Generating Known Unknowns through Known Knowns

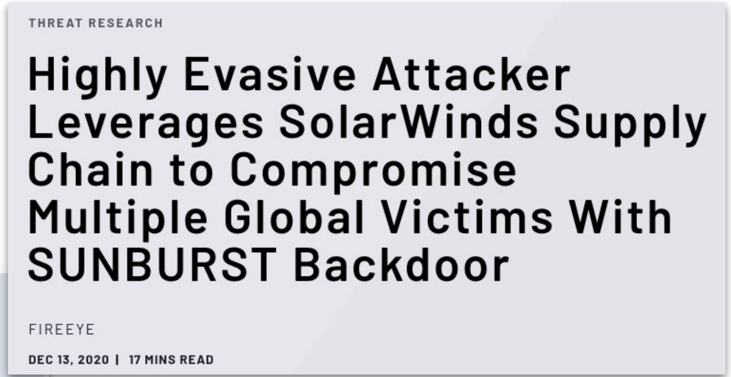
Synthetic Adversarial Log Objects

Hi, I’m Marcus LaFerrera (@mlaferrera)

- Staff Security Strategist @ Splunk
- Previously supported many things DoD, most recently @ DARPA
- Avid open-source contributor, mostly python
- Former lead developer of **stoQ**
- Dreams about **automating** everything

SURGe
by Splunk

first... some background



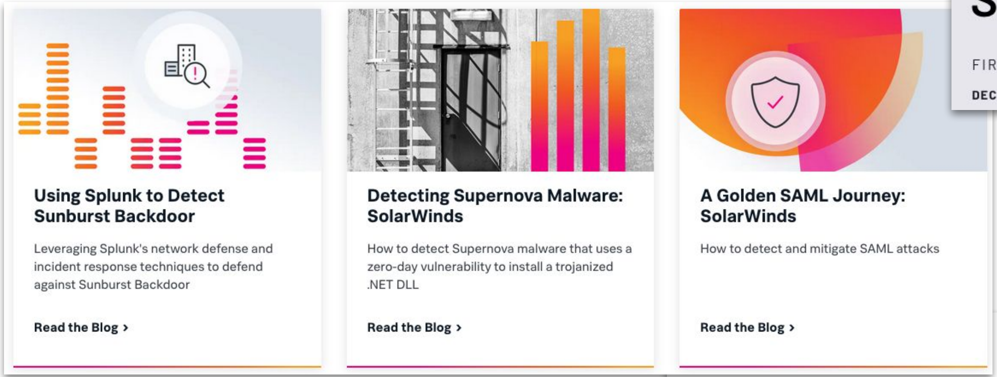
THREAT RESEARCH

Highly Evasive Attacker Leverages SolarWinds Supply Chain to Compromise Multiple Global Victims With SUNBURST Backdoor

FIREEYE

DEC 13, 2020 | 17 MINS READ

*



Using Splunk to Detect Sunburst Backdoor

Leveraging Splunk's network defense and incident response techniques to defend against Sunburst Backdoor

Read the Blog ›

Detecting Supernova Malware: SolarWinds

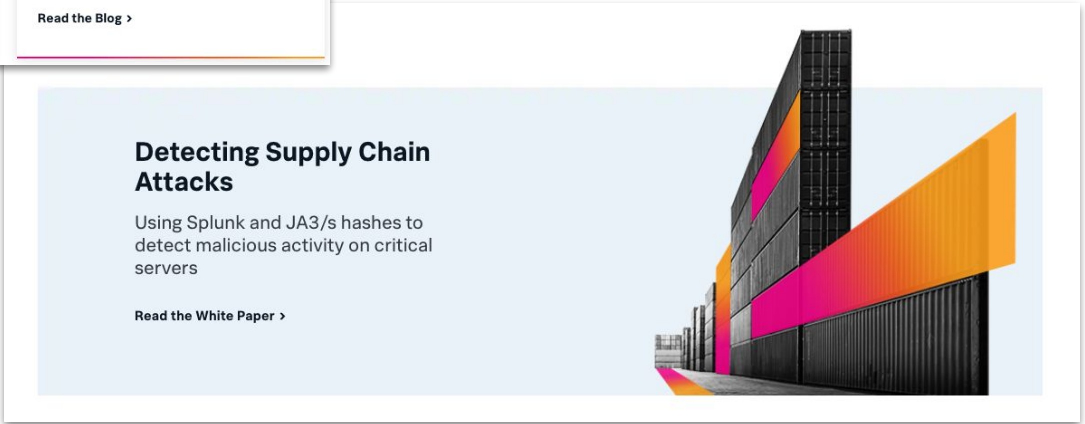
How to detect Supernova malware that uses a zero-day vulnerability to install a trojanized .NET DLL

Read the Blog ›

A Golden SAML Journey: SolarWinds

How to detect and mitigate SAML attacks

Read the Blog ›



Detecting Supply Chain Attacks

Using Splunk and JA3/s hashes to detect malicious activity on critical servers

Read the White Paper ›

* https://www.mandiant.com/resources/evasive-attacker-leverages-solarwinds-supply-chain-compromises-with-sunburst-backdoor

http

ssl

cloud providers

wire data

syslog

pcap

DATA

endpoints

xml

dns

schemas

infrastructure

json

servers

There’s a (better) way

SALO

https://github.com/splunk/salo

Caveat:

This is a work in progress

Synthetic Adversarial Log Objects (SALO) is a framework for the generation of log events without the need for infrastructure or actions to initiate the event that causes a log event. The purpose of this framework is to allow security practitioners, data scientists, and researchers the ability to create log events in a simple, repeatable, and randomized way without the overhead of traditional required resources.

But, why another fake log generator?

Low barrier to entry

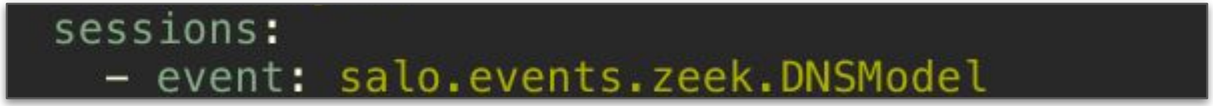
Minimal effort

Repeatable process

Highly customizable

Let’s walk through a simple scenario

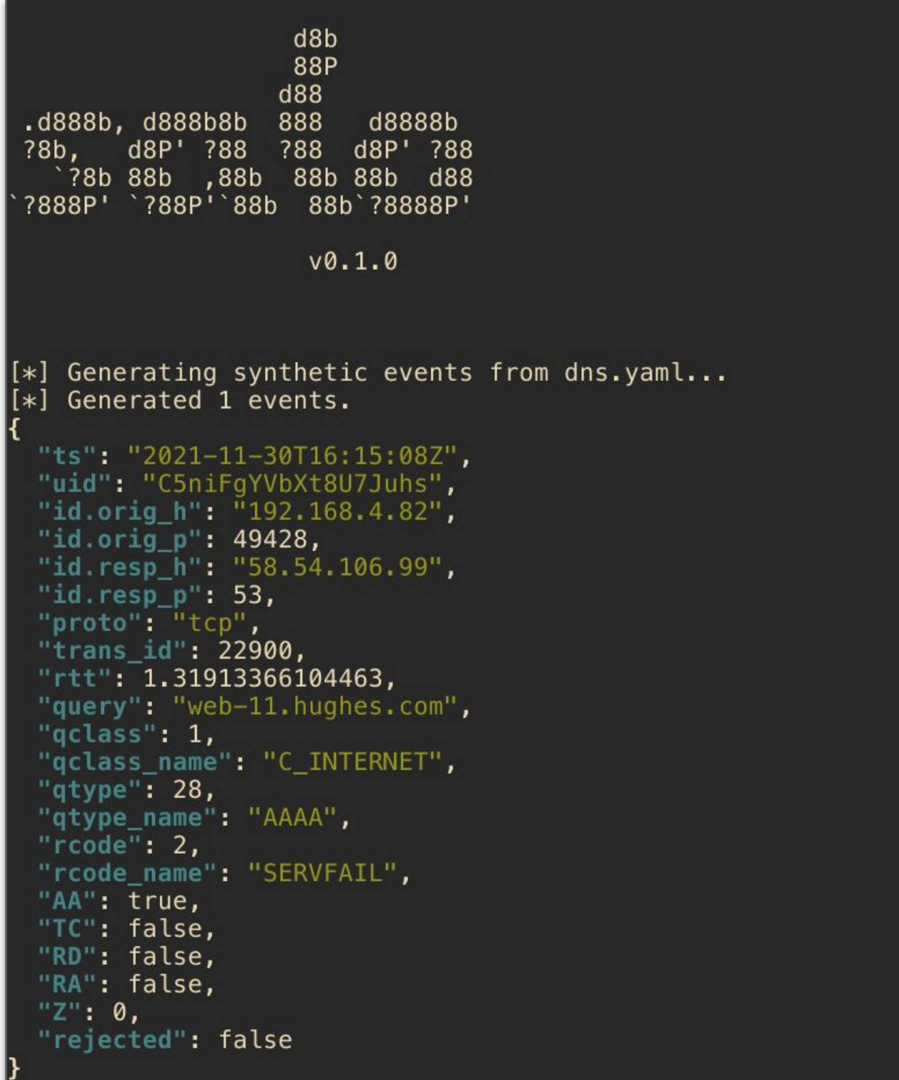
1. Create a new recipe, `dns.yaml`



```
sessions:
  - event: salo.events.zeek.DNSModel
```

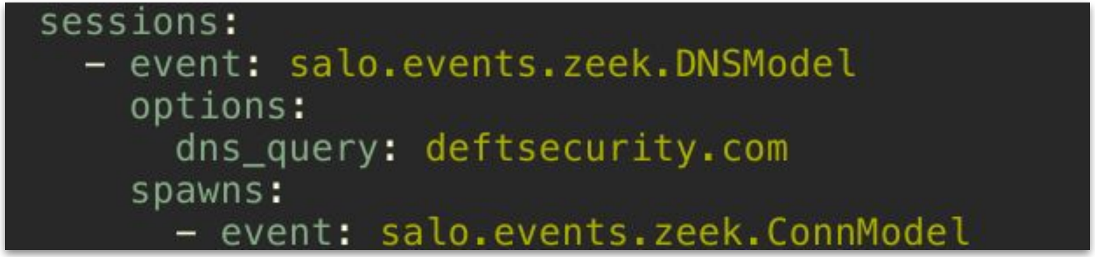
2. Run `salo`, and generate our data

```
$ salo recipe dns.yaml
```



```
                   d8b
                   88P
                  d88
 .d888b, d888b8b  888    d8888b
 ?8b,    d8P' ?88  ?88   d8P' ?88
   `?8b 88b  ,88b   88b 88b  d88
`?888P' `?88P'`88b   88b`?8888P'

                    v0.1.0



[*] Generating synthetic events from dns.yaml...
[*] Generated 1 events.
{
  "ts": "2021-11-30T16:15:08Z",
  "uid": "C5niFgYVbXt8U7Juhs",
  "id.orig_h": "192.168.4.82",
  "id.orig_p": 49428,
  "id.resp_h": "58.54.106.99",
  "id.resp_p": 53,
  "proto": "tcp",
  "trans_id": 22900,
  "rtt": 1.31913366104463,
  "query": "web-11.hughes.com",
  "qclass": 1,
  "qclass_name": "C_INTERNET",
  "qtype": 28,
  "qtype_name": "AAAA",
  "rcode": 2,
  "rcode_name": "SERVFAIL",
  "AA": true,
  "TC": false,
  "RD": false,
  "RA": false,
  "Z": 0,
  "rejected": false
}
```

3. Define a domain to query
4. Spawn a Zeek conn event



```
sessions:
  - event: salo.events.zeek.DNSModel
    options:
      dns_query: deftsecurity.com
    spawns:
      - event: salo.events.zeek.ConnModel
```

5. Run `salo` again

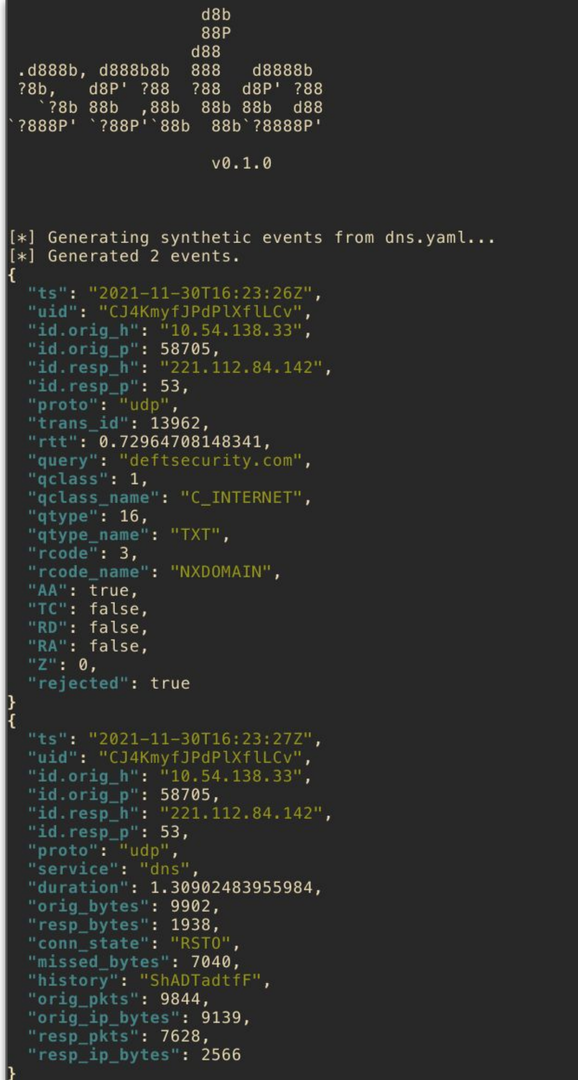
```
$ salo recipe dns.yaml
```

or...

```
$ salo recipe -o output.yaml --splunk dns.yaml
```

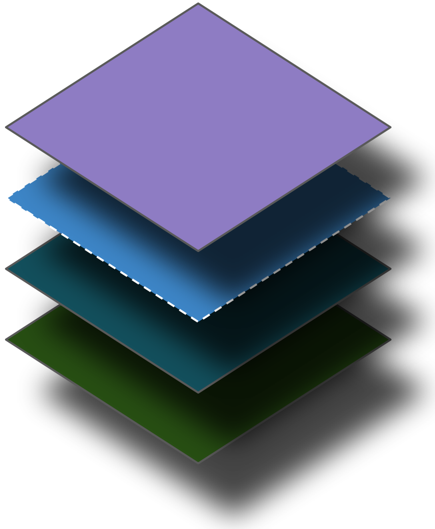
or...

```
$ salo recipe -o output.yaml --file dns.yaml
```

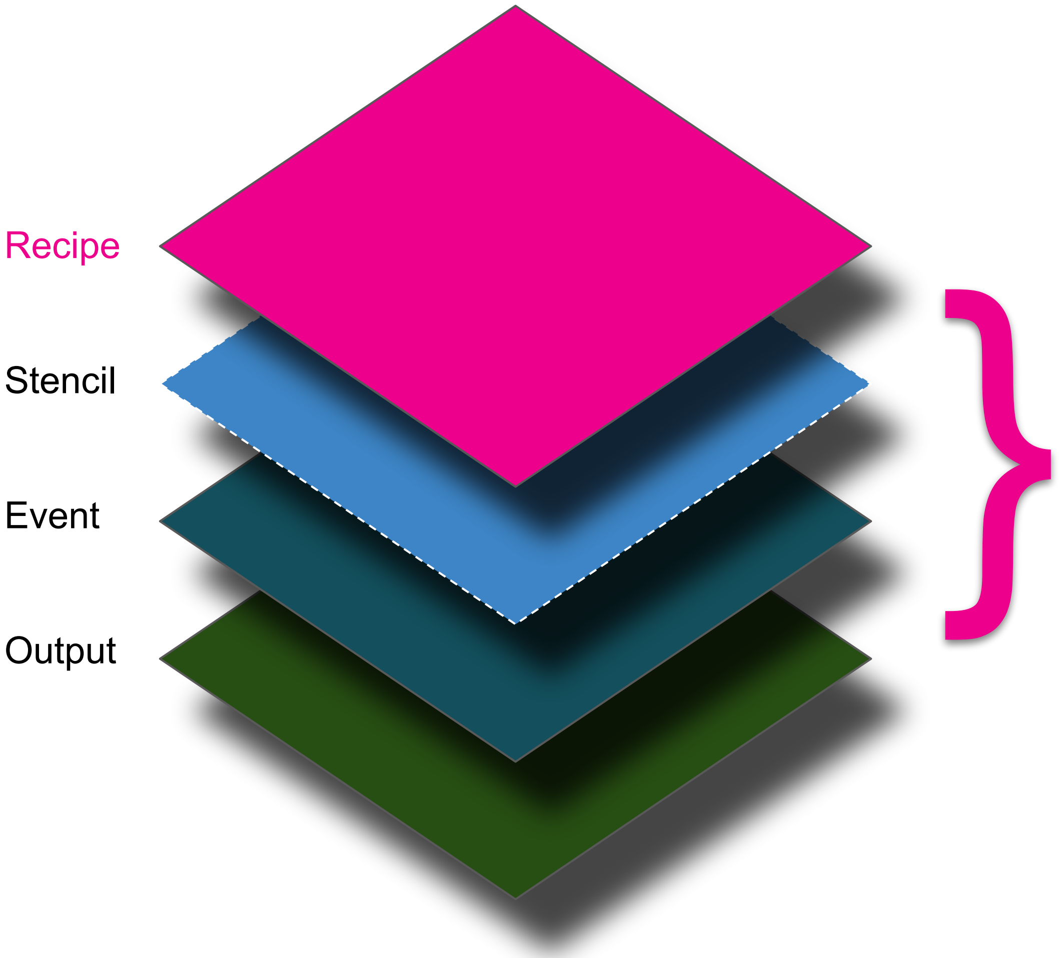


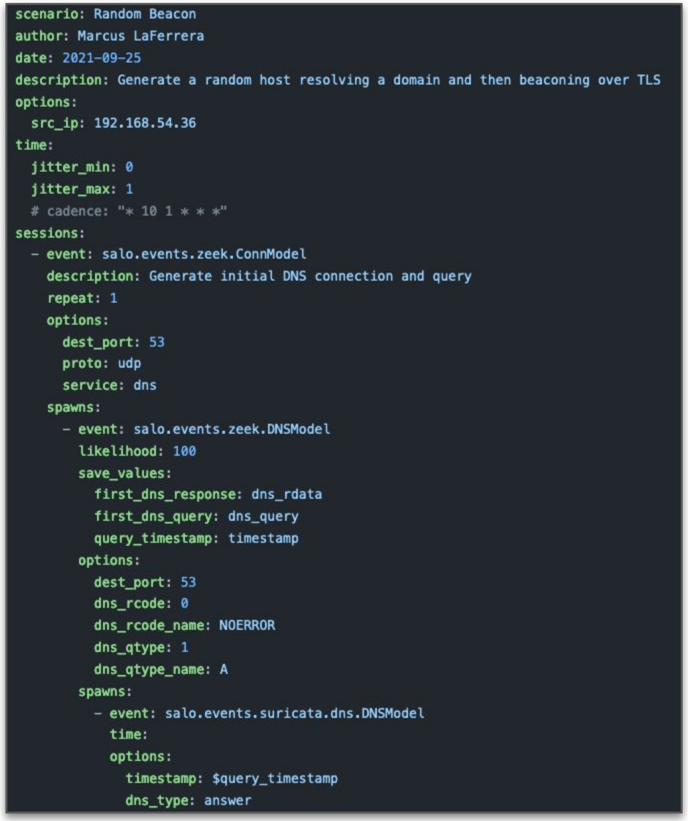
```
                   d8b
                   88P
                  d88
 .d888b, d888b8b  888    d8888b
 ?8b,    d8P' ?88  ?88  d8P' ?88
   `?8b 88b  ,88b  88b 88b  d88
`?888P' `?88P'`88b  88b`?8888P'

                    v0.1.0


[*] Generating synthetic events from dns.yaml...
[*] Generated 2 events.
{
  "ts": "2021-11-30T16:23:26Z",
  "uid": "CJ4KmyfJPdPlXflLCv",
  "id.orig_h": "10.54.138.33",
  "id.orig_p": 58705,
  "id.resp_h": "221.112.84.142",
  "id.resp_p": 53,
  "proto": "udp",
  "trans_id": 13962,
  "rtt": 0.72964708148341,
  "query": "deftsecurity.com",
  "qclass": 1,
  "qclass_name": "C_INTERNET",
  "qtype": 16,
  "qtype_name": "TXT",
  "rcode": 3,
  "rcode_name": "NXDOMAIN",
  "AA": true,
  "TC": false,
  "RD": false,
  "RA": false,
  "Z": 0,
  "rejected": true
}
{
  "ts": "2021-11-30T16:23:27Z",
  "uid": "CJ4KmyfJPdPlXflLCv",
  "id.orig_h": "10.54.138.33",
  "id.orig_p": 58705,
  "id.resp_h": "221.112.84.142",
  "id.resp_p": 53,
  "proto": "udp",
  "service": "dns",
  "duration": 1.30902483955984,
  "orig_bytes": 9902,
  "resp_bytes": 1938,
  "conn_state": "RSTO",
  "missed_bytes": 7040,
  "history": "ShADTadtfF",
  "orig_pkts": 9844,
  "orig_ip_bytes": 9139,
  "resp_pkts": 7628,
  "resp_ip_bytes": 2566
}
```

Recipes | Stencils | Events | Outputs

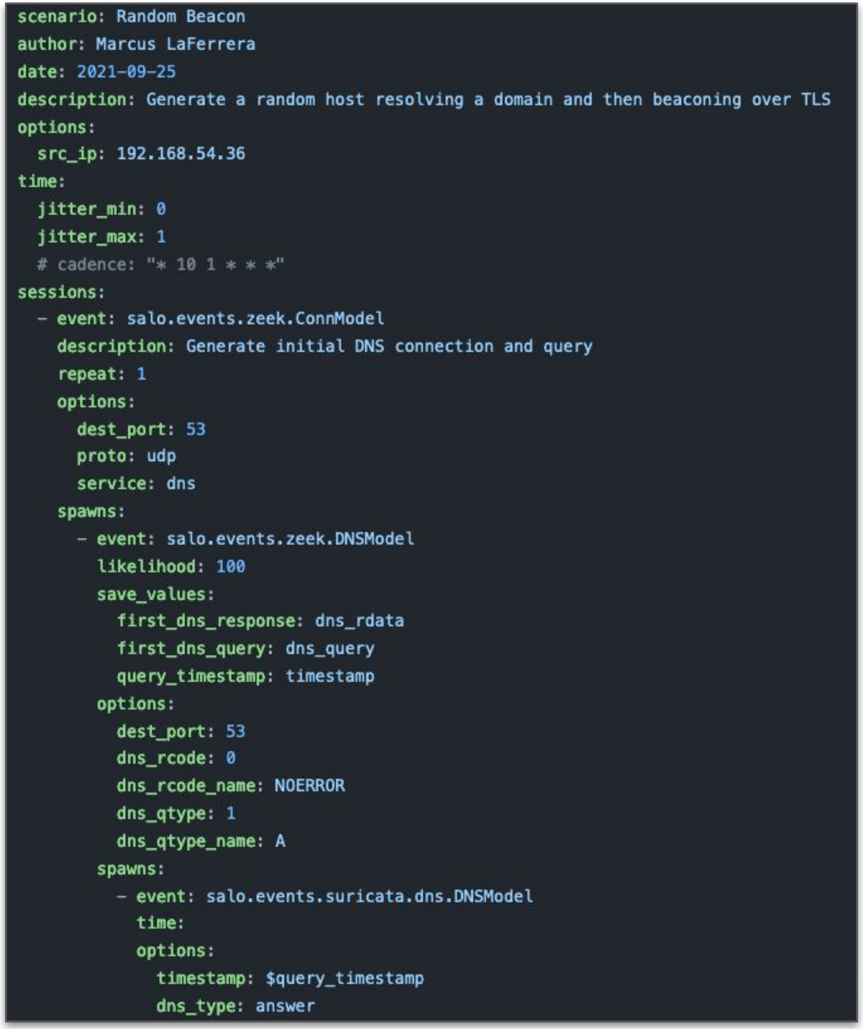


define a scenario

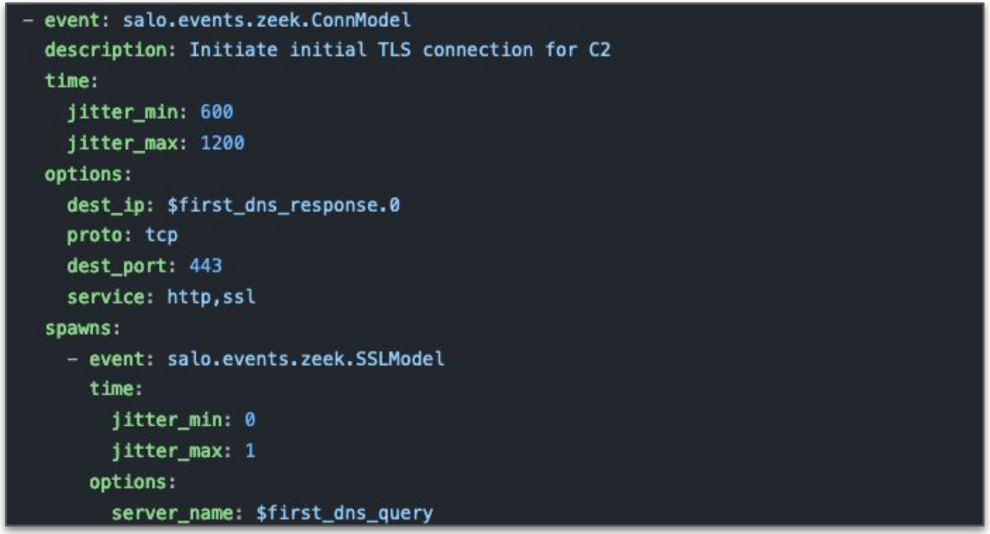




```yaml
scenario: Random Beacon
author: Marcus LaFerrera
date: 2021-09-25
description: Generate a random host resolving a domain and then beaconing over TLS
options:
  src_ip: 192.168.54.36
time:
  jitter_min: 0
  jitter_max: 1
  # cadence: "* 10 1 * * *"
sessions:
  - event: salo.events.zeek.ConnModel
    description: Generate initial DNS connection and query
    repeat: 1
    options:
      dest_port: 53
      proto: udp
      service: dns
    spawns:
      - event: salo.events.zeek.DNSModel
        likelihood: 100
        save_values:
          first_dns_response: dns_rdata
          first_dns_query: dns_query
          query_timestamp: timestamp
        options:
          dest_port: 53
          dns_rcode: 0
          dns_rcode_name: NOERROR
          dns_qtype: 1
          dns_qtype_name: A
        spawns:
          - event: salo.events.suricata.dns.DNSModel
            time:
            options:
              timestamp: $query_timestamp
              dns_type: answer
```

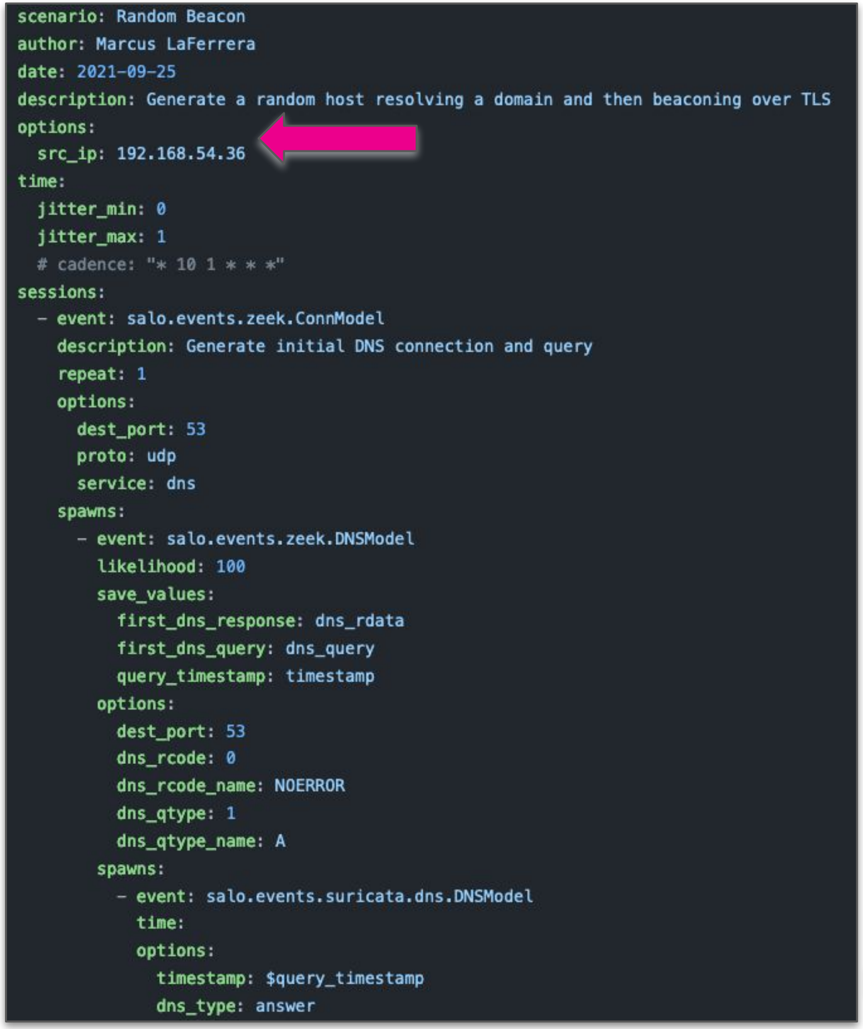


```yaml
scenario: Random Beacon
author: Marcus LaFerrera
date: 2021-09-25
description: Generate a random host resolving a domain and then beaconing over TLS
options:
  src_ip: 192.168.54.36
time:
  jitter_min: 0
  jitter_max: 1
  # cadence: "* 10 1 * * *"
sessions:
  - event: salo.events.zeek.ConnModel
    description: Generate initial DNS connection and query
    repeat: 1
    options:
      dest_port: 53
      proto: udp
      service: dns
    spawns:
      - event: salo.events.zeek.DNSModel
        likelihood: 100
        save_values:
          first_dns_response: dns_rdata
          first_dns_query: dns_query
          query_timestamp: timestamp
        options:
          dest_port: 53
          dns_rcode: 0
          dns_rcode_name: NOERROR
          dns_qtype: 1
          dns_qtype_name: A
        spawns:
          - event: salo.events.suricata.dns.DNSModel
            time:
            options:
              timestamp: $query_timestamp
              dns_type: answer
```

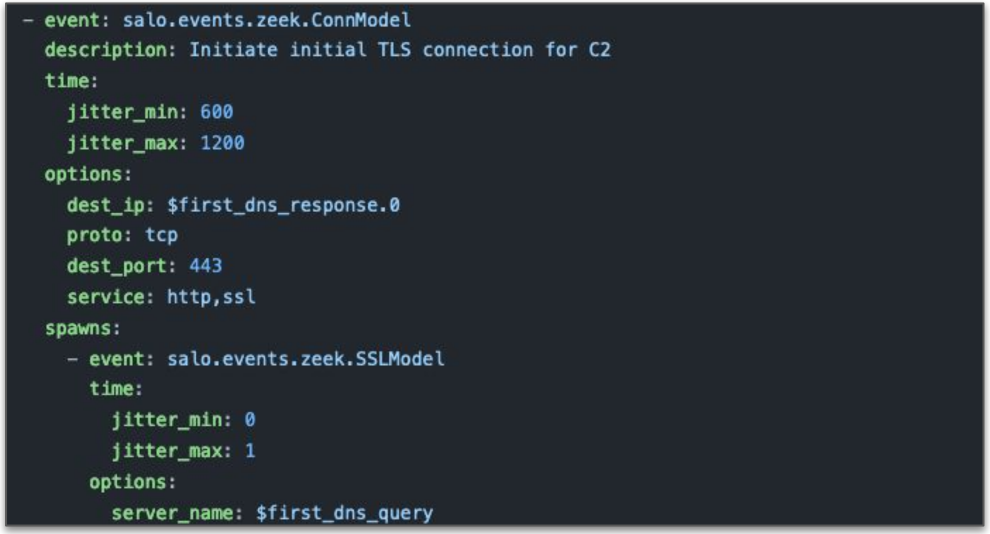


```yaml
- event: salo.events.zeek.ConnModel
  description: Initiate initial TLS connection for C2
  time:
    jitter_min: 600
    jitter_max: 1200
  options:
    dest_ip: $first_dns_response.0
    proto: tcp
    dest_port: 443
    service: http,ssl
  spawns:
    - event: salo.events.zeek.SSLModel
      time:
        jitter_min: 0
        jitter_max: 1
      options:
        server_name: $first_dns_query
```

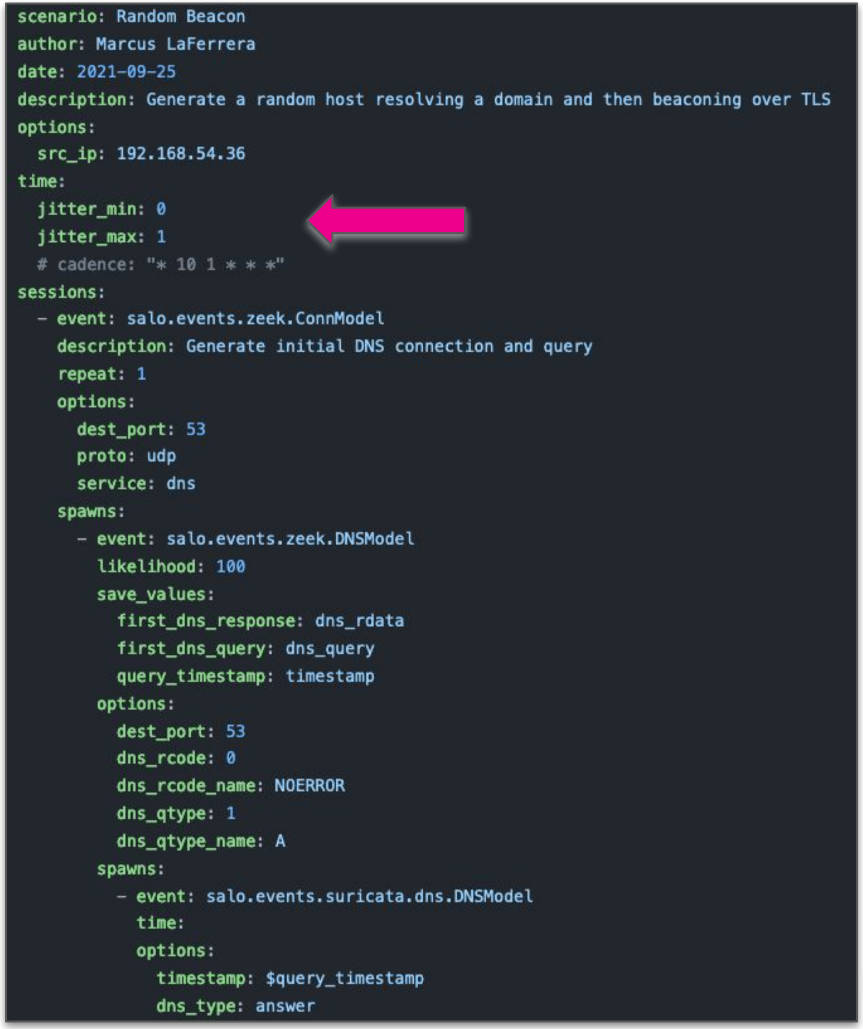
Global Options



```yaml
scenario: Random Beacon
author: Marcus LaFerrera
date: 2021-09-25
description: Generate a random host resolving a domain and then beaconing over TLS
options:
  src_ip: 192.168.54.36
time:
  jitter_min: 0
  jitter_max: 1
  # cadence: "* 10 1 * * *"
sessions:
  - event: salo.events.zeek.ConnModel
    description: Generate initial DNS connection and query
    repeat: 1
    options:
      dest_port: 53
      proto: udp
      service: dns
    spawns:
      - event: salo.events.zeek.DNSModel
        likelihood: 100
        save_values:
          first_dns_response: dns_rdata
          first_dns_query: dns_query
          query_timestamp: timestamp
        options:
          dest_port: 53
          dns_rcode: 0
          dns_rcode_name: NOERROR
          dns_qtype: 1
          dns_qtype_name: A
        spawns:
          - event: salo.events.suricata.dns.DNSModel
            time:
            options:
              timestamp: $query_timestamp
              dns_type: answer
```

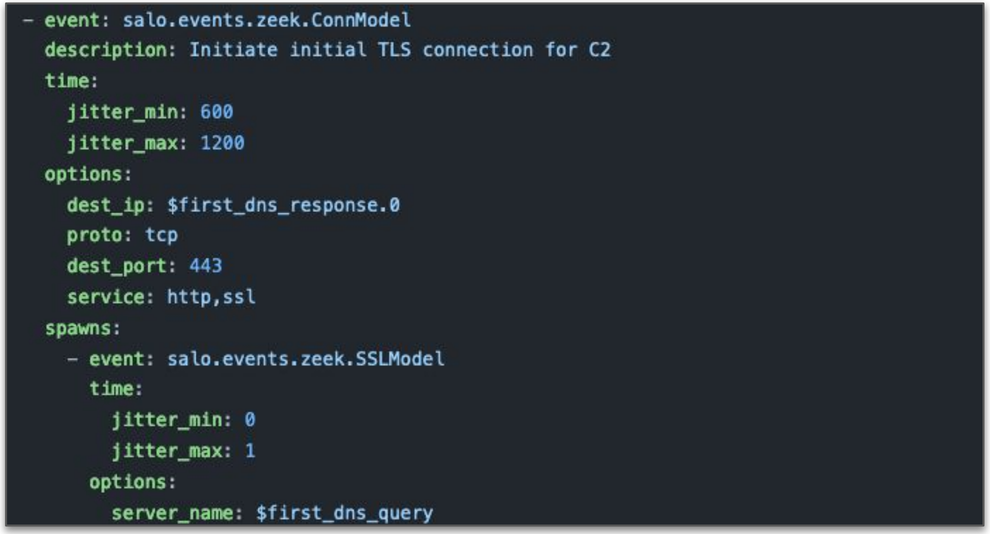


```yaml
- event: salo.events.zeek.ConnModel
  description: Initiate initial TLS connection for C2
  time:
    jitter_min: 600
    jitter_max: 1200
  options:
    dest_ip: $first_dns_response.0
    proto: tcp
    dest_port: 443
    service: http,ssl
  spawns:
    - event: salo.events.zeek.SSLModel
      time:
        jitter_min: 0
        jitter_max: 1
      options:
        server_name: $first_dns_query
```



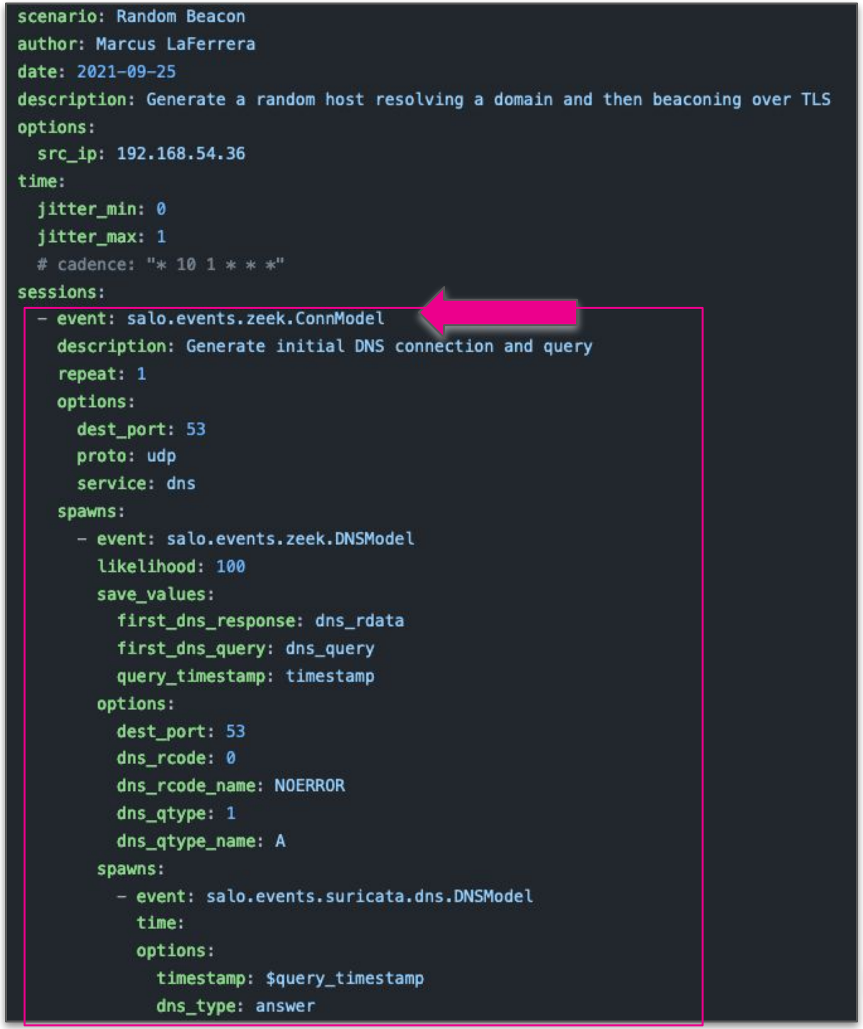
```
scenario: Random Beacon
author: Marcus LaFerrera
date: 2021-09-25
description: Generate a random host resolving a domain and then beaconing over TLS
options:
  src_ip: 192.168.54.36
time:
  jitter_min: 0
  jitter_max: 1
  # cadence: "* 10 1 * * *"
sessions:
  - event: salo.events.zeek.ConnModel
    description: Generate initial DNS connection and query
    repeat: 1
    options:
      dest_port: 53
      proto: udp
      service: dns
    spawns:
      - event: salo.events.zeek.DNSModel
        likelihood: 100
        save_values:
          first_dns_response: dns_rdata
          first_dns_query: dns_query
          query_timestamp: timestamp
        options:
          dest_port: 53
          dns_rcode: 0
          dns_rcode_name: NOERROR
          dns_qtype: 1
          dns_qtype_name: A
        spawns:
          - event: salo.events.suricata.dns.DNSModel
            time:
            options:
              timestamp: $query_timestamp
              dns_type: answer
```

Timestamps, Jitter, and Cadence

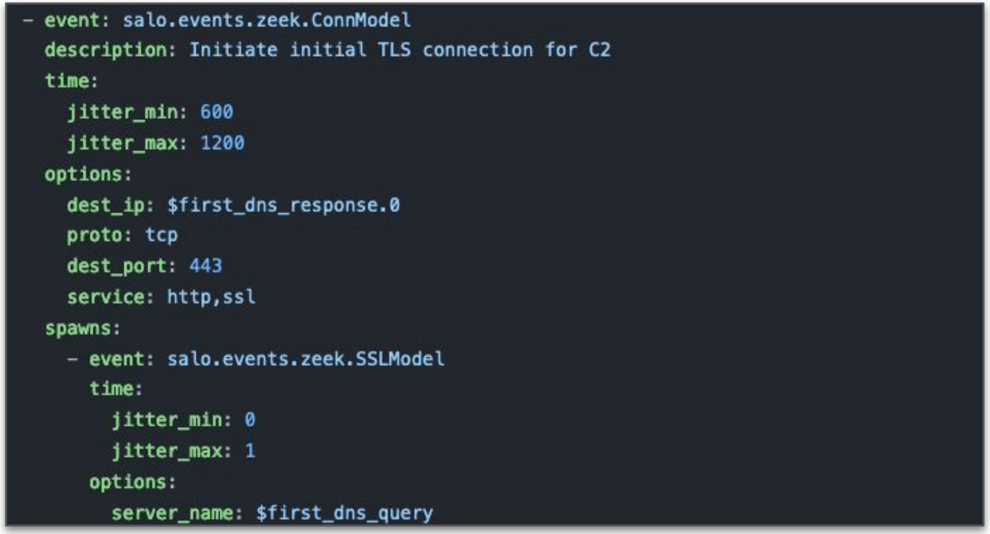


```
- event: salo.events.zeek.ConnModel
  description: Initiate initial TLS connection for C2
  time:
    jitter_min: 600
    jitter_max: 1200
  options:
    dest_ip: $first_dns_response.0
    proto: tcp
    dest_port: 443
    service: http,ssl
  spawns:
    - event: salo.events.zeek.SSLModel
      time:
        jitter_min: 0
        jitter_max: 1
      options:
        server_name: $first_dns_query
```

Sessions and Events

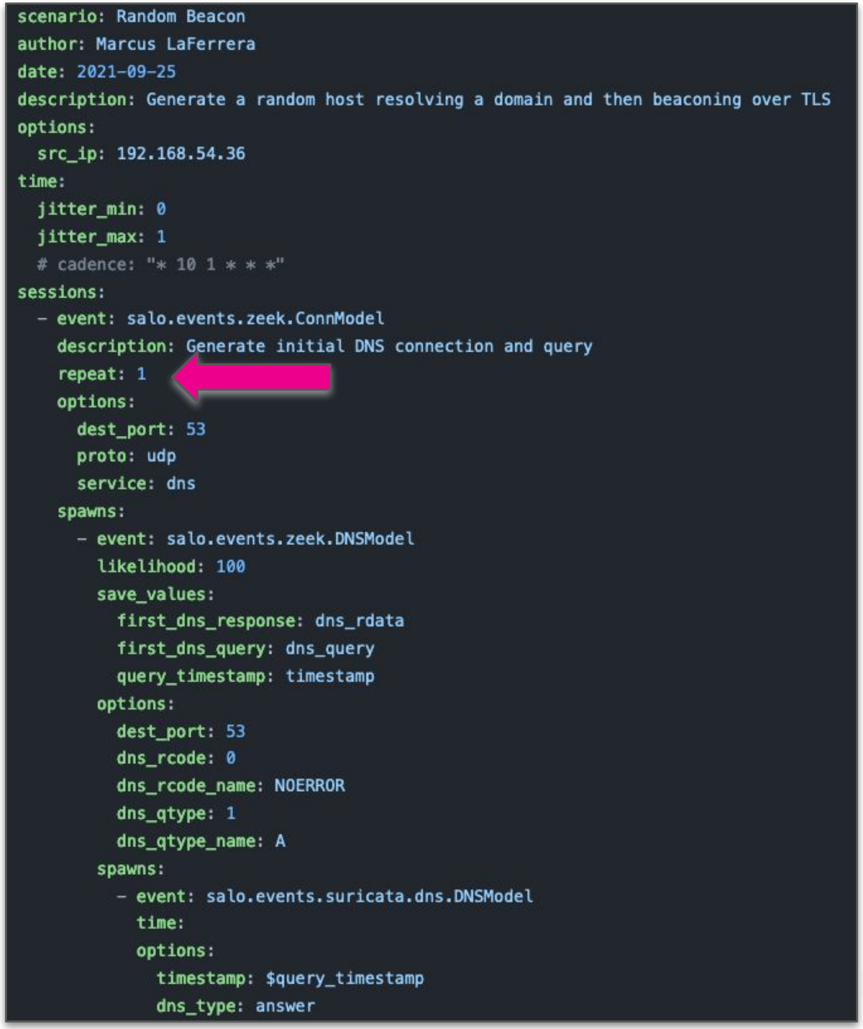


```
scenario: Random Beacon
author: Marcus LaFerrera
date: 2021-09-25
description: Generate a random host resolving a domain and then beaconing over TLS
options:
  src_ip: 192.168.54.36
time:
  jitter_min: 0
  jitter_max: 1
  # cadence: "* 10 1 * * *"
sessions:
  - event: salo.events.zeek.ConnModel
    description: Generate initial DNS connection and query
    repeat: 1
    options:
      dest_port: 53
      proto: udp
      service: dns
    spawns:
      - event: salo.events.zeek.DNSModel
        likelihood: 100
        save_values:
          first_dns_response: dns_rdata
          first_dns_query: dns_query
          query_timestamp: timestamp
        options:
          dest_port: 53
          dns_rcode: 0
          dns_rcode_name: NOERROR
          dns_qtype: 1
          dns_qtype_name: A
        spawns:
          - event: salo.events.suricata.dns.DNSModel
            time:
            options:
              timestamp: $query_timestamp
              dns_type: answer
```

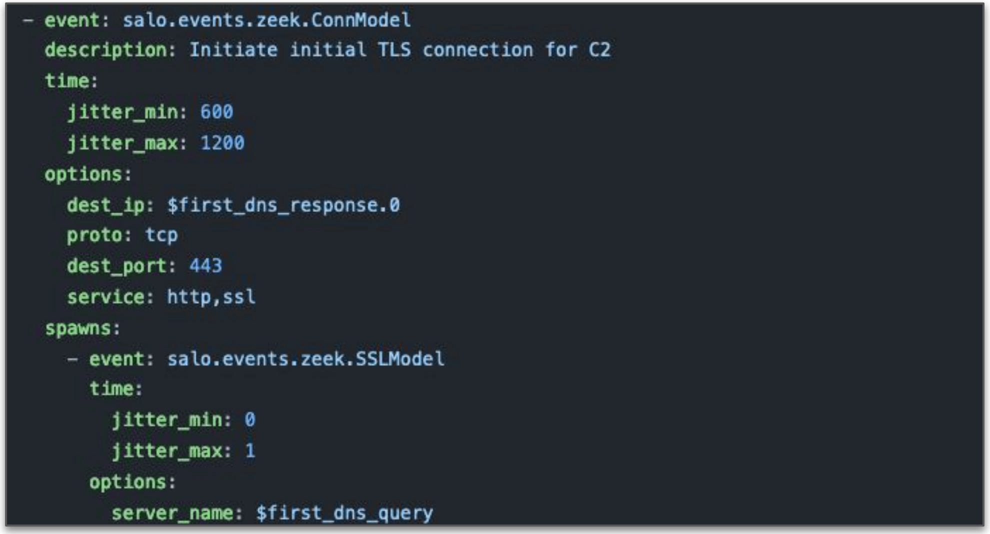


```
- event: salo.events.zeek.ConnModel
  description: Initiate initial TLS connection for C2
  time:
    jitter_min: 600
    jitter_max: 1200
  options:
    dest_ip: $first_dns_response.0
    proto: tcp
    dest_port: 443
    service: http,ssl
  spawns:
    - event: salo.events.zeek.SSLModel
      time:
        jitter_min: 0
        jitter_max: 1
      options:
        server_name: $first_dns_query
```

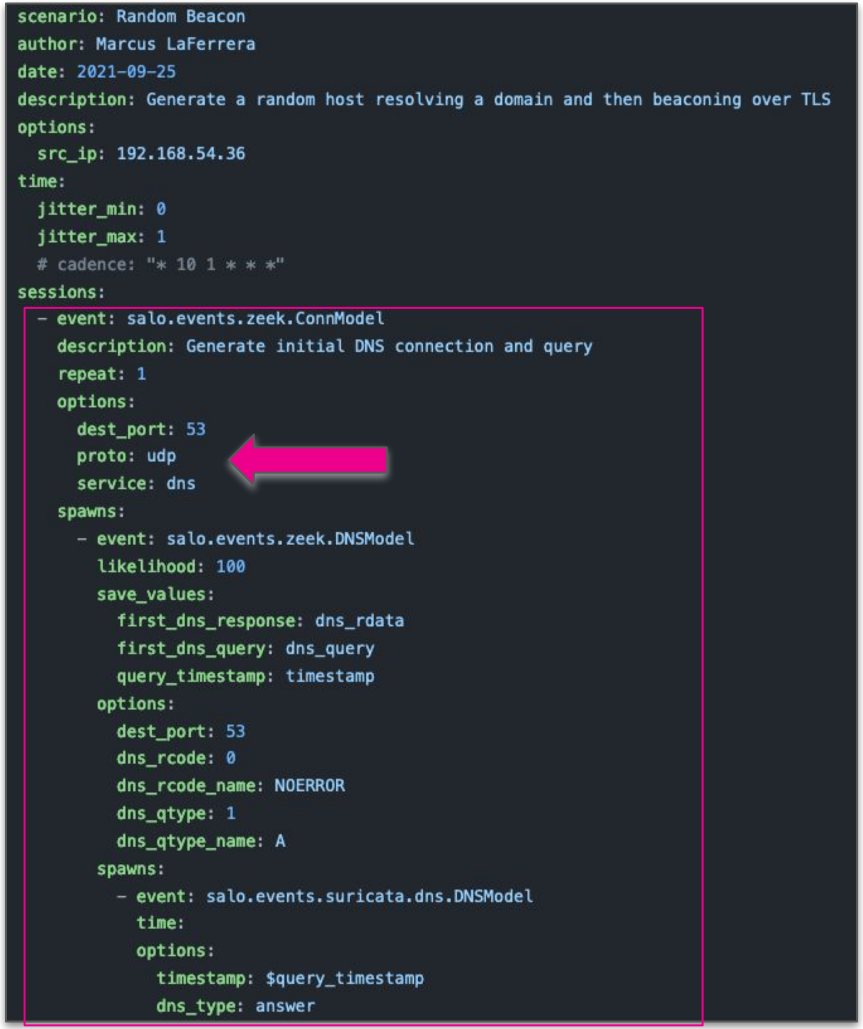
Repeat Events



```
scenario: Random Beacon
author: Marcus LaFerrera
date: 2021-09-25
description: Generate a random host resolving a domain and then beaconing over TLS
options:
  src_ip: 192.168.54.36
time:
  jitter_min: 0
  jitter_max: 1
  # cadence: "* 10 1 * * *"
sessions:
  - event: salo.events.zeek.ConnModel
    description: Generate initial DNS connection and query
    repeat: 1
    options:
      dest_port: 53
      proto: udp
      service: dns
    spawns:
      - event: salo.events.zeek.DNSModel
        likelihood: 100
        save_values:
          first_dns_response: dns_rdata
          first_dns_query: dns_query
          query_timestamp: timestamp
        options:
          dest_port: 53
          dns_rcode: 0
          dns_rcode_name: NOERROR
          dns_qtype: 1
          dns_qtype_name: A
        spawns:
          - event: salo.events.suricata.dns.DNSModel
            time:
            options:
              timestamp: $query_timestamp
              dns_type: answer
```

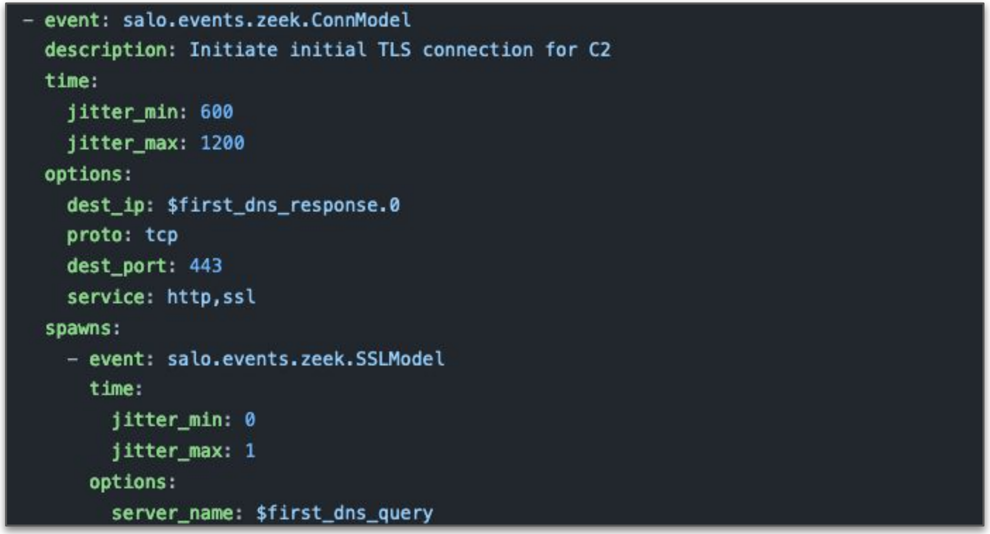


```
- event: salo.events.zeek.ConnModel
  description: Initiate initial TLS connection for C2
  time:
    jitter_min: 600
    jitter_max: 1200
  options:
    dest_ip: $first_dns_response.0
    proto: tcp
    dest_port: 443
    service: http,ssl
  spawns:
    - event: salo.events.zeek.SSLModel
      time:
        jitter_min: 0
        jitter_max: 1
      options:
        server_name: $first_dns_query
```



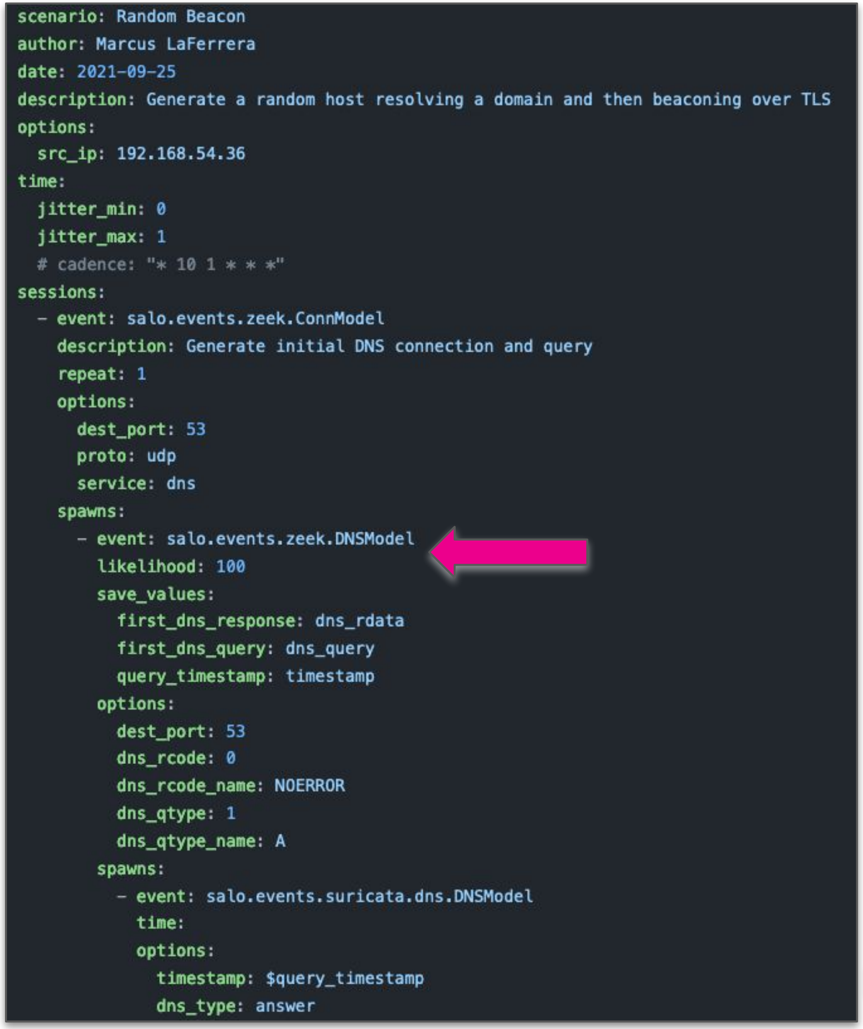
```
scenario: Random Beacon
author: Marcus LaFerrera
date: 2021-09-25
description: Generate a random host resolving a domain and then beaconing over TLS
options:
  src_ip: 192.168.54.36
time:
  jitter_min: 0
  jitter_max: 1
  # cadence: "* 10 1 * * *"
sessions:
  - event: salo.events.zeek.ConnModel
    description: Generate initial DNS connection and query
    repeat: 1
    options:
      dest_port: 53
      proto: udp
      service: dns
    spawns:
      - event: salo.events.zeek.DNSModel
        likelihood: 100
        save_values:
          first_dns_response: dns_rdata
          first_dns_query: dns_query
          query_timestamp: timestamp
        options:
          dest_port: 53
          dns_rcode: 0
          dns_rcode_name: NOERROR
          dns_qtype: 1
          dns_qtype_name: A
        spawns:
          - event: salo.events.suricata.dns.DNSModel
            time:
            options:
              timestamp: $query_timestamp
              dns_type: answer
```

Event Options

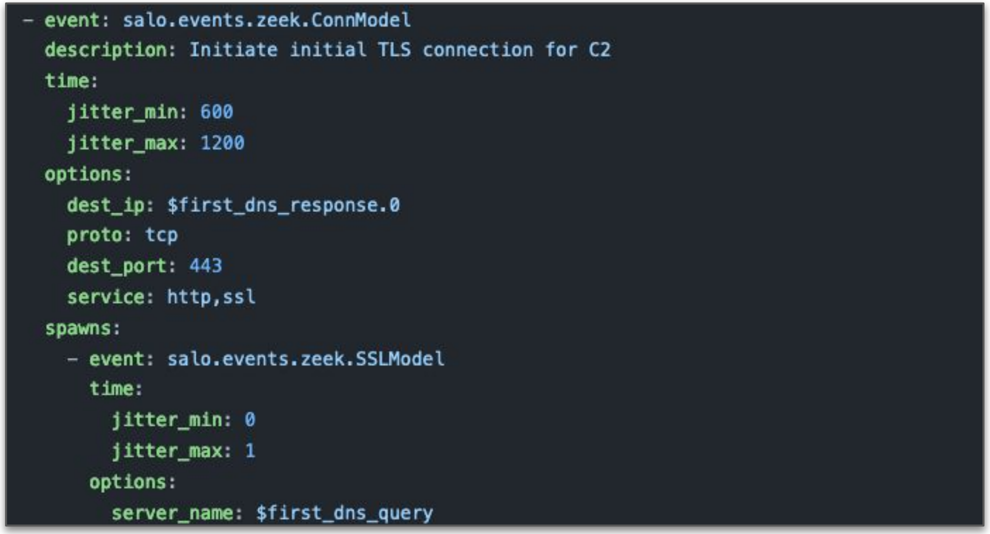


```
- event: salo.events.zeek.ConnModel
  description: Initiate initial TLS connection for C2
  time:
    jitter_min: 600
    jitter_max: 1200
  options:
    dest_ip: $first_dns_response.0
    proto: tcp
    dest_port: 443
    service: http,ssl
  spawns:
    - event: salo.events.zeek.SSLModel
      time:
        jitter_min: 0
        jitter_max: 1
      options:
        server_name: $first_dns_query
```

Spawned Events

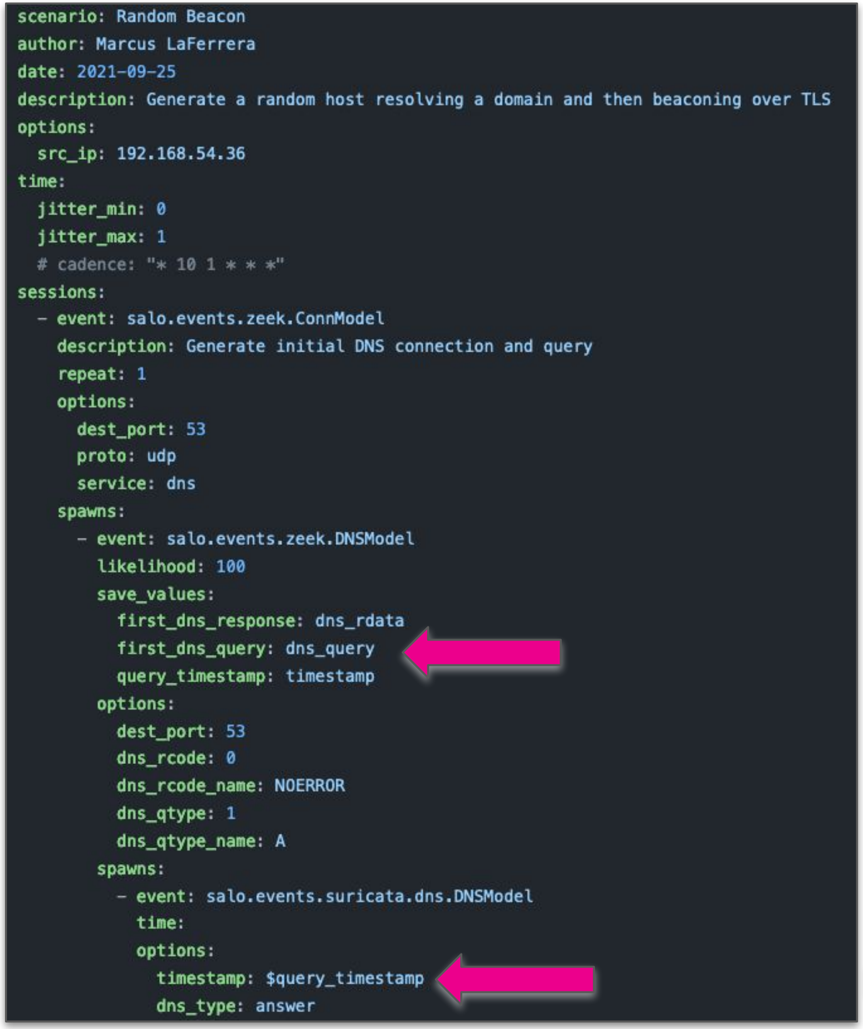


```
scenario: Random Beacon
author: Marcus LaFerrera
date: 2021-09-25
description: Generate a random host resolving a domain and then beaconing over TLS
options:
  src_ip: 192.168.54.36
time:
  jitter_min: 0
  jitter_max: 1
  # cadence: "* 10 1 * * *"
sessions:
  - event: salo.events.zeek.ConnModel
    description: Generate initial DNS connection and query
    repeat: 1
    options:
      dest_port: 53
      proto: udp
      service: dns
    spawns:
      - event: salo.events.zeek.DNSModel
        likelihood: 100
        save_values:
          first_dns_response: dns_rdata
          first_dns_query: dns_query
          query_timestamp: timestamp
        options:
          dest_port: 53
          dns_rcode: 0
          dns_rcode_name: NOERROR
          dns_qtype: 1
          dns_qtype_name: A
        spawns:
          - event: salo.events.suricata.dns.DNSModel
            time:
            options:
              timestamp: $query_timestamp
              dns_type: answer
```

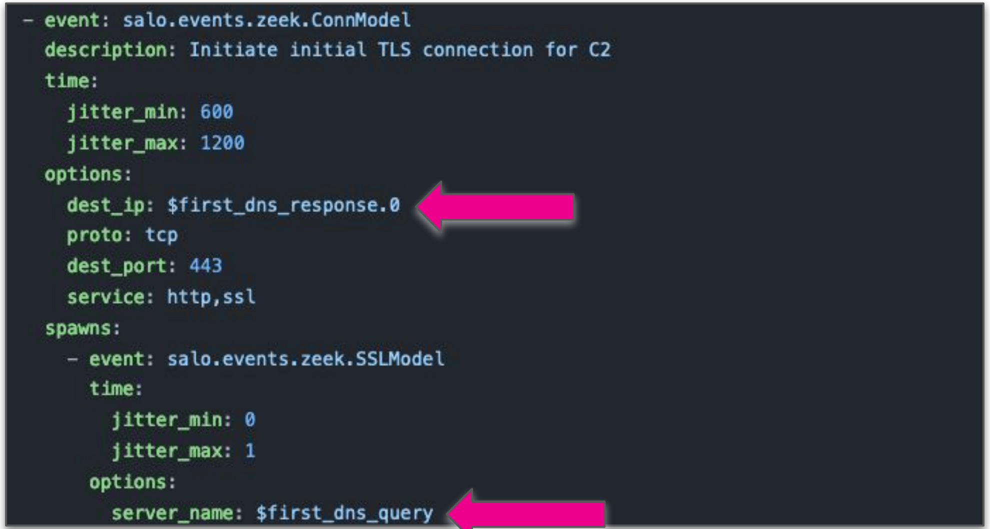


```
- event: salo.events.zeek.ConnModel
  description: Initiate initial TLS connection for C2
  time:
    jitter_min: 600
    jitter_max: 1200
  options:
    dest_ip: $first_dns_response.0
    proto: tcp
    dest_port: 443
    service: http,ssl
  spawns:
    - event: salo.events.zeek.SSLModel
      time:
        jitter_min: 0
        jitter_max: 1
      options:
        server_name: $first_dns_query
```

Saved Values

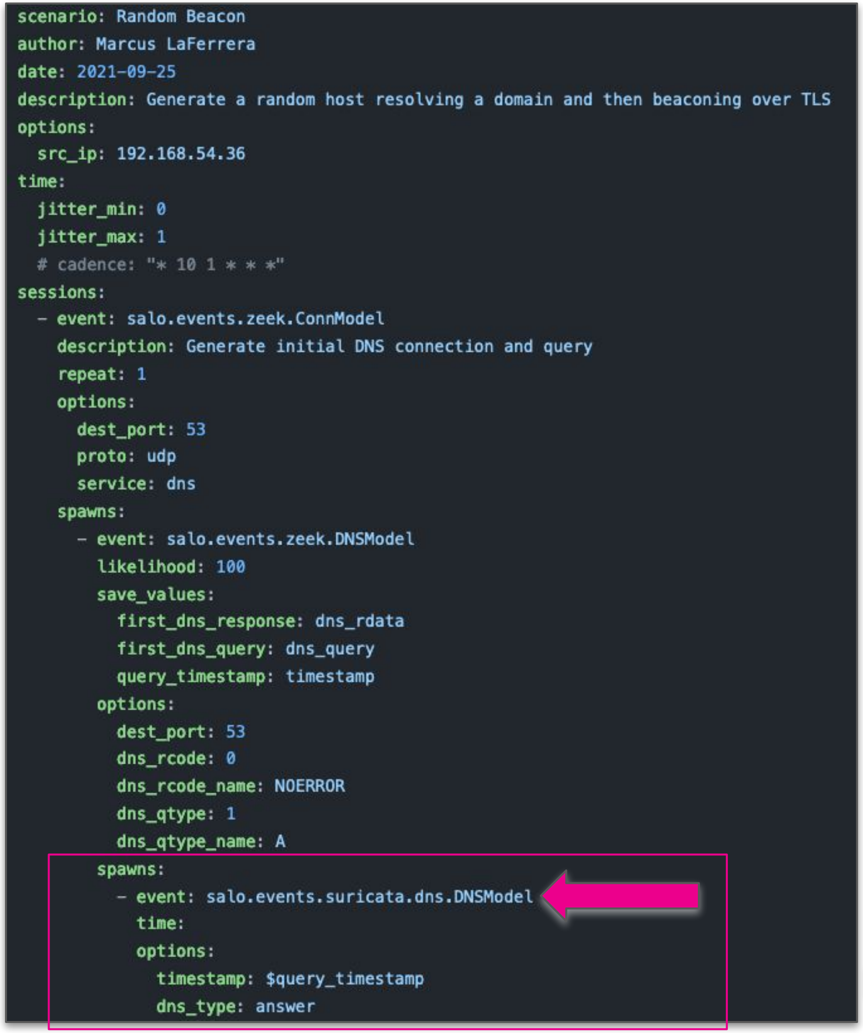


```
scenario: Random Beacon
author: Marcus LaFerrera
date: 2021-09-25
description: Generate a random host resolving a domain and then beaconing over TLS
options:
  src_ip: 192.168.54.36
time:
  jitter_min: 0
  jitter_max: 1
  # cadence: "* 10 1 * * *"
sessions:
  - event: salo.events.zeek.ConnModel
    description: Generate initial DNS connection and query
    repeat: 1
    options:
      dest_port: 53
      proto: udp
      service: dns
    spawns:
      - event: salo.events.zeek.DNSModel
        likelihood: 100
        save_values:
          first_dns_response: dns_rdata
          first_dns_query: dns_query
          query_timestamp: timestamp
        options:
          dest_port: 53
          dns_rcode: 0
          dns_rcode_name: NOERROR
          dns_qtype: 1
          dns_qtype_name: A
        spawns:
          - event: salo.events.suricata.dns.DNSModel
            time:
            options:
              timestamp: $query_timestamp
              dns_type: answer
```

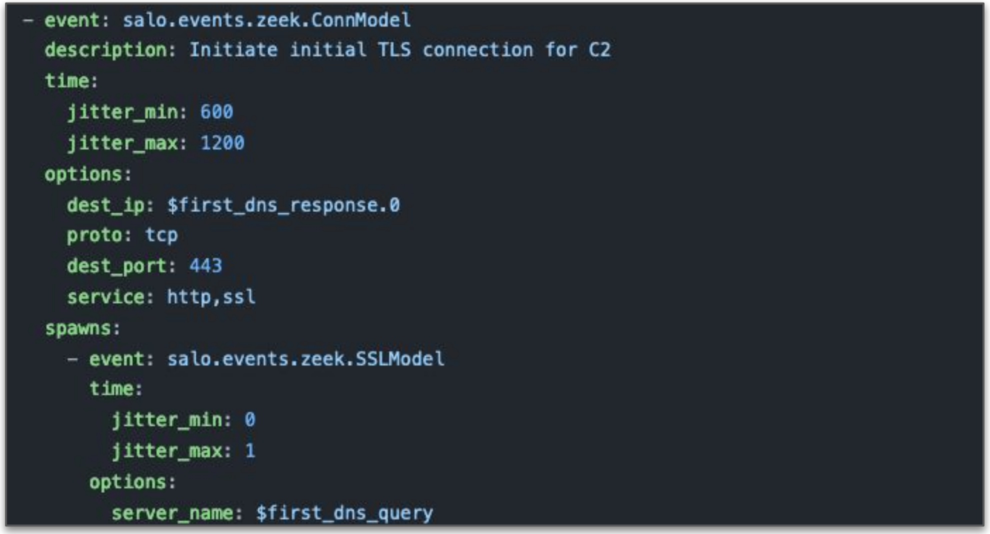


```
- event: salo.events.zeek.ConnModel
  description: Initiate initial TLS connection for C2
  time:
    jitter_min: 600
    jitter_max: 1200
  options:
    dest_ip: $first_dns_response.0
    proto: tcp
    dest_port: 443
    service: http,ssl
  spawns:
    - event: salo.events.zeek.SSLModel
      time:
        jitter_min: 0
        jitter_max: 1
      options:
        server_name: $first_dns_query
```

Recursive Spawning

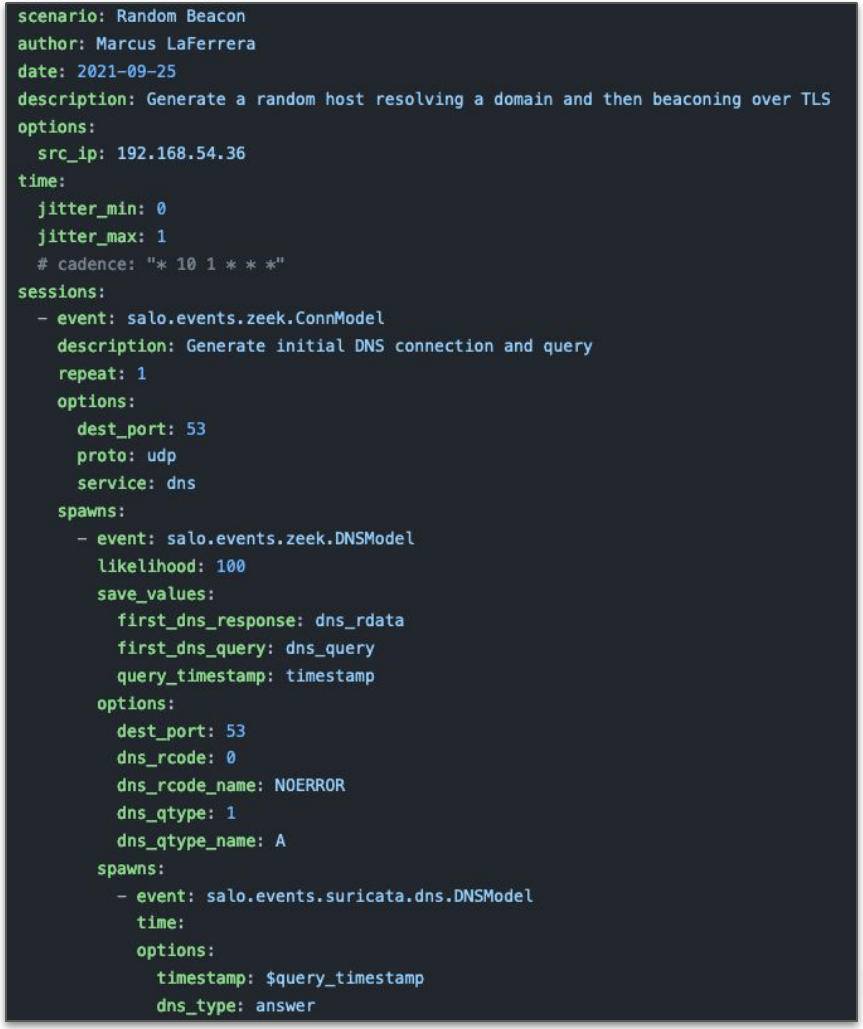


```yaml
scenario: Random Beacon
author: Marcus LaFerrera
date: 2021-09-25
description: Generate a random host resolving a domain and then beaconing over TLS
options:
  src_ip: 192.168.54.36
time:
  jitter_min: 0
  jitter_max: 1
  # cadence: "* 10 1 * * *"
sessions:
  - event: salo.events.zeek.ConnModel
    description: Generate initial DNS connection and query
    repeat: 1
    options:
      dest_port: 53
      proto: udp
      service: dns
    spawns:
      - event: salo.events.zeek.DNSModel
        likelihood: 100
        save_values:
          first_dns_response: dns_rdata
          first_dns_query: dns_query
          query_timestamp: timestamp
        options:
          dest_port: 53
          dns_rcode: 0
          dns_rcode_name: NOERROR
          dns_qtype: 1
          dns_qtype_name: A
        spawns:
          - event: salo.events.suricata.dns.DNSModel
            time:
            options:
              timestamp: $query_timestamp
              dns_type: answer
```

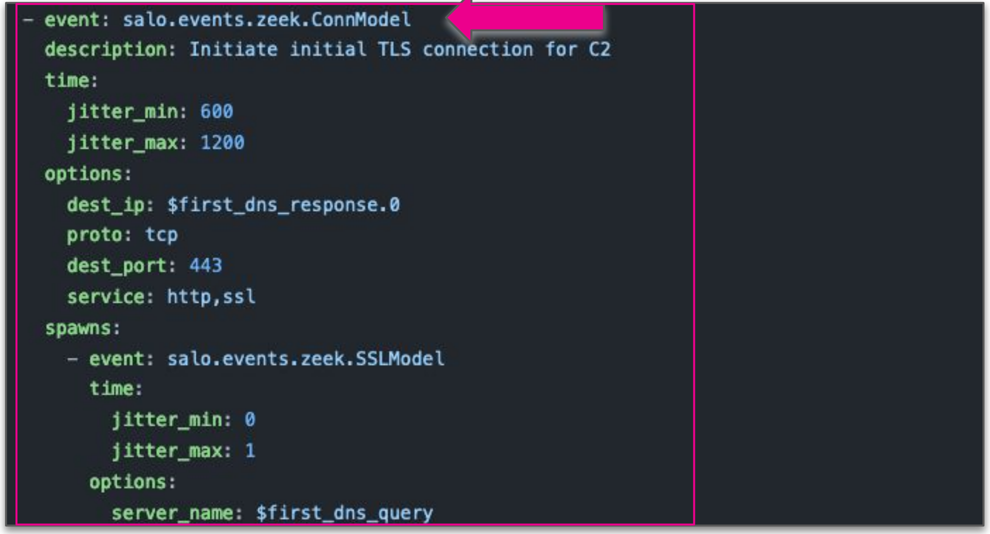


```yaml
- event: salo.events.zeek.ConnModel
  description: Initiate initial TLS connection for C2
  time:
    jitter_min: 600
    jitter_max: 1200
  options:
    dest_ip: $first_dns_response.0
    proto: tcp
    dest_port: 443
    service: http,ssl
  spawns:
    - event: salo.events.zeek.SSLModel
      time:
        jitter_min: 0
        jitter_max: 1
      options:
        server_name: $first_dns_query
```

Multiple Events in Session

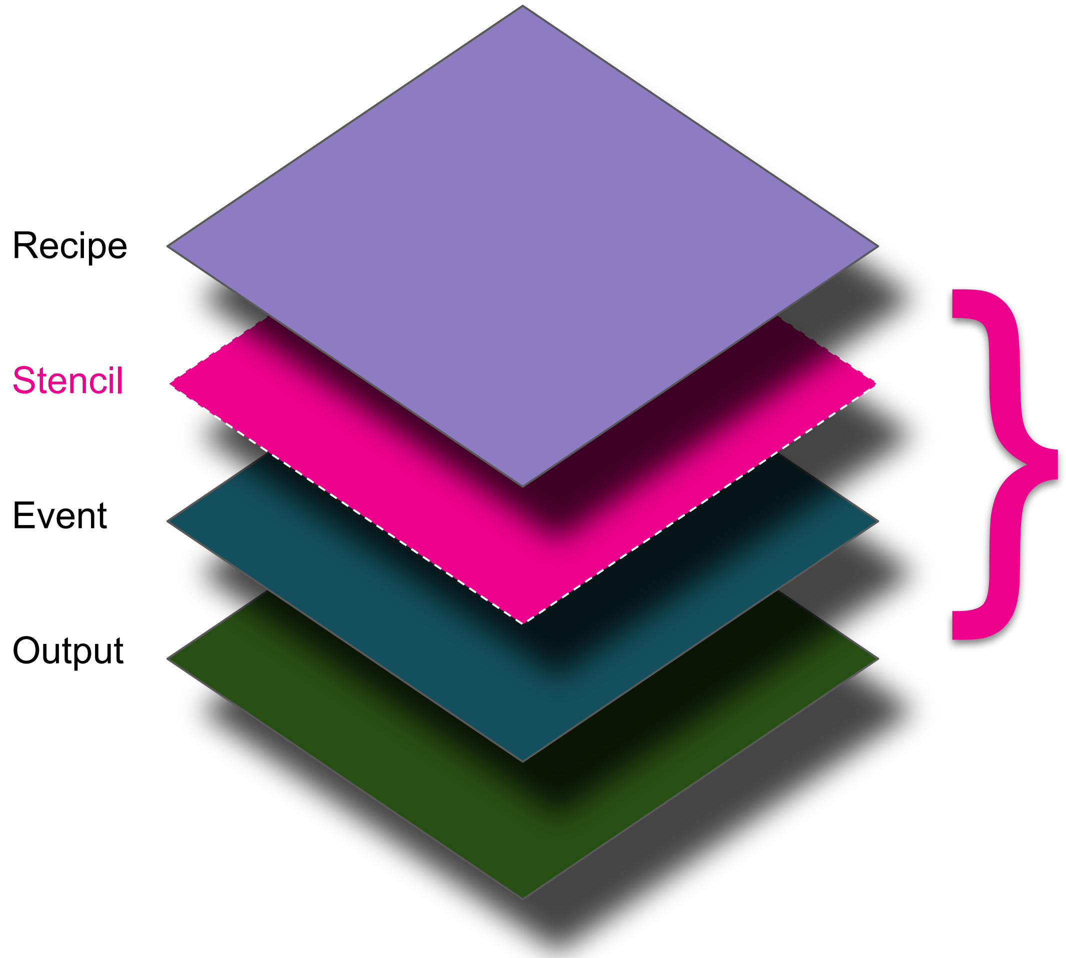


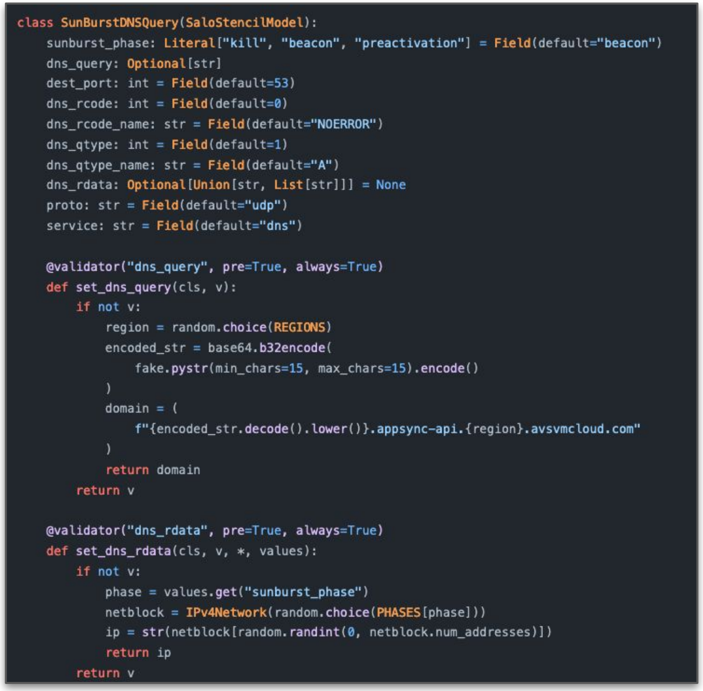
```
scenario: Random Beacon
author: Marcus LaFerrera
date: 2021-09-25
description: Generate a random host resolving a domain and then beaconing over TLS
options:
  src_ip: 192.168.54.36
time:
  jitter_min: 0
  jitter_max: 1
  # cadence: "* 10 1 * * *"
sessions:
  - event: salo.events.zeek.ConnModel
    description: Generate initial DNS connection and query
    repeat: 1
    options:
      dest_port: 53
      proto: udp
      service: dns
    spawns:
      - event: salo.events.zeek.DNSModel
        likelihood: 100
        save_values:
          first_dns_response: dns_rdata
          first_dns_query: dns_query
          query_timestamp: timestamp
        options:
          dest_port: 53
          dns_rcode: 0
          dns_rcode_name: NOERROR
          dns_qtype: 1
          dns_qtype_name: A
        spawns:
          - event: salo.events.suricata.dns.DNSModel
            time:
            options:
              timestamp: $query_timestamp
              dns_type: answer
```



```
- event: salo.events.zeek.ConnModel
  description: Initiate initial TLS connection for C2
  time:
    jitter_min: 600
    jitter_max: 1200
  options:
    dest_ip: $first_dns_response.0
    proto: tcp
    dest_port: 443
    service: http,ssl
  spawns:
    - event: salo.events.zeek.SSLModel
      time:
        jitter_min: 0
        jitter_max: 1
      options:
        server_name: $first_dns_query
```

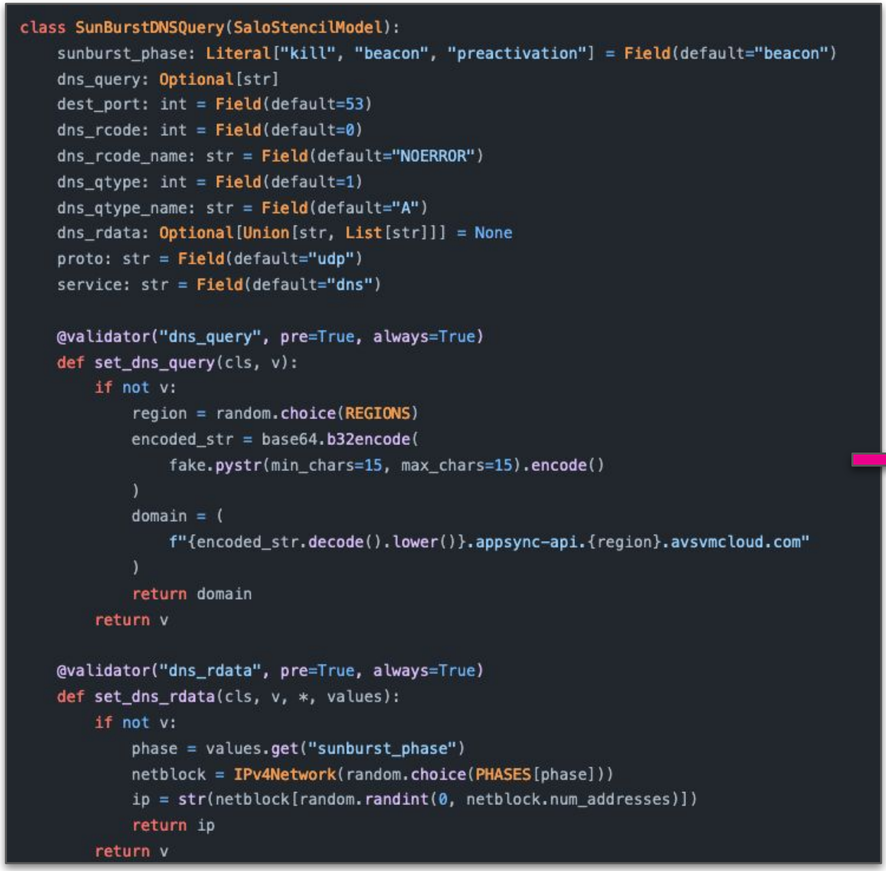
Complex data scaffolding





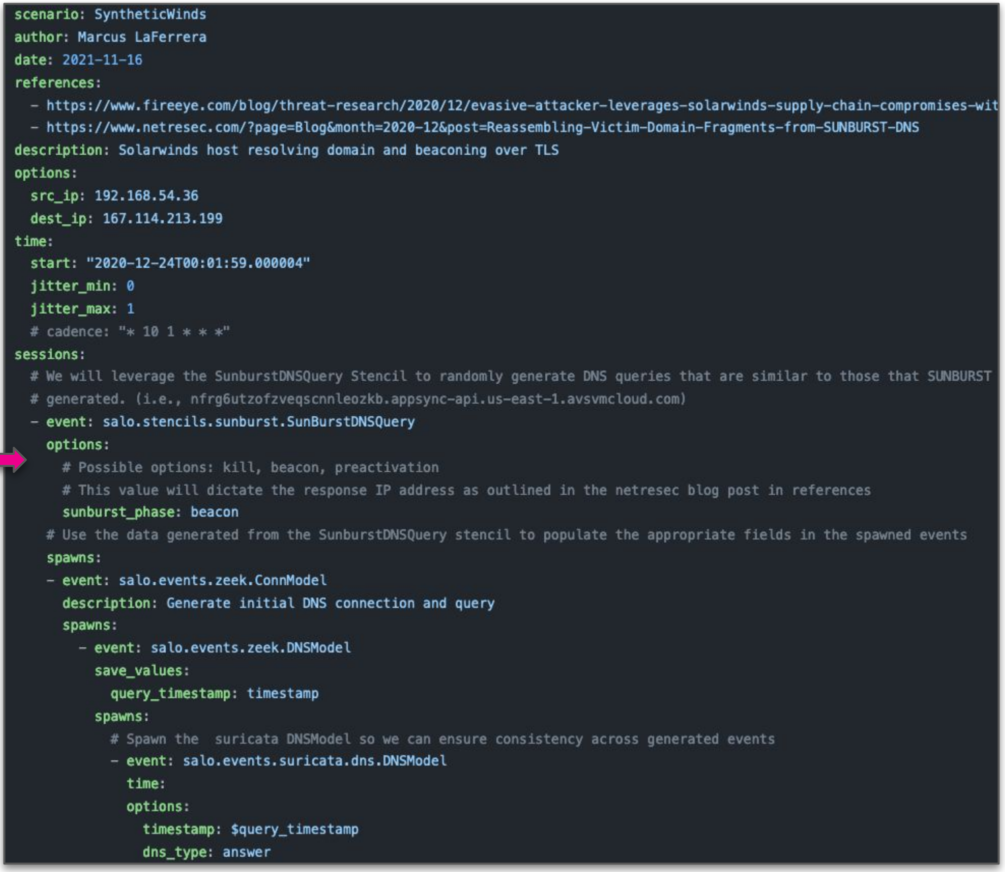
```python
class SunBurstDNSQuery(SaloStencilModel):
    sunburst_phase: Literal["kill", "beacon", "preactivation"] = Field(default="beacon")
    dns_query: Optional[str]
    dest_port: int = Field(default=53)
    dns_rcode: int = Field(default=0)
    dns_rcode_name: str = Field(default="NOERROR")
    dns_qtype: int = Field(default=1)
    dns_qtype_name: str = Field(default="A")
    dns_rdata: Optional[Union[str, List[str]]] = None
    proto: str = Field(default="udp")
    service: str = Field(default="dns")

    @validator("dns_query", pre=True, always=True)
    def set_dns_query(cls, v):
        if not v:
            region = random.choice(REGIONS)
            encoded_str = base64.b32encode(
                fake.pystr(min_chars=15, max_chars=15).encode()
            )
            domain = (
                f"{encoded_str.decode().lower()}.appsync-api.{region}.avsvmcloud.com"
            )
            return domain
        return v

    @validator("dns_rdata", pre=True, always=True)
    def set_dns_rdata(cls, v, *, values):
        if not v:
            phase = values.get("sunburst_phase")
            netblock = IPv4Network(random.choice(PHASES[phase]))
            ip = str(netblock[random.randint(0, netblock.num_addresses)])
            return ip
        return v
```

stencil

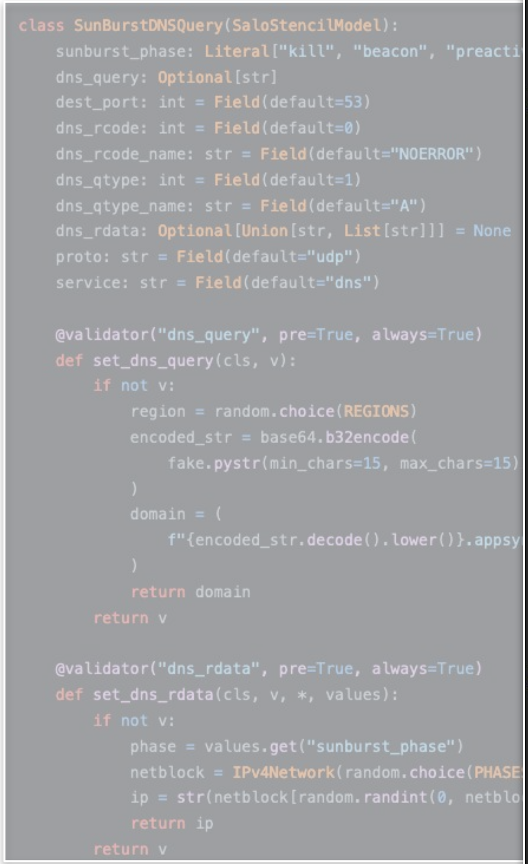


```python
class SunBurstDNSQuery(SaloStencilModel):
    sunburst_phase: Literal["kill", "beacon", "preactivation"] = Field(default="beacon")
    dns_query: Optional[str]
    dest_port: int = Field(default=53)
    dns_rcode: int = Field(default=0)
    dns_rcode_name: str = Field(default="NOERROR")
    dns_qtype: int = Field(default=1)
    dns_qtype_name: str = Field(default="A")
    dns_rdata: Optional[Union[str, List[str]]] = None
    proto: str = Field(default="udp")
    service: str = Field(default="dns")

    @validator("dns_query", pre=True, always=True)
    def set_dns_query(cls, v):
        if not v:
            region = random.choice(REGIONS)
            encoded_str = base64.b32encode(
                fake.pystr(min_chars=15, max_chars=15).encode()
            )
            domain = (
                f"{encoded_str.decode().lower()}.appsync-api.{region}.avsvmcloud.com"
            )
            return domain
        return v

    @validator("dns_rdata", pre=True, always=True)
    def set_dns_rdata(cls, v, *, values):
        if not v:
            phase = values.get("sunburst_phase")
            netblock = IPv4Network(random.choice(PHASES[phase]))
            ip = str(netblock[random.randint(0, netblock.num_addresses)])
            return ip
        return v
```

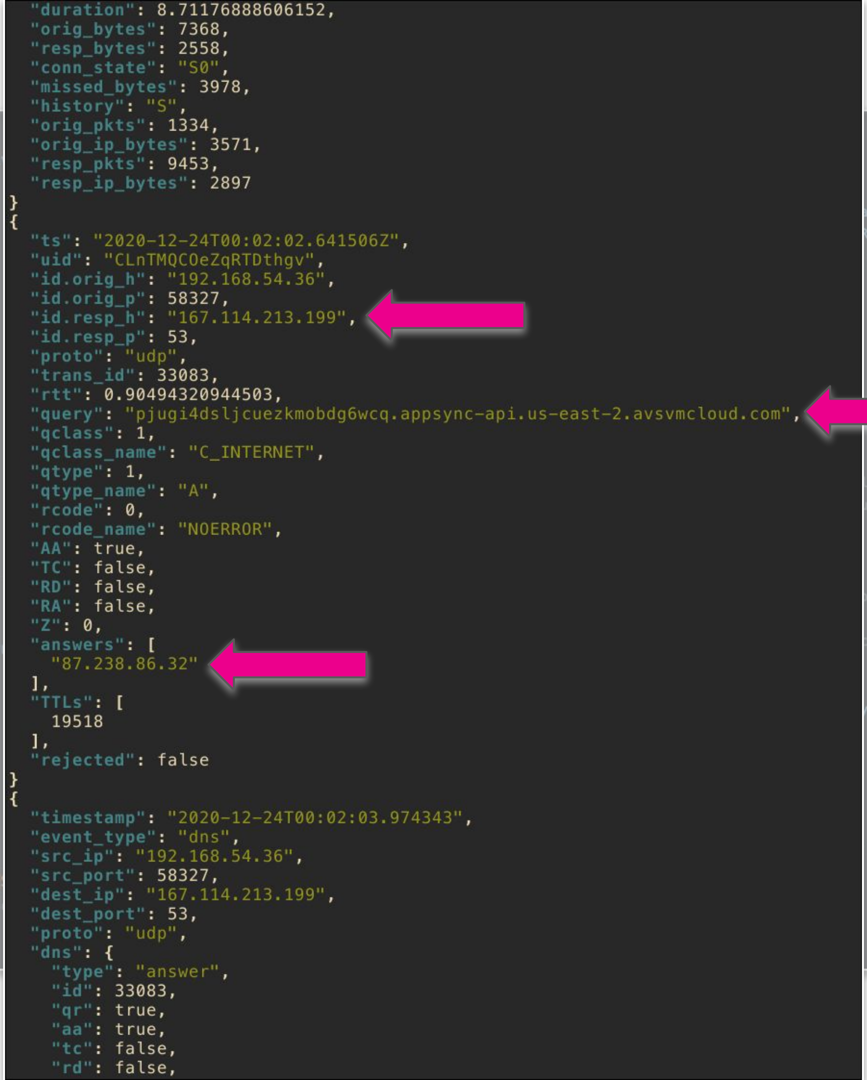
recipe



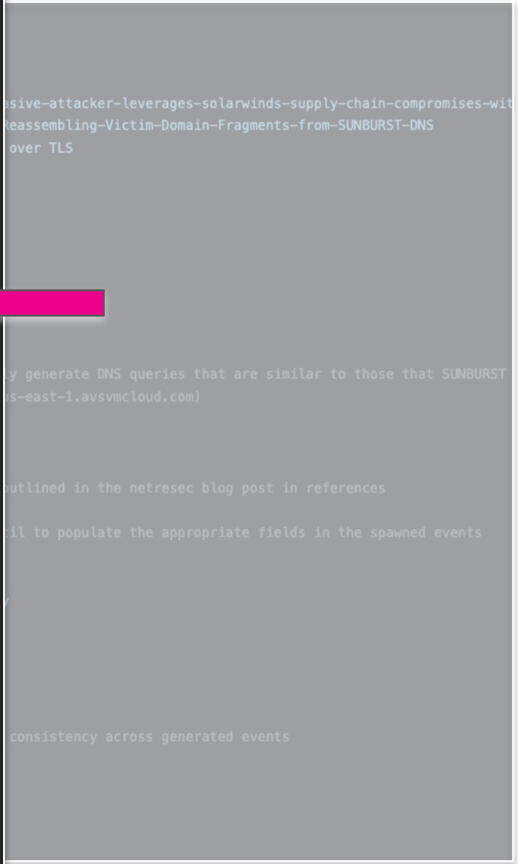
```yaml
scenario: SyntheticWinds
author: Marcus LaFerrera
date: 2021-11-16
references:
  - https://www.fireeye.com/blog/threat-research/2020/12/evasive-attacker-leverages-solarwinds-supply-chain-compromises-wit
  - https://www.netresec.com/?page=Blog&month=2020-12&post=Reassembling-Victim-Domain-Fragments-from-SUNBURST-DNS
description: Solarwinds host resolving domain and beaconing over TLS
options:
  src_ip: 192.168.54.36
  dest_ip: 167.114.213.199
time:
  start: "2020-12-24T00:01:59.000004"
  jitter_min: 0
  jitter_max: 1
  # cadence: "* 10 1 * * *"
sessions:
  # We will leverage the SunburstDNSQuery Stencil to randomly generate DNS queries that are similar to those that SUNBURST
  # generated. (i.e., nfrg6utzofzveqscnnleozkb.appsync-api.us-east-1.avsvmcloud.com)
  - event: salo.stencils.sunburst.SunBurstDNSQuery
    options:
      # Possible options: kill, beacon, preactivation
      # This value will dictate the response IP address as outlined in the netresec blog post in references
      sunburst_phase: beacon
    # Use the data generated from the SunburstDNSQuery stencil to populate the appropriate fields in the spawned events
    spawns:
    - event: salo.events.zeek.ConnModel
      description: Generate initial DNS connection and query
      spawns:
        - event: salo.events.zeek.DNSModel
          save_values:
            query_timestamp: timestamp
          spawns:
            # Spawn the  suricata DNSModel so we can ensure consistency across generated events
            - event: salo.events.suricata.dns.DNSModel
              time:
              options:
                timestamp: $query_timestamp
                dns_type: answer
```



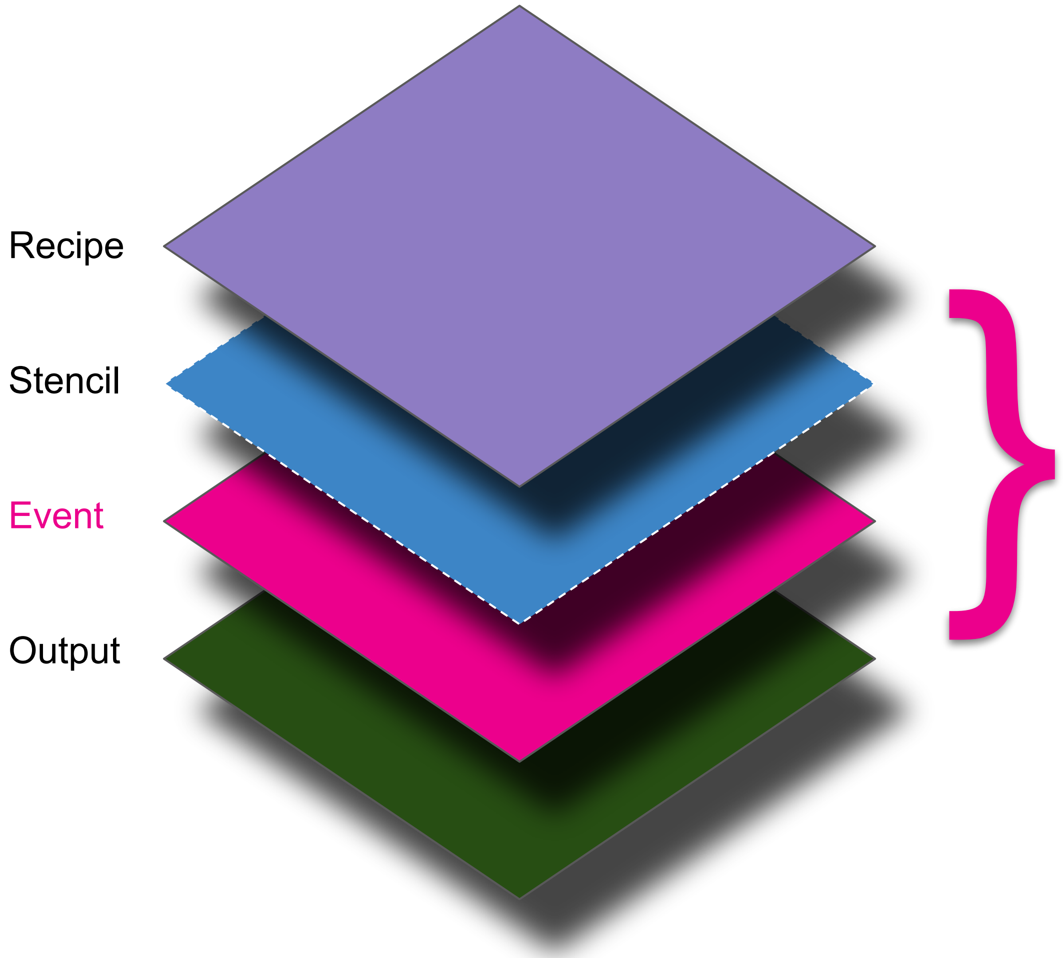
```
class SunBurstDNSQuery(SaloStencilModel):
    sunburst_phase: Literal["kill", "beacon", "preacti
    dns_query: Optional[str]
    dest_port: int = Field(default=53)
    dns_rcode: int = Field(default=0)
    dns_rcode_name: str = Field(default="NOERROR")
    dns_qtype: int = Field(default=1)
    dns_qtype_name: str = Field(default="A")
    dns_rdata: Optional[Union[str, List[str]]] = None
    proto: str = Field(default="udp")
    service: str = Field(default="dns")

    @validator("dns_query", pre=True, always=True)
    def set_dns_query(cls, v):
        if not v:
            region = random.choice(REGIONS)
            encoded_str = base64.b32encode(
                fake.pystr(min_chars=15, max_chars=15)
            )
            domain = (
                f"{encoded_str.decode().lower()}.appsy
            )
            return domain
        return v

    @validator("dns_rdata", pre=True, always=True)
    def set_dns_rdata(cls, v, *, values):
        if not v:
            phase = values.get("sunburst_phase")
            netblock = IPv4Network(random.choice(PHASE
            ip = str(netblock[random.randint(0, netblo
            return ip
        return v
```



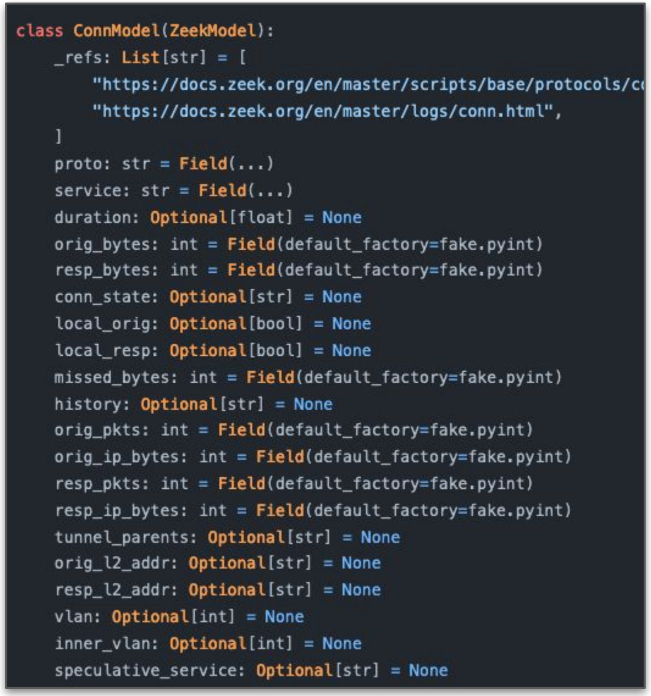
```
  "duration": 8.71176888606152,
  "orig_bytes": 7368,
  "resp_bytes": 2558,
  "conn_state": "S0",
  "missed_bytes": 3978,
  "history": "S",
  "orig_pkts": 1334,
  "orig_ip_bytes": 3571,
  "resp_pkts": 9453,
  "resp_ip_bytes": 2897
}
{
  "ts": "2020-12-24T00:02:02.641506Z",
  "uid": "CLnTMQCOeZqRTDthgv",
  "id.orig_h": "192.168.54.36",
  "id.orig_p": 58327,
  "id.resp_h": "167.114.213.199",
  "id.resp_p": 53,
  "proto": "udp",
  "trans_id": 33083,
  "rtt": 0.90494320944503,
  "query": "pjugi4dsljcuezkmobdg6wcq.appsync-api.us-east-2.avsvmcloud.com",
  "qclass": 1,
  "qclass_name": "C_INTERNET",
  "qtype": 1,
  "qtype_name": "A",
  "rcode": 0,
  "rcode_name": "NOERROR",
  "AA": true,
  "TC": false,
  "RD": false,
  "RA": false,
  "Z": 0,
  "answers": [
    "87.238.86.32"
  ],
  "TTLs": [
    19518
  ],
  "rejected": false
}
{
  "timestamp": "2020-12-24T00:02:03.974343",
  "event_type": "dns",
  "src_ip": "192.168.54.36",
  "src_port": 58327,
  "dest_ip": "167.114.213.199",
  "dest_port": 53,
  "proto": "udp",
  "dns": {
    "type": "answer",
    "id": 33083,
    "qr": true,
    "aa": true,
    "tc": false,
    "rd": false,
```



```
asive-attacker-leverages-solarwinds-supply-chain-compromises-wit
Reassembling-Victim-Domain-Fragments-from-SUNBURST-DNS
 over TLS

ly generate DNS queries that are similar to those that SUNBURST
us-east-1.avsvmcloud.com)

outlined in the netresec blog post in references

cil to populate the appropriate fields in the spawned events

 consistency across generated events
```



Individual log objects



```
class ConnModel(ZeekModel):
    _refs: List[str] = [
        "https://docs.zeek.org/en/master/scripts/base/protocols/c
        "https://docs.zeek.org/en/master/logs/conn.html",
    ]
    proto: str = Field(...)
    service: str = Field(...)
    duration: Optional[float] = None
    orig_bytes: int = Field(default_factory=fake.pyint)
    resp_bytes: int = Field(default_factory=fake.pyint)
    conn_state: Optional[str] = None
    local_orig: Optional[bool] = None
    local_resp: Optional[bool] = None
    missed_bytes: int = Field(default_factory=fake.pyint)
    history: Optional[str] = None
    orig_pkts: int = Field(default_factory=fake.pyint)
    orig_ip_bytes: int = Field(default_factory=fake.pyint)
    resp_pkts: int = Field(default_factory=fake.pyint)
    resp_ip_bytes: int = Field(default_factory=fake.pyint)
    tunnel_parents: Optional[str] = None
    orig_l2_addr: Optional[str] = None
    resp_l2_addr: Optional[str] = None
    vlan: Optional[int] = None
    inner_vlan: Optional[int] = None
    speculative_service: Optional[str] = None
```

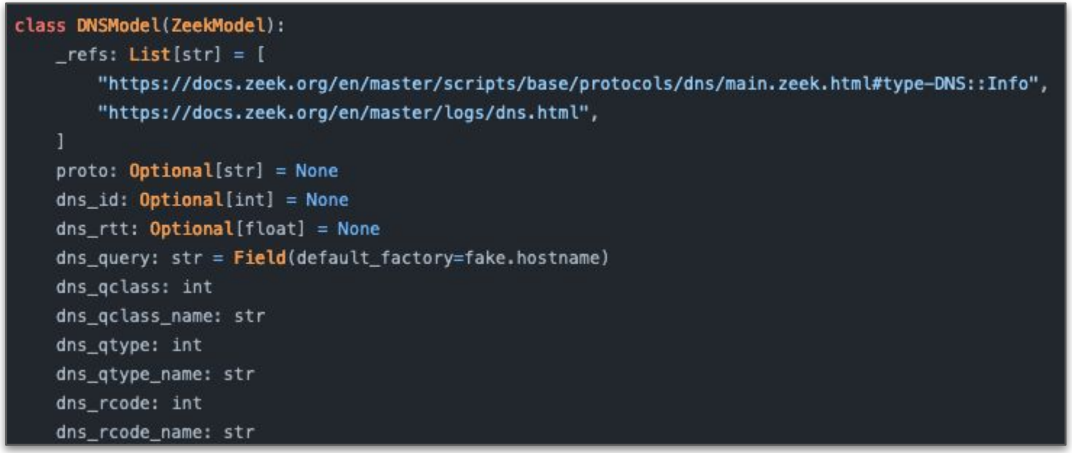
Built with pydantic*

models define the structure of log events

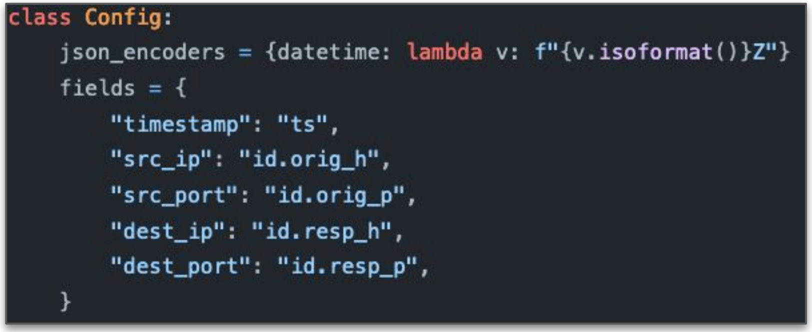
* https://pydantic-docs.helpmanual.io/

Native Model Features

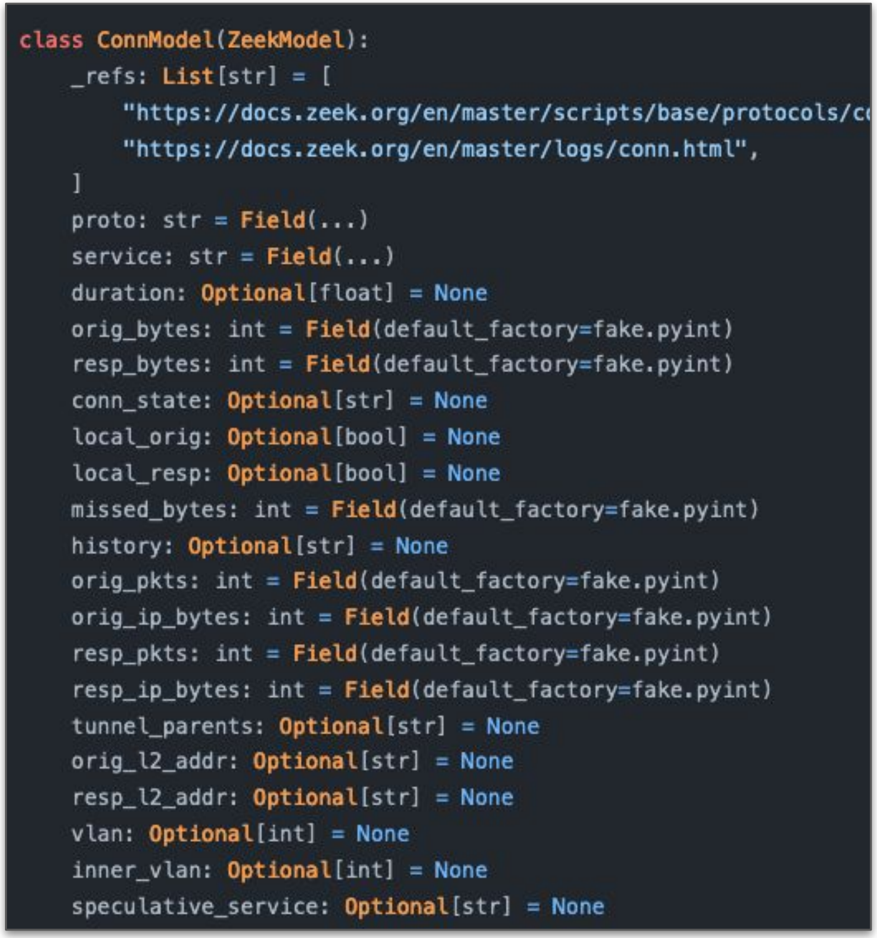
- Data validation
- Type Enforcement
- Field aliases
- Randomized field content



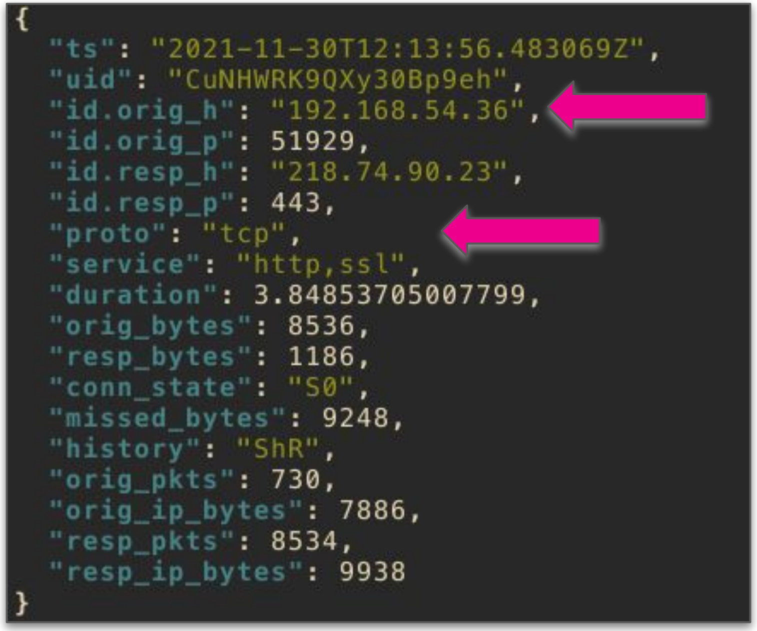
```python
class DNSModel(ZeekModel):
    _refs: List[str] = [
        "https://docs.zeek.org/en/master/scripts/base/protocols/dns/main.zeek.html#type-DNS::Info",
        "https://docs.zeek.org/en/master/logs/dns.html",
    ]
    proto: Optional[str] = None
    dns_id: Optional[int] = None
    dns_rtt: Optional[float] = None
    dns_query: str = Field(default_factory=fake.hostname)
    dns_qclass: int
    dns_qclass_name: str
    dns_qtype: int
    dns_qtype_name: str
    dns_rcode: int
    dns_rcode_name: str
```



```python
class Config:
    json_encoders = {datetime: lambda v: f"{v.isoformat()}Z"}
    fields = {
        "timestamp": "ts",
        "src_ip": "id.orig_h",
        "src_port": "id.orig_p",
        "dest_ip": "id.resp_h",
        "dest_port": "id.resp_p",
    }
```



```
class ConnModel(ZeekModel):
    _refs: List[str] = [
        "https://docs.zeek.org/en/master/scripts/base/protocols/c
        "https://docs.zeek.org/en/master/logs/conn.html",
    ]
    proto: str = Field(...)
    service: str = Field(...)
    duration: Optional[float] = None
    orig_bytes: int = Field(default_factory=fake.pyint)
    resp_bytes: int = Field(default_factory=fake.pyint)
    conn_state: Optional[str] = None
    local_orig: Optional[bool] = None
    local_resp: Optional[bool] = None
    missed_bytes: int = Field(default_factory=fake.pyint)
    history: Optional[str] = None
    orig_pkts: int = Field(default_factory=fake.pyint)
    orig_ip_bytes: int = Field(default_factory=fake.pyint)
    resp_pkts: int = Field(default_factory=fake.pyint)
    resp_ip_bytes: int = Field(default_factory=fake.pyint)
    tunnel_parents: Optional[str] = None
    orig_l2_addr: Optional[str] = None
    resp_l2_addr: Optional[str] = None
    vlan: Optional[int] = None
    inner_vlan: Optional[int] = None
    speculative_service: Optional[str] = None
```



```
{
  "ts": "2021-11-30T12:13:56.483069Z",
  "uid": "CuNHWRK9QXy30Bp9eh",
  "id.orig_h": "192.168.54.36",
  "id.orig_p": 51929,
  "id.resp_h": "218.74.90.23",
  "id.resp_p": 443,
  "proto": "tcp",
  "service": "http,ssl",
  "duration": 3.84853705007799,
  "orig_bytes": 8536,
  "resp_bytes": 1186,
  "conn_state": "S0",
  "missed_bytes": 9248,
  "history": "ShR",
  "orig_pkts": 730,
  "orig_ip_bytes": 7886,
  "resp_pkts": 8534,
  "resp_ip_bytes": 9938
}
```

Event Models Available

Zeek

- conn
- dns
- files
- http
- rdp
- smtp
- ssl

Suricata

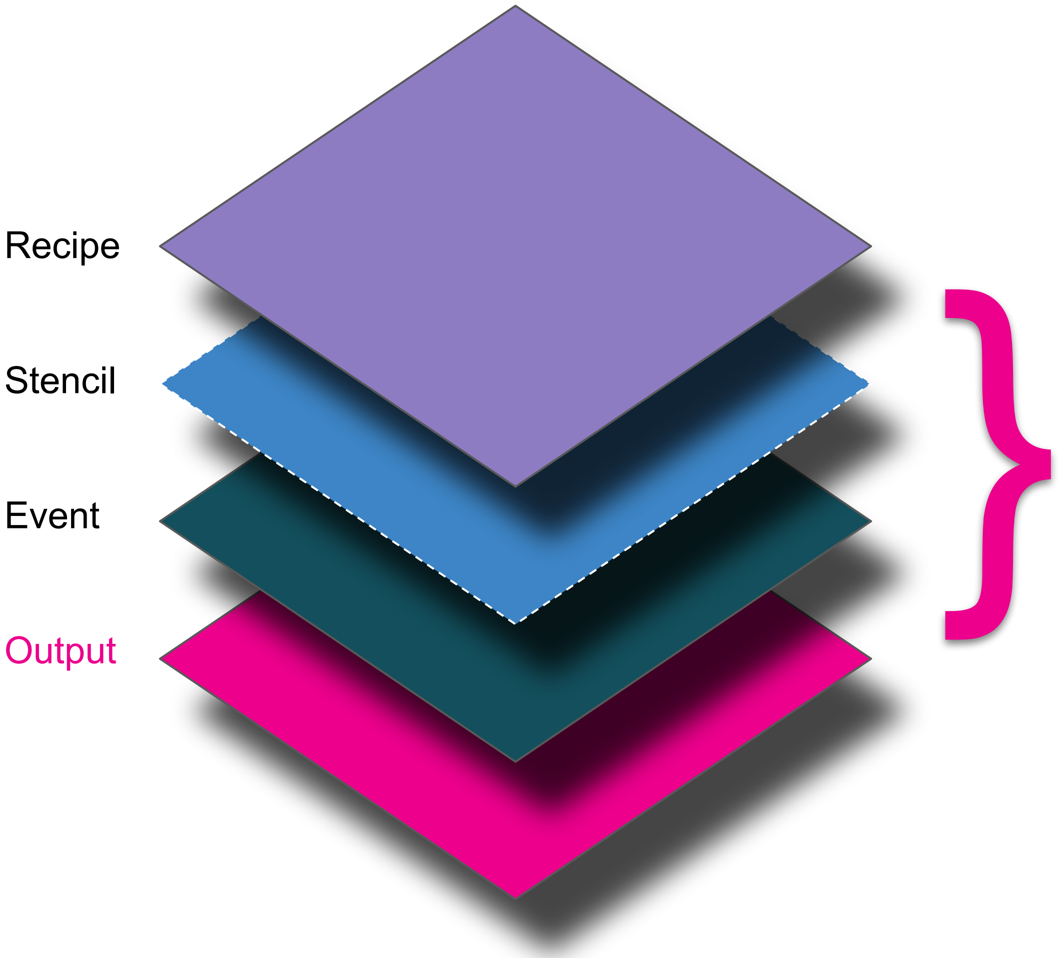
- dns
- http

Sysmon

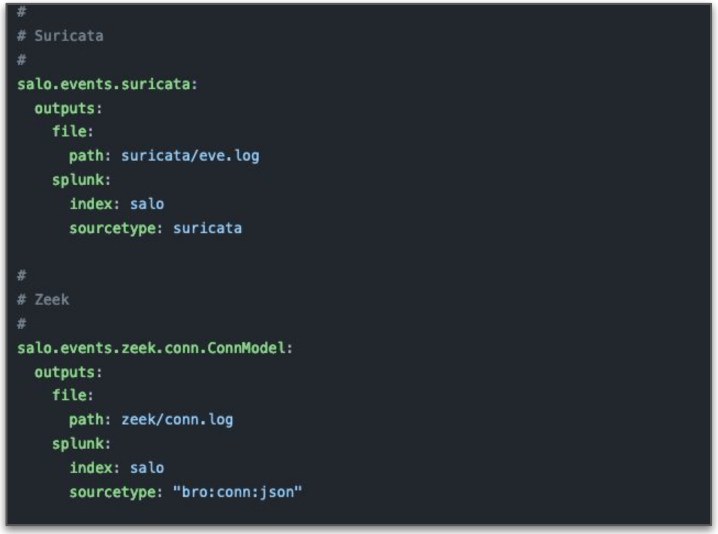
- eventcode 3

GitHub Audit

- business
- git
- hook
- integrations
- org
- repo
- repository
- team



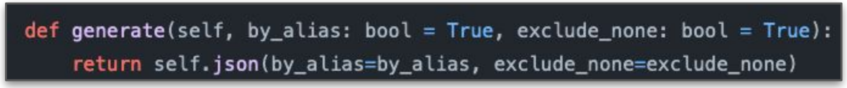
console | log file | Splunk



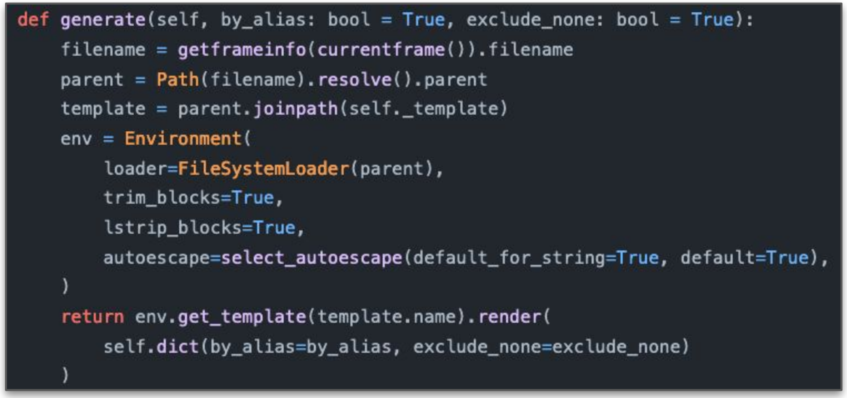
```
#
# Suricata
#
salo.events.suricata:
  outputs:
    file:
      path: suricata/eve.log
    splunk:
      index: salo
      sourcetype: suricata

#
# Zeek
#
salo.events.zeek.conn.ConnModel:
  outputs:
    file:
      path: zeek/conn.log
    splunk:
      index: salo
      sourcetype: "bro:conn:json"
```

Flexible Output Schema

- Any log format
- JSON is natively supported
- Simple to add additional output schemas



```python
def generate(self, by_alias: bool = True, exclude_none: bool = True):
    return self.json(by_alias=by_alias, exclude_none=exclude_none)
```



```python
def generate(self, by_alias: bool = True, exclude_none: bool = True):
    filename = getframeinfo(currentframe()).filename
    parent = Path(filename).resolve().parent
    template = parent.joinpath(self._template)
    env = Environment(
        loader=FileSystemLoader(parent),
        trim_blocks=True,
        lstrip_blocks=True,
        autoescape=select_autoescape(default_for_string=True, default=True),
    )
    return env.get_template(template.name).render(
        self.dict(by_alias=by_alias, exclude_none=exclude_none)
    )
```

Roadmap Wishlist

- Community involvement
- More recipes and stencils
- Additional event models

SALO

https://github.com/splunk/salo

Questions?

Marcus LaFerrera | @mlaferrera | mlaferrera@splunk.com

SALO

code
https://github.com/splunk/salo

documentation
https://splunk.github.io/salo