iterations to get right.

Memory at Scale

Loading 500K+ entries into Zeek tables works — until you multiply by 32 worker nodes on a cluster. We learned to keep large feeds on the manager only and distribute via Broker events. Bloom filters saved us on the IP2ASN dataset.

Bug that took 8+ years to show up

Darknet reading subnets.txt file still blocked CERN's IPs on allocated nets - race condition took 8+ years to manifest.

"The feeds that cause the most damage aren't the wrong ones — they're the ones that silently stop updating."

How to Start Building Situational Awareness

1

Know Your network

Map your network topology.
Which subnets are allocated?
Which are darknet?

2

Pick One Data Source

Start with something you already have: DHCP logs, DNS zones, ARP tables.

3

Create a TSV Feed

Add #fields header line.
Tab-separated values.
One record per line.

4

Write 10 Lines of Zeek

Input::add_table() with REREAD mode. Define Idx and Val record types.

5

Use Data in Events

Look up the table in connection_established or other event handlers.

6

Generate NOTICE

Create actionable alerts with context from feeds.
Iterate and add more!

"Start with subnets. Add DHCP. Then DNS zones. Before you know it, Zeek knows your network better than you do."

Know Your Network

Four layers of institutional intelligence — assembled before the first packet arrives

① Foundation

What space do you own?

- Network Topology & Addressing
- Security Zones & Honeynets
- Institutional Segments

Know your CIDR blocks, allocated vs darknet, BOGON space, DHCP scope, and enclave boundaries. Any traffic touching unallocated space is an instant alert.

② Identity

Who and what is on the network?

- Asset Inventory & EDR Coverage
- DNS & Name Resolution
- Directory & Employee Data

Marry IPs to people and devices. DHCP + LDAP + EDR gives you: this IP = Jane Smith's laptop, managed, Building 50, Physics Dept. Not just an IP.

③ Infrastructure

What services are running?

- Network Services (VPN, NTP, RADIUS)
- Device Classes (Printers, NAS, IoT)
- Software & Version Inventory

Know your VPN endpoints, mail servers, and NTP nodes. Track legacy OS (Windows XP/7), open RDP/SSH, and which printers are making outbound calls.

④ External

What's beyond your perimeter?

- Cloud & CDN Ranges (AWS, GCP, Cloudflare)
- Authorized Scanners & Assessors
- Partner & Peer Networks

Enumerate cloud provider CIDRs so bulk transfers are contextual, not alarming. Whitelist DHS/Censys so authorized scans don't flood your SOC.

Know Your Network

Normalized taxonomy of network intelligence objects — consolidated from 110+ raw feed names

10	70	14	50+	200+
Categories	Unique Feeds (from 110+ raw)	Auto-generated YURT objects	Active Feeds	Notice Types

① FOUNDATION — WHAT SPACE DO YOU OWN?

Network Topology & Addressing

Allocated blocks, dynamic pools, segment boundaries

8 feeds

DETECTION OUTCOME → **Darknet hits, misrouted traffic, address spoofing**

`YURT::ALLOCATED_NETS` `YURT::BOGON` `YURT::RFC1918` `DHCP_SCOPE ×3` `WIRED ×3` `WIRELESS ×3` `SCAN_NETS ×3` `NEVER_DROP_NETS`

Zeek heuristic: Traffic to/from unallocated or BOGON space → HONEY_NET or darknet alert. DHCP scope tells you if an IP should exist at all.

Security Zones & Enforcement Points

Honeynets, darknets, firewalls, perimeter devices

7 feeds

DETECTION OUTCOME → **Unauthorized traversal, perimeter bypass, honeynet contact**

`YURT::HONEY_NET` `DARKNET` `FIREWALL_SUBNETS ×2` `External Pentesters ×3` `PFSENSE` `JUNIPER_SRX` `ROUTER_INTERFACES ×2`

Zeek heuristic: Any connection to HONEY_NET or DARKNET space is unconditionally suspicious — legitimate traffic never goes there. Zero false-positive rate.

Institutional Segments & Special Zones

Facility-specific subnets, guest networks, critical enclaves

6 feeds

DETECTION OUTCOME → **Lateral movement between enclaves, guest-to-internal pivots**

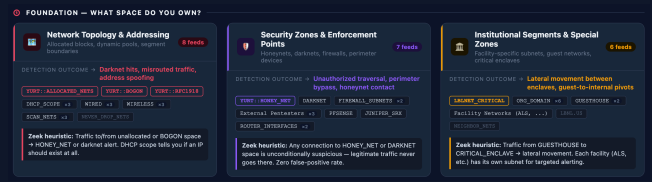
`LBLNET_CRITICAL` `ORG_DOMAIN ×6` `GUESTHOUSE ×2` `Facility Networks (ALS, ...)` `LBNL.US` `NEIGHBOR_NETS`

Zeek heuristic: Traffic from GUESTHOUSE to CRITICAL_ENCLAVE → lateral movement. Each facility (ALS, etc.) has its own subnet for targeted alerting.

Know Your Network

Normalized taxonomy of network intelligence objects — consolidated from 110+ raw feed names

10	70	14	50+	200+
Categories	Unique Feeds (from 110+ raw)	Auto-generated YURT objects	Active Feeds	Notice Types



② IDENTITY — WHO AND WHAT IS ON THE NETWORK?

Asset Inventory & Identity Mapping

Who owns each IP? Which device is it? Is it managed?

7 feeds

DETECTION OUTCOME → **Unmanaged device alerts, user attribution, rogue endpoints**

`YURT::DHCP_HOSTS` `YURT::CrowdStrike`

`LDAP:Contacts ×3` `IDM` `CrowdStrike ×3`

`MAC_VENDOR_OUI` `MACBLOCKS ×3`

Zeek heuristic: IP connecting that has no DHCP lease and no EDR agent → unmanaged device. Cross-reference MAC OUI to classify: laptop, printer, IoT.

DNS & Name Resolution

Authoritative zones, resolvers, stale/external records

6 feeds

DETECTION OUTCOME → **DNS hijacking, shadow IT, rogue resolvers, stale zone data**

`YURT::NAMESERVERS` `DNS=A ×4` `DNS=AAAA`

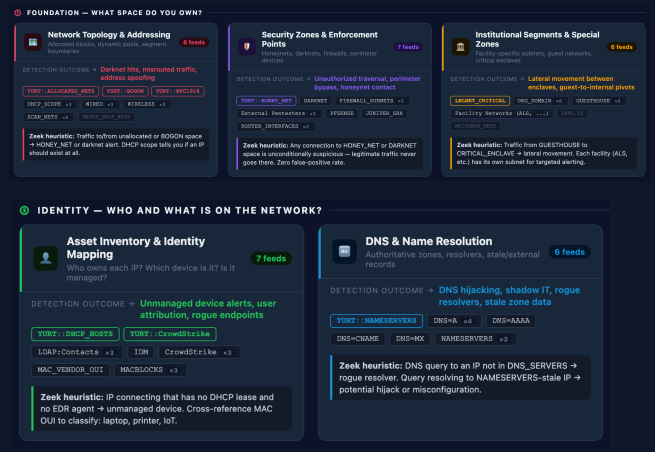
`DNS=CNAME` `DNS=MX` `NAMESERVERS ×3`

Zeek heuristic: DNS query to an IP not in DNS_SERVERS → rogue resolver. Query resolving to NAMESERVERS-stale IP → potential hijack or misconfiguration.

Know Your Network

Normalized taxonomy of network intelligence objects — consolidated from 110+ raw feed names

10	70	14	50+	200+
Categories	Unique Feeds (from 110+ raw)	Auto-generated YURT objects	Active Feeds	Notice Types



③ INFRASTRUCTURE — WHAT SERVICES ARE RUNNING?

Network Services & Infrastructure

VPN, NTP, RADIUS, mail servers, performance monitors

8 feeds

DETECTION OUTCOME → **Service abuse, unauthorized VPN, NTP amplification**

`YURT::LBL_VPN` `AUTHORIZED_EMAIL_SERVERS` `NTP_SERVERS` `DNS_SERVERS` `RADIUS_CLIENTS` `VPN_SERVERS ×3` `PERFSONAR ×2` `DB_LISTEN`

Zeek heuristic: NTP traffic to IPs not in NTP_SERVERS → potential amplification or tunneling. VPN connections not from VPN_SERVERS → split-tunnel abuse.

Device Inventory by Class

Printers, storage, IoT, lab equipment, power systems

8 feeds

DETECTION OUTCOME → **Unexpected outbound from printers/IoT, rogue device behavior**

`PRINTERS ×5` `NAS / Storage ×2` `NETGEAR` `RASPBERRY_PI` `EATON (UPS)` `XILINX (FPGA)` `LABVIEW` `IMMOVABLES ×2`

Zeek heuristic: Printer making outbound HTTP/HTTPS → immediately suspicious. NAS device initiating connections → ransomware lateral movement indicator.

Software & Version Intelligence

OS versions, open ports, legacy systems, web services

7 feeds

DETECTION OUTCOME → **Vulnerable software targeting, legacy OS exploitation**

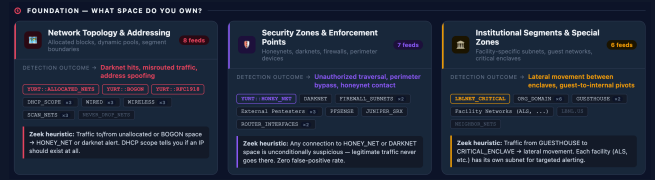
`WINDOWS_legacy ×3` `Apache_Version` `SSH_Version ×3` `RDP_Version` `NoMachine ×2` `OpenWebServers ×4`

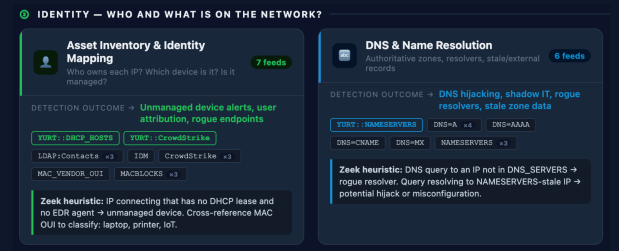
Zeek heuristic: Inbound SMB to WINDOWS_XP or WINDOWS_7 → immediate priority alert. RDP brute-force against known RDP_Version hosts → targeted attack indicator.

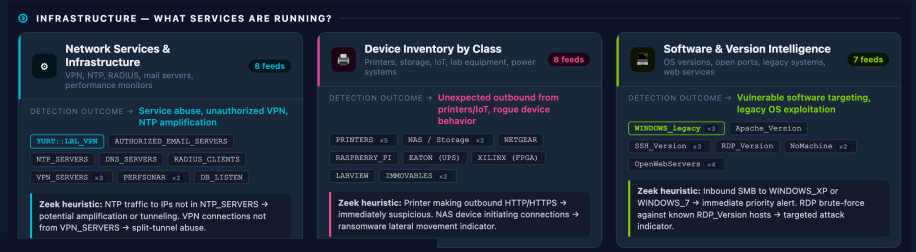
Know Your Network

Normalized taxonomy of network intelligence objects — consolidated from 110+ raw feed names

10	70	14	50+	200+
Categories	Unique Feeds (from 110+ raw)	Auto-generated YURT objects	Active Feeds	Notice Types







④ EXTERNAL — WHAT'S BEYOND YOUR PERIMETER?

Cloud & External Infrastructure

Cloud providers, CDNs, partner institutions, hosting

6 feeds

DETECTION OUTCOME → **Data exfil to cloud, C2 over legitimate infra, shadow IT**

`YURT::AWS` `YURT::CLOUDFLARE` `AWS` `GCP_IP_RANGE` `CLOUDFLARE ×5` `CLOUD_HOSTING ×2`

Zeek heuristic: Large data transfer to AWS/GCP from a workstation → exfil candidate. Traffic to cloud IPs not matching any known SaaS → shadow app or C2.

Authorized Scanners & External Assessors

Approved vulnerability scanners, DHS, bug bounty, pentesting

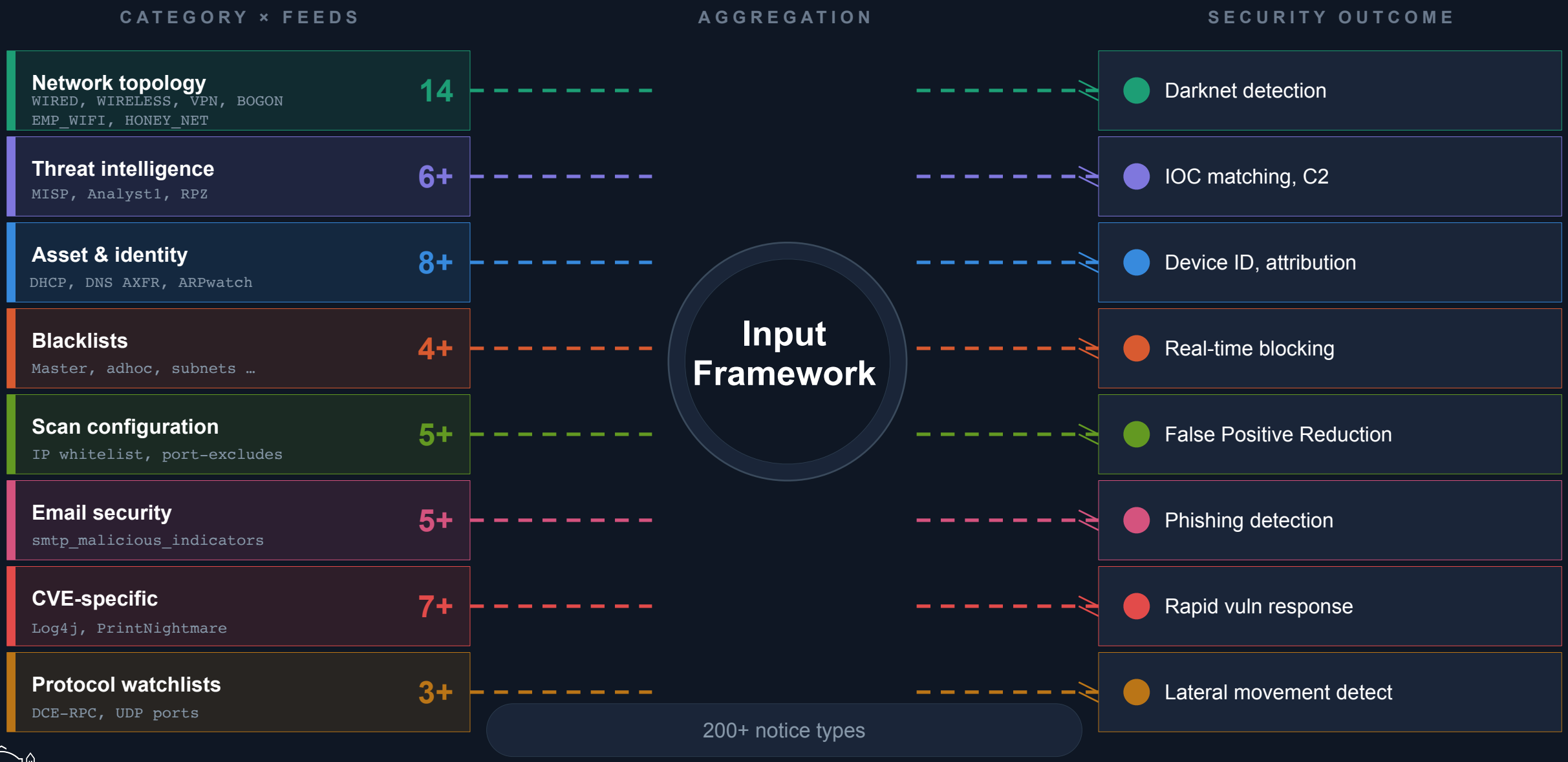
7 feeds

DETECTION OUTCOME → **Suppress false positives from authorized scans; detect unauthorized ones**

`CENSYS` `DHS_SCAN ×2` `QADIUM / Net Systems Research` `SYNACK ×2` `TENABLE` `WIZ` `Internal Scanners`

Zeek heuristic: Scan traffic from IPs in this list → suppress and log only. Identical scan pattern from an IP *not* in this list → high-priority alert.

Feed Inventory — Category to Outcome Map



"Situational awareness — it's a practice."

We started with one text file — our list of allocated subnets!

" Zeek knows our network - its IPs, its subnets, enriching it to know its devices and users."

And it all still runs on plain text files that anyone on the team can edit.

50+	37	200+	14	0
feeds	packages	notices	YURT objects	packet overhead

Start with what you know. Feed it to Zeek. Iterate. … #profit

Questions?

On behalf of LBL Cyber Security Team

Aashish Sharma

Cyber Security • security@lbl.gov

Lawrence Berkeley National Laboratory

These feeds scripts and zeek packages are sharable — let's talk!

50+ Feeds | 37+ Packages | 200+ Notices | 14 YURT Objects

The Input Framework: The tech behind

Zeek's bidirectional data bridge — imports external data into live script state

`Input::add_table()`

External file → queryable Zeek table
Most common pattern. Each row becomes a table entry keyed by Idx record.

`Input::add_event()`

Each row fires a Zeek event
Ideal for streaming data or when you need per-record processing logic.

`Input::add_analysis()`

Forward to File Analysis Framework
Used for binary/file-based inputs that need deeper inspection.

Read Modes

MANUAL

Read once at init
Static config, reference data

REREAD

Watch file, reload on change
Blacklists, whitelists, threat intel

STREAM

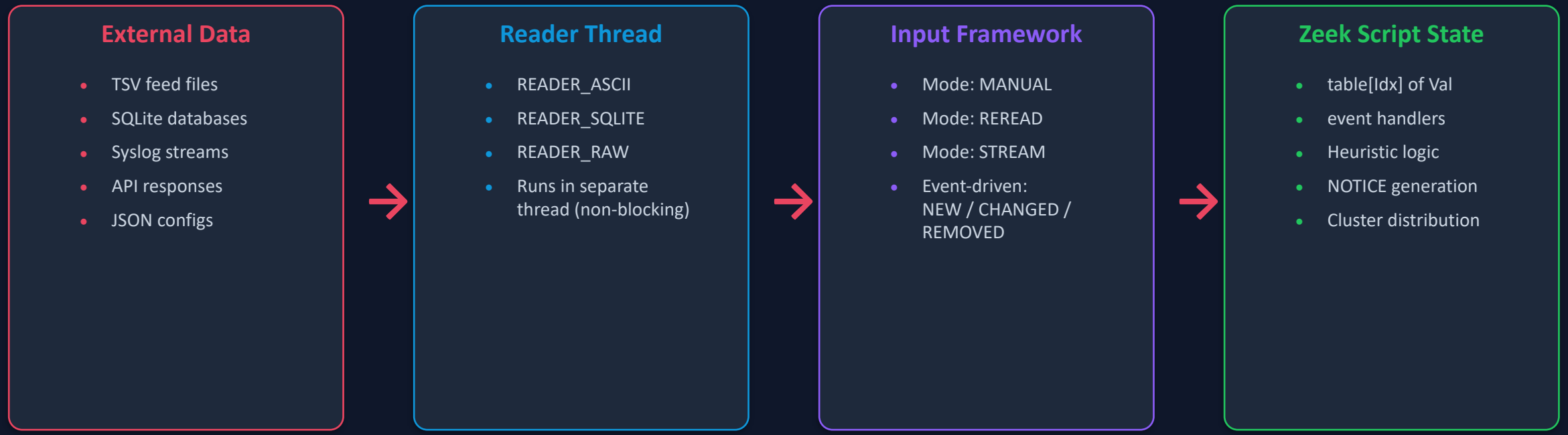
Tail -f continuous streaming
Real-time syslog, API webhooks

Key Design Points

- Reader runs in separate thread — never blocks packet processing
- Data readers: ASCII (TSV), SQLite, Raw, Binary
- REREAD mode: file monitored for changes, auto-reloaded — no Zeek restart needed
- Internally used by Intelligence Framework and Configuration Framework

https://docs.zeek.org/en/master/frameworks/input.html

Input Framework: How It Works




```
type Idx: record { ip: addr; };
type Val: record { reason: string; };
global blocklist: table[addr] of Val;

event zeek_init() {
    Input::add_table([
        $source="/feeds/blocklist.tsv",
        $name="blocklist",
        $idx=Idx, $val=Val,
        $destination=blocklist,
        $mode=Input::REREAD
    ]);
}
```

Pattern: Event-Driven Updates

The $ev parameter lets you react to individual record changes:

```
event read_handler(desc: Input::TableDescription,
                   tpe: Input::Event,
                   left: Idx, right: Val) {
    if (tpe == Input::EVENT_NEW)
        # New record added to feed
    if (tpe == Input::EVENT_REMOVED)
        delete blocklist[left$ip];
}
```

Feed Knowledge into Zeek: Inventory at a Glance

Category	Feeds	Key Examples	Security Outcome
Network Topology	14	YURT::ALLOCATED_NETS, HONEY_NET, BOGON … …	Darknet detection, asset classification
Threat Intelligence	6+	TOR.24hrs, MISP, PhishTank, RPZ, Analyst1 … …	IOC matching, C2 detection
Asset & Identity	8+	Employee DB, DHCP, DNS AXFR, ARP, CrowdStrike … …	Device ID, user attribution
Blacklists	4+	adhoc IPs, subnets, Tippers, repeat offenders … …	Real-time dynamic blocking
Scan Configuration	5+	port-exclude, IP whitelist, subnet whitelist … …	80%+ false positive reduction
Email Security	5+	malicious indicators, phish patterns, FP list … …	Phishing & spam detection
CVE-Specific	7+	Log4j IOCs, PrintNightmare, SIGRed feeds … …	Rapid vulnerability response
Protocol Watchlists	3+	DCE-RPC ops, UDP ports, services … …	Lateral movement detection
TOTAL	50+	Feeding into 37+ packages generating 200+ notice types	