CHIRON

A HOME BASED ML IDS

By @rodsoto @josephzadeh

$whoami...

Rod Soto

Director of Security Research at JASK.AI. Co-founder of Hack The Valley %27. Creator/Developer of Kommand & KonTroll CTF. Co-founder of Hackmiami. Previously at Prolexic, Caspida, AKAMAI, Splunk.

Joseph Zadeh

Director of Data Science at JASK.AI. Co-founder of Hack The Valley %27. Previously at Pivotal, Kaiser Permanente, Caspida, Splunk.

Bringing Machine Learning to Home Defense

No longer a silicon valley, tech class thing

- *Our effort is to bring a basic analytics tool to the masses. We believe this technology can help everyone, not just large enterprises.*
- *We tried to make it simple and easy to use.*
- *Machine learning can help everyone at home and realize what is going on in their networks.*
- *Seeing is believing, these basic statistics will provide an overall yet sufficient picture to understand what is going on at your home network*

Advantages of ML

- *Using ML allows us to put together simple analytics, on the main elements of home networking. (Assets, Services, Protocols, Users, Traffic)*
- *ML allows us to go beyond of static signature based technologies. This aims to go beyond simple antivirus and static firewalls*
- *ML creates an scenario where detection of threats based on dynamic and multi contextual indicators is possible.*

What is this based on?

- *Lambda defense*

  Use of big data tech for distributed processing and analytics

- *Micro behaviors*

  Basic units within a exploit chain or threat

The Lambda defense at home…

- *Large organizations are currently attempting to scale intrusion detection by using big data distributed processing technologies (mainly Hadoop).*
- *CHIRON does not depend on a Hadoop infrastructure even though it does use some libraries part of the ecosystem. CHIRON cannot scale beyond a single home on current state however it is based on the same analytic principles and some of its ML frameworks*
- *By identifying, analyzing and providing key analytics for home network security, CHIRON gives home users a vision that is simply not available at this point without technical expertise and hardware.*

The Lambda Defense: Decomposing Behaviors for Intrusion Detection

DEFINITION 1. ***The Lambda Defense Pattern***

1. $|S| = |\mathbb{R}|$: *Start with the threat surface S for a fixed adversarial environment. (The examples in this paper S will usually be the threat surface associated to a large enterprise network). The space of possible behaviors is a continuum at this point.*

2. *Enumerate the threat surface: How do we reduce the space of possible behaviors to actionable units that we can process in an upstream functional workflow? To answer this question we project down to modeling only a finite set $S \approx \{TTP_1, \ldots TTP_n\}$. For a given threat surface the art of the initial design is to identify the core attack patterns needed to be modeled and to anticipate which of those parts are most likely to "drift" [19]. Better yet assign metrics to this finite approximation of the threat surface (see figure 1).*

3. *Functional Decomposition: For each TTP_i we model the behavior as $TTP_i := B_i + NB_i$. Where B_i is what we call the sequential component of the behavior and NB_i is called the non-sequential component. Formally speaking each TTP_i is actually a stochastic process (sequence of random variable) $TTP_i = \{TTP_i(t) : t \geq 0\}$.*

4. *Behavioral Model: For each sub component B_i, NB_i we assign one or more models $M_1(B_i), \ldots, M_k(B_i), M_{k+1}(NB_i), M_{k+2}(NB_i), \ldots$ and we evolve these models over time in the next step. We keep the definition of model intentionally vague here but in practice each model is usually machine learning based (supervised/semi-supervised/unsupervised) or heuristically engineered around a use case that is not driven by a pattern recognition problem. It is important to note in that we are looking for the right tool for the job and machine learning is sometimes only used as a pre-filtering step or not at all. In some cases we combine graph processing or techniques from signal analysis where it helps with overall problem of a specific model implementation.*

5. *Map/Reduce Based Model Life Cycle: We can update simultaneous models in a scalable way with a simple map and reduce operation given by a signature of the form [41]: ModelRDD.aggregate[M](seqOp: (M, Data) => M(Data), combOp: (M_{t_0}, M_{t_1}) => M_{t_2})*

Step 1: Break the problem in to use cases

DEFINITION 1. ***The Lambda Defense Pattern***

1. $|S| = |\mathbb{R}|$: *Start with the threat surface S for a fixed adversarial environment. (The examples in this paper S will usually be the threat surface associated to a large enterprise network). The space of possible behaviors is a continuum at this point.*

Step 2: Find what use cases have highest security impact

DEFINITION 1. ***The Lambda Defense Pattern***

1. $|S| = |\mathbb{R}|$: *Start with the threat surface S for a fixed adversarial environment. (The examples in this paper S will usually be the threat surface associated to a large enterprise network). The space of possible behaviors is a continuum at this point.*

2. *Enumerate the threat surface: How do we reduce the space of possible behaviors to actionable units that we can process in an upstream functional workflow? To answer this question we project down to modeling only a finite set $S \approx \{TTP_1, \ldots TTP_n\}$. For a given threat surface the art of the initial design is to identify the core attack patterns needed to be modeled and to anticipate which of those parts are most likely to "drift" [19]. Better yet assign metrics to this finite approximation of the threat surface (see figure 1).*

Step 2: Find what use cases have highest security impact

DEFINITION 1. ***The Lambda Defense Pattern***

1. $|S| = |\mathbb{R}|$: *Start with the threat surface S for a fixed adversarial environment. (The examples in this paper S will usually be the threat surface associated to a large enterprise network). The space of possible behaviors is a continuum at this point.*

2. *Enumerate the threat surface: How do we reduce the space of possible behaviors to actionable units that we can process in an upstream functional workflow? To answer this question we project down to modeling only a finite set $S \approx \{TTP_1, \ldots TTP_n\}$. For a given threat surface the art of the initial design is to identify the core attack patterns needed to be modeled and to anticipate which of those parts are most likely to "drift" [19]. Better yet assign metrics to this finite approximation of the threat surface (see figure 1).*

Step 2: Find what use cases have highest security impact

DEFINITION 1. ***The Lambda Defense Pattern***

1. $|S| = |\mathbb{R}|$: *Start with the threat surface S for a fixed adversarial environment. (The examples in this paper S will usually be the threat surface associated to a large enterprise network). The space of possible behaviors is a continuum at this point.*

2. *Enumerate the threat surface: How do we reduce the space of possible behaviors to actionable units that we can process in an upstream functional workflow? To answer this question we project down to modeling only a finite set $S \approx \{TTP_1, \ldots TTP_n\}$. For a given threat surface the art of the initial design is to identify the core attack patterns needed to be modeled and to anticipate which of those parts are most likely to "drift" [19]. Better yet assign metrics to this finite approximation of the threat surface (see figure 1).*

Step 3: Decompose the problem into two types of computation

Mapping Behaviors to

DEFINITION 1. ***The Lambda Defense Pattern***

1. $|S| = |\mathbb{R}|$: *Start with the threat surface S for a fixed adversarial environment. (The examples in this paper S will usually be the threat surface associated to a large enterprise network). The space of possible behaviors is a continuum at this point.*

2. *Enumerate the threat surface: How do we reduce the space of possible behaviors to actionable units that we can process in an upstream functional workflow? To answer this question we project down to modeling only a finite set $S \approx \{TTP_1, \ldots TTP_n\}$. For a given threat surface the art of the initial design is to identify the core attack patterns needed to be modeled and to anticipate which of those parts are most likely to "drift" [19]. Better yet assign metrics to this finite approximation of the threat surface (see figure 1).*

3. *Functional Decomposition: For each TTP_i we model the behavior as $TTP_i := B_i + NB_i$. Where B_i is what we call the sequential component of the behavior and NB_i is called the non-sequential component. Formally speaking each TTP_i is actually a stochastic process (sequence of random variable) $TTP_i = \{TTP_i(t) : t \geq 0\}$.*

In complexity theory, the notion of **P-complete** decision problems is useful in the analysis of both:

1. which problems are difficult to parallelize effectively, and;
2. which problems are difficult to solve in limited space.

Arbitrary User Behavior = Sequential Component + "Un-Ordered" Component

1. Count()
2. Average()
3. Time series()
4. Local state computations
   - Per user/IP/account/...

Examples

- **Sequential Behaviors**
  - Lead to Parallelize (NC Complete Complexity)
  1. ***Exploit Chains***
  2. ***Timing Analysis (Periodicity)***
  3. ***Active Directory Sequence***
  4. ***Authentication Graph***
- **Non Sequential Behaviors**
  1. ***Fingerprints***
  2. ***Grouping Behaviors***
  3. ***Application Counts***
  4. ***Rare file extension counts for Webshell detection***

Sequential Behaviors

1. Count()
2. Rank()
3. Median()
4. Keeping that kind of keeps track of global state

Machine Learning Computations

- Matrix Inversion or Stochastic Gradient Descent

Step 4: Build an ML Model for each important sub-behavior

DEFINITION 1. ***The Lambda Defense Pattern***

1. *$|S| = |\mathbb{R}|$: Start with the threat surface S for a fixed adversarial environment. (The examples in this paper S will usually be the threat surface associated to a large enterprise network). The space of possible behaviors is a continuum at this point.*

2. *Enumerate the threat surface: How do we reduce the space of possible behaviors to actionable units that we can process in an upstream functional workflow? To answer this question we project down to modeling only a finite set $S \approx \{TTP_1, \ldots TTP_n\}$. For a given threat surface the art of the initial design is to identify the core attack patterns needed to be modeled and to anticipate which of those parts are most likely to "drift" [19]. Better yet assign metrics to this finite approximation of the threat surface (see figure 1).*

3. *Functional Decomposition: For each TTP_i we model the behavior as $TTP_i := B_i + NB_i$. Where B_i is what we call the sequential component of the behavior and NB_i is called the non-sequential component. Formally speaking each TTP_i is actually a stochastic process (sequence of random variable) $TTP_i = \{TTP_i(t) : t \geq 0\}$.*

4. *Behavioral Model: For each sub component B_i, NB_i we assign one or more models $M_1(B_i), \ldots, M_k(B_i), M_{k+1}(NB_i), M_{k+2}(NB_i), \ldots$ and we evolve these models over time in the next step. We keep the definition of model intentionally vague here but in practice each model is usually machine learning based (supervised/semi-supervised/unsupervised) or heuristically engineered around a use case that is not driven by a pattern recognition problem. It is important to note in that we are looking for the right tool for the job and machine learning is sometimes only used as a pre-filtering step or not at all. In some cases we combine graph processing or techniques from signal analysis where it helps with overall problem of a specific model implementation.*

Each Model can be batch, real-time or hybrid mode

Domain Name	TotalCnt	RiskFactor	AGD	SessionTime	RefEntropy	NullUs
europartsplus.org	144	6.05	1	1	0	0
jjeyv2u37an30.com	6192	5.05	0	1	0	0
cdn4s.steelhousemedia.com	107	3	0	0	0	0
log.tagcade.com	111	2	0	1	0	0
go.vidprocess.com	170	2	0	0	0	0
statse.webtrendslive.com	310	2	0	1	0	0
cdn4s.steelhousemedia.com	107	1	0	0	0	0
log.tagcade.com	111	1	0	1	0	0

```
val rfModel: RandomForestModel = RandomForest.trainClassifier(trainData.union(cvData),
  numClasses, categoricalFeaturesInfo, loopNumTrees, featureSubsetStrategy, loopImpurity, loopDepth, loopBins)
import org.apache.spark.mllib.tree.model.RandomForestModel
```

```
//decorate the (domain,feature vector) pair with predicted output of the model
//use pair RDD's to recover the full data we need for anomaly generation by a inner join
val predictedOutput = vectorizeFeatures.map(x => (x._1, randomForestModel.predict(x._2)))
```

Step 5: Operationalize the Model Life Cycle

•Whe

1.

2.

3.

speaking each TTP_i is actually a stochastic process (sequence of random variable) $TTP_i = \{TTP_i(t) : t \geq 0\}$.

4. Behavioral Models: For each sub component B_i, NB_i we assign one or more models $M_1(B_i), \ldots, M_k(B_i), M_{k+1}(NB_i), M_{k+2}(NB_i), \ldots$ and we evolve these models over time in the next step. We keep the definition of model intentionally vague here but in practice each model is usually machine learning based (supervised/semi-supervised/unsupervised) or heuristically engineered around a use case that is not driven by a pattern recognition problem. It is important to

t tool for the
nly used as a
ases we com-
signal anal-
of a specific

e can update
with a simple
nature of the
p: (M, Data)
M_{t_2})

DEFINITION 1. ***The Lambda Defense Pattern***

1. *$|S| = |\mathbb{R}|$: Start with the threat surface S for a fixed adversarial environment. (The examples in this paper S will usually be the threat surface associated to a large enterprise network). The space of possible behaviors is a continuum at this point.*

2. *Enumerate the threat surface: How do we reduce the space of possible behaviors to actionable units that we can process in an upstream functional workflow? To answer this question we project down to modeling only*

a fini
surfa
core
ipate
Bette
of the

3. *Func*
the be
we co
NB_i

an update
a simple
ure of the
(M, Data)
)

Figure 3: Model Stabilization Over Time

How do we programmatically learn new patterns over time?

The Lambda Defense: A Complex Design Pattern

DEFINITION 1. ***The Lambda Defense Pattern***

1. $|S| = |\mathbb{R}|$: *Start with the threat surface S for a fixed adversarial environment. (The examples in this paper S will usually be the threat surface associated to a large enterprise network). The space of possible behaviors is a continuum at this point.*

2. *Enumerate the threat surface: How do we reduce the space of possible behaviors to actionable units that we can process in an upstream functional workflow? To answer this question we project down to modeling only a finite set $S \approx \{TTP_1, \ldots TTP_n\}$. For a given threat surface the art of the initial design is to identify the core attack patterns needed to be modeled and to anticipate which of those parts are most likely to "drift" [19]. Better yet assign metrics to this finite approximation of the threat surface (see figure 1).*

3. *Functional Decomposition: For each TTP_i we model the behavior as $TTP_i := B_i + NB_i$. Where B_i is what we call the sequential component of the behavior and NB_i is called the non-sequential component. Formally speaking each TTP_i is actually a stochastic process (sequence of random variable) $TTP_i = \{TTP_i(t) : t \geq 0\}$.*

4. *Behavioral Model: For each sub component B_i, NB_i we assign one or more models $M_1(B_i), \ldots, M_k(B_i), M_{k+1}(NB_i), M_{k+2}(NB_i), \ldots$ and we evolve these models over time in the next step. We keep the definition of model intentionally vague here but in practice each model is usually machine learning based (supervised/semi-supervised/unsupervised) or heuristically engineered around a use case that is not driven by a pattern recognition problem. It is important to note in that we are looking for the right tool for the job and machine learning is sometimes only used as a pre-filtering step or not at all. In some cases we combine graph processing or techniques from signal analysis where it helps with overall problem of a specific model implementation.*

5. *Map/Reduce Based Model Life Cycle: We can update simultaneous models in a scalable way with a simple map and reduce operation given by a signature of the form [41]: ModelRDD.aggregate[M](seqOp: (M, Data) => M(Data), combOp: (M_{t_0},M_{t_1}) => M_{t_2})*

One More Technical Definition

- The reference to the pattern comes from "Lambda Architecture" Coined by Nathan Mars

- GOTO 2012 • Runaway Complexity in Big Data Systems...and a Plan to Stop it • Nathan Marz: https://www.youtube.com/watch?v=ucHjyb6jv08

Lambda Architecture

- *Architecture is described by three simple equations:*

batch view = function(all data)
realtime view = function(realtime view, new data)
query = function(batch view, realtime view)

Layered ML

- **Shades of Grey**
  - *The layered security approach fuses multiple pieces of evidence together using a combination of models rules and statistics to move past the traditional detection solutions*
- **Sequencing Security Behaviors**
  - *The next generation SIEM indexes all outputs and outcomes and uses rules, statistics, IOC's and intelligence along with the fusion of ML models to build a central nervous system view of all possible risks in an environment*
- **Evidence Fusion**: Overlay risk categories on top of each system in the environment

  Defense Science Board, ***Resilient Military Systems and the Advanced Cyber Threat*** *(Jan. 2013)*

Figure ES.2 Risk Management Parameters

Advesarial Models

Non-Automatable Actions: Hybrid Human/Computer Analysis

Automatable Actions: Good for ML

Operator Time is Valuable!

- *Googles Experience with ML in Cybersecurity: https://web.stanford.edu/class/cs259d/lectures/Session11.pdf*

Google

Why is statistical anomaly detection for security hard?

	Learning	Cost of error FP / FN	Goal	Attacker
Anomaly detection	Unsupervised	Medium / High	Classify & ***explain***	Adaptive
Spam detection	Supervised	High / Low	Classify	Adaptive
Product recommendation	Supervised	Low / Low	Classify	N/A

For more on this topic, consider "On Using Machine Learning For Network Intrusion Detection", Sommer and Paxson, Oakland 2010.

Google

An alternative: rule-based detection

Manually created rules

→ Characterize attacks or deviations, not normalcy

- Locality: each rule covers a small set of logs and features
- Explainability: direct connection between rules and alerts
- Specificity: make better use of analysts' expertise

How do we Detect Webshells Using ML: Global Stats

Anomalies on rare paths

U→S

S→U !!

U→U (LAN to LAN)

S→S (DMZ to LAN)!!

Seeing the Analytic In Action

Once identity resolution/learning process is complete we create new anomalies based on new paths/actions that are rare for a particular population profile

Phishing Attacks Had A Big Year In 2016, But Exploit Kits Will Make A Comeback In 2017

Spandas Lui
Feb 7, 2017, 12:30pm · Filed to: Australian Stories

Share

Image: iStock

Last year saw a steep increase in malware distributed through email attachments in phishing attacks but 2017 will be the year of malware exploit kits, according to a report by security vendor Malwarebytes. We take a closer look at malware trends from 2016 and

Successful Exploit Chain Post Phish (Angler EK)

User clicks on email link

Exploit is looking for vulnerability

Exploit is successful ->payload delivered

This is what it looks like in web proxy logs

1. ***Initial Redirect From Poisoned Domain:*** *[29/Apr/2015:16:52:23 -0700] "Nico Rosberg" 192.168.122.177 69.162.78.253 1500 200 TCP_HIT "GET* ***http://forbes.com/gels-contrariness-domain-punchable/1.html/54882841592027674 8*** *HTTP/1.1" "Internet Services" "low risk"* ***"text/html"*** *604 142 "Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; WOW64; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0)" "http://forbes.com/gels-contrariness-domain-punchable/1.html" "-" "0" "" "-"*

2. ***Flash Exploit:*** *[29/Apr/2015:16:52:26 -0700] "Nico Rosberg" 192.168.122.177 69.162.78.253 1500 200 TCP_HIT "**GET*** ***http://portcullisesposturen.europartsplus.org/lMvOBBZKDLqAJ1HD-o2t5hMMNyzBLN_q4kafJkVNqJV1n1md*** *HTTP/1.1" "Internet Services" "low risk"* ***"application/x-shockwave-flash"*** *518 821 "Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; WOW64; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0)" "http://forbes.com/gels-contrariness-domain-punchable/1.html/548828415920276748" "-" "0" "" "-"*

3. ***Payload:*** *[29/Apr/2015:16:52:27 -0700] "Nico Rosberg" 192.168.122.177 69.162.78.253 1500 200 TCP_HIT "**GET*** ***http://portcullisesposturen.europartsplus.org/UX7u1YkbNn8FUV6Q2VtEzLj_p_gLvRklWEWmz3r7Ug8suRiY_*** *HTTP/1.1" "Internet Services" "low risk"* ***"application/octet-stream"*** *136 915 "" "" "-" "0" "" "-"*

Micro behaviors what is that?

- *Basic units present within a context of a threat or exploit kill chain. In order to define Micro behaviors we need to label data and give it a value within a context of threat or exploit kill chain.*
- *Here is a list of Micro behaviors present in a ransomware attack: - Payload delivery. Focused on traffic to malicious sites and the related indicators when malicious code is served. Including things such as URI entropy, redirects, domain generated by algorithms (DGAs), types and sequences of MIME content presented to victim during payload delivery.*

Micro behaviors what is that? II

- Call backs (Phone home) patterns, including user agent, URI strings, HTTP "GET" or "POST" requests, DNS queries, URI strings, frequency of call-backs, periodicity of connections.

- Covert Channel indicators, such as non HTTP traffic (HTTPS), and non DNS traffic present during such communications.

- Entropy in URI/URL (number of dots, special characters, URL crazy) - Whois (registrar with bad rep / geolocation / bulletproof hosting

What are Micro behaviors? III

- URL Shorteners (Tiny URL, bit.ly, etc)

- Spelling errors, passwords fields

- Presence of iframes, javascript, pagerank?, source of images - Presence of .exe .pdf .bat .ps1 .ps .bin .bat .jar .bin .zip

By analyzing these Micro behaviors within a context of exploitation and entity behavior we can develop methods for detection. Then train the models, then test detection.

How does Chiron learn?

1. *Key insight is to take action on new behaviors observed on a small network segment/s*
2. *New Behavior means a change point in underlying data:*
   a. *New service interaction (IP1 → IP2 via port /app never observeed before)*
   b. *New DMZ interaction*
   c. *New*
3. *Exploit detection learning (standalone model based off Aktaion)*

Chiron Architecture Notes

ML Exploit Detection:
Aktaion V2
parsers
analytics
exports
Bro Logs
NMap
Pof
Custom
Parsing
Bro Data Frame
NMap Data Frame
Pof Data Frame
Write each file to disk and normalize into single format
ELK + Viz
NMap
New asset discovered :
automated NMAP
Cron Job 1 -
(Sensor job) :
Run NMap 1 hour
Pof deamon mode
Bro Daemon mode
Cron Job 2 -
(Read / Move Once) :
0 24 * * * .... crontab python3 pof_parser(/pofdirectory/) |
move_old_logs(/pofdirectory/)
Cron Job 3 -
(Export / Redirect) :
Write data to disk. If NMap discovers any new activity, trace it and re-write data.

Analytics
AktaionV2: BRO
Bro Data Frame
NMap Data Frame
Pof Data Frame
Bro Public .conn Log Dst IPs
Bro Public .conn LogSrc IPs
Bro Private .conn Log Dst IPs
Bro Private .conn Log Src IPs
Private IPs
Public IPs
Private IPs
Public IPs
Public IP DF (normalize and make each IP a single row, nmap_cnt, pof_cnt, etc.).
Private IP DF
Top N Stats
Write to Disk
Top N Stats
Write to Disk
Step 1
Step 2
Step 3
Step 4

V2 of Architecture

Leverage OS query to do active defense interaction for targeting one specific use case:

Given a network flow on the home Router what process on what host generated that flow.

Automate the response workflow for this problem for the home environment

Scaling Op-Sec to the Home Environment

- *There are tools for defense but there is a learning curve to deployment models see for instance "DEFCON 17: Asymmetric Defense: How to Fight Off the NSA Red Team with Five People or Less"* https://www.defcon.org/images/defcon-17/dc-17-presentations/defcon-17-efstratios_gavas-nsa_red_team.pdf

Some of the tools mentioned + bootstraping FreeBSD

Monowall, IPCop, Untangle, pfSense

Chiron Deployment Configuraitons Scoped

- *Passive network tap*
- *Inline*
- *Shadow IDS (active response disabled)*
  - *Receive but no Transmit for a hardwired network tap*

Chiron

Active Defense

Analytics + leanring

Aktaion

Active Defense for Change Point Detection on new network behaviors

Challenges in home networking defense

- *Internet of Things (toasters, refrigerators, thermostats, cameras, wearables, AI, routers, printers, etc)*
- *Unpatched vulnerable devices become Zombies of very large botnets feeding the crime ecosystem (CPEs are prime target)*
- *Common users have no idea or knowledge of what to do with all these "home" infrastructure becoming prey of snake oil pharisees*
- *Huge privacy risks, (I.E Unknown external IP accessing webcam or child monitor, somebody unlocking doors remotely, large data transfer from Home NAS to unknown IP address)*

Challenges in home network defense II

- *Wifi poachers, p2p piracy, someone using your home network as "VPN"*
- *Targeted attacks against your organization via your home network*
- *Targeted attacks against personal property (identity, finance, insurance)*
- *Port 25/110/143/993/465 running?. Is your home network being used for SPAM*
- *9001/9030/9050/9150 ? Why is somebody using TOR at home?*

CHIRON - ML network home security use cases

- *Malicious file downloads*
- *Fingeprints users, services, and protocols*
- *High risk users, assets, domains*
- *Data usage*
- *North - South traffic*
- *Relationships between assets*
- *Social Media usage*
- *Odd application/traffic/services*

Bringing Machine Learning to home defense

- *You do not need a hadoop cluster at home in order to use machine learning*
- *There are several open source tools that we can use to have a decent proof of concept of network analytics plus machine learning*
- *No "SNAKE OIL" OPEN SOURCE github.com/jzadeh/chiron*
- *No need for a network tap, we are using passive detection tool p0f in combination with NMAP. You can use a network tap and pipe the output if you want.*
- *It can be run from security onion*
- *FIRE AND FORGET NO NEED TO CODE*

A case to bring ML to home defense

- *Home networks are your castle*
- *Pivot of targeting/phishing employees at their homes (remote workers)*
- *IoT -- big leakers of private info, unpatch, vulnerable, significant risk*
- *Do you live in highly dense building? Anybody poaching your Internet service?*
- *Where all those devices connect to?*
- *Where all my users connecting to?*
- *Is there any suspicious NORTH-SOUTH traffic?*
- *Dynamic asset discovery*

A case to bring ML to home defense

Three basic tasks:

- *Perform basic discovery and analytics of home network assets (IoTs, Workstations, Laptops, Servers, Routers)*
- *Fingetprints users, services, and protocols*
- *Applies analytics to users and devices (Averarge session length, Traffic, Visited sited)*

The Tool

Github

http://www.github.com/jzadeh/chiron-elk

Requirements

- *Can be deployed on a VM - 60GB HD 1GB RAM*
- *Chiron is good for a home network 5 - 10 users*
- *30 day log retention standard*

ML-Exploit Detection: Aktaion V2

parsers

- Bro Logs
- NMap
- Pof

Custom Parsing

analytics

- Bro Data Frame
- NMap Data Frame
- Pof Data Frame

exports

Write each file to disk and normalize into single format

ELK + Viz

NMap

New asset discovered : *automated NMAP*

Cron Job 1 - (Sensor job) :
Run NMap 1 hour
Pof deamon mode
Bro Daemon mode

Cron Job 2 - (Read / Move Once) :
*0 24 * * * .... crontab python3 pof_parser(/pofdirectory/) | move_old_logs(/pofdirectory/)*

Cron Job 3 - (Export / Redirect) :
Write data to disk. If NMap discovers any new activity, trace it and re-write data.