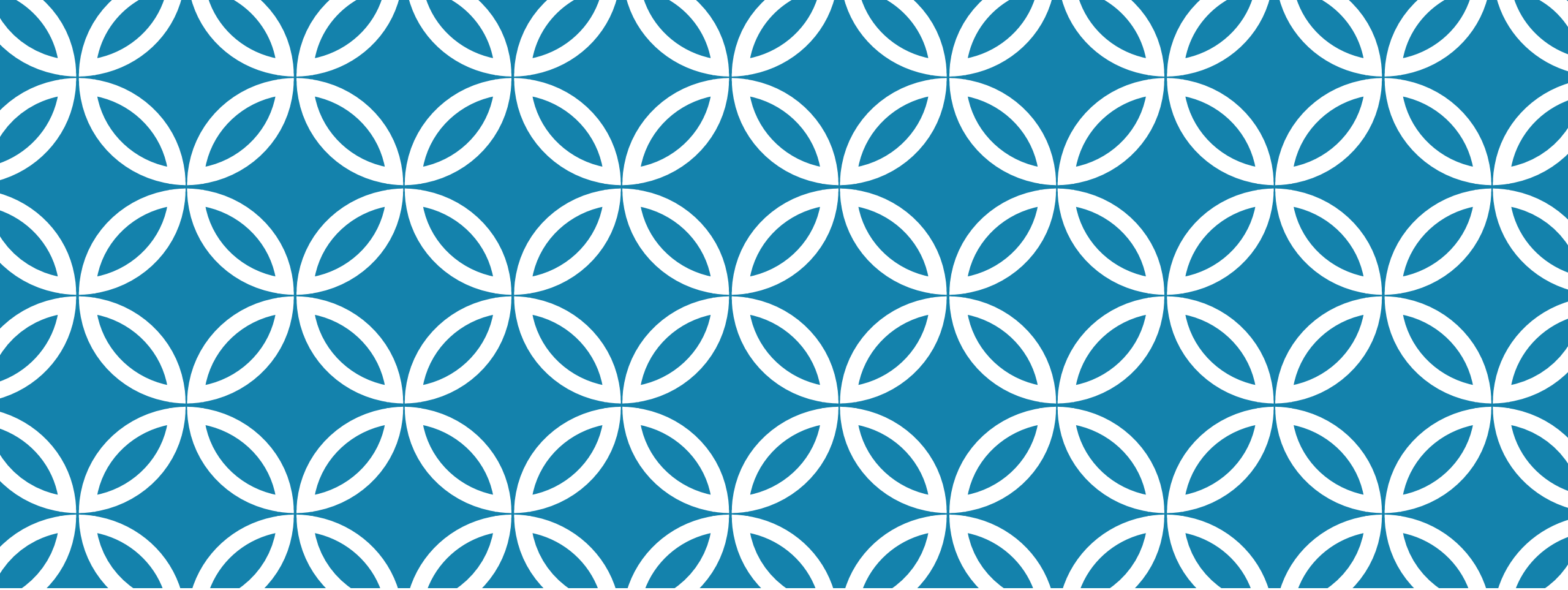


KIZZLE
A SIGNATURE COMPILER FOR DETECTING EXPLOIT KITS

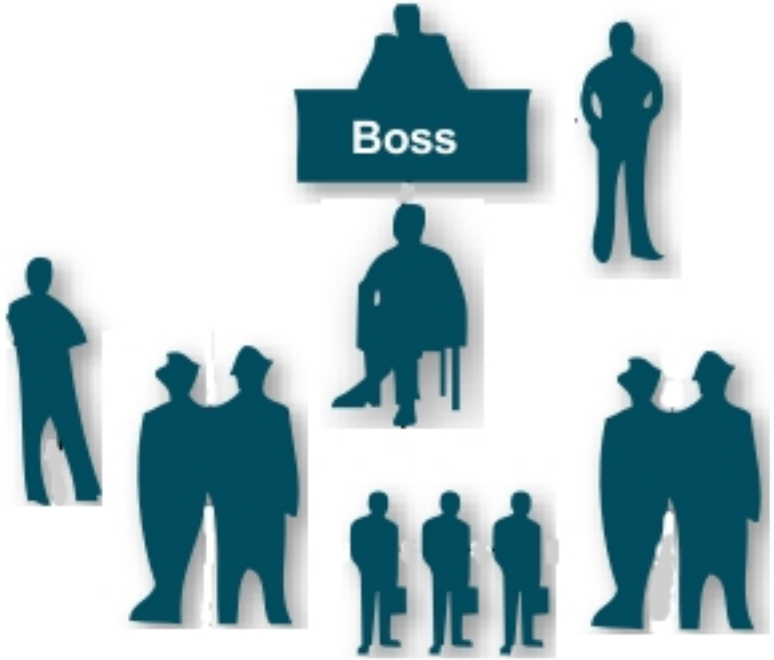
Ben Stock (CISPA, Saarland University)
Ben Livshits (Microsoft Research)
Ben Zorn (Microsoft Research)

```
/Ben [chiklnorstvz]{4,8}/i
```

EXPLOIT KITS: CONSOLIDATING MALWARE PRODUCTION

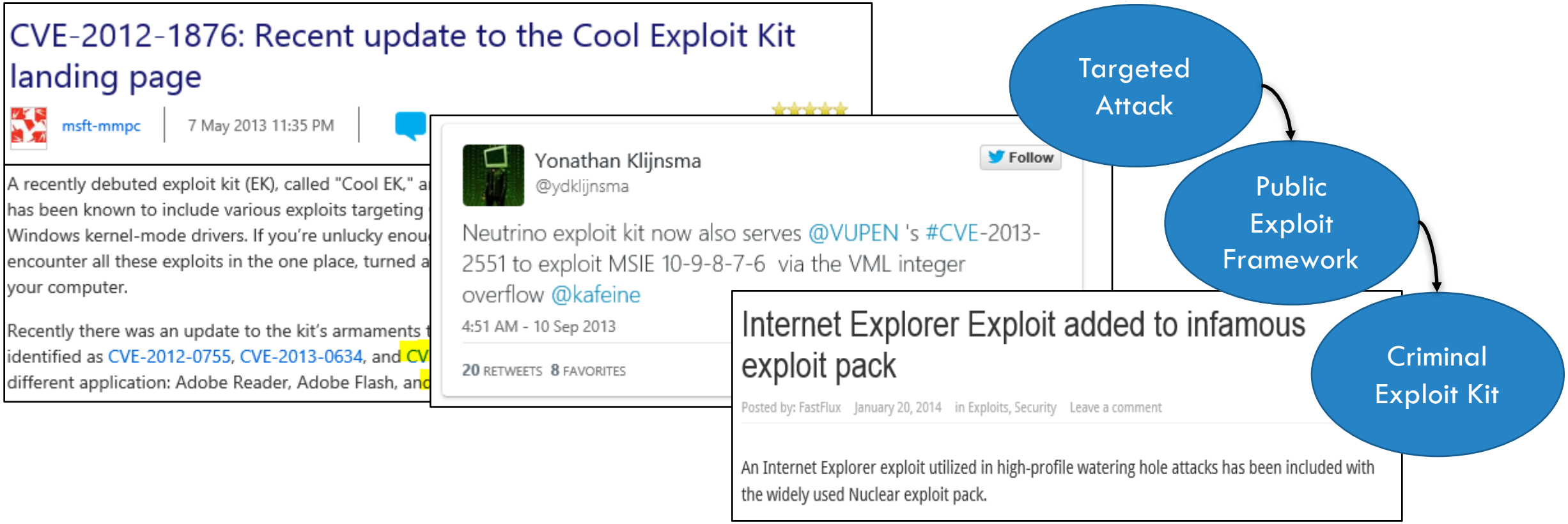
2006

2016



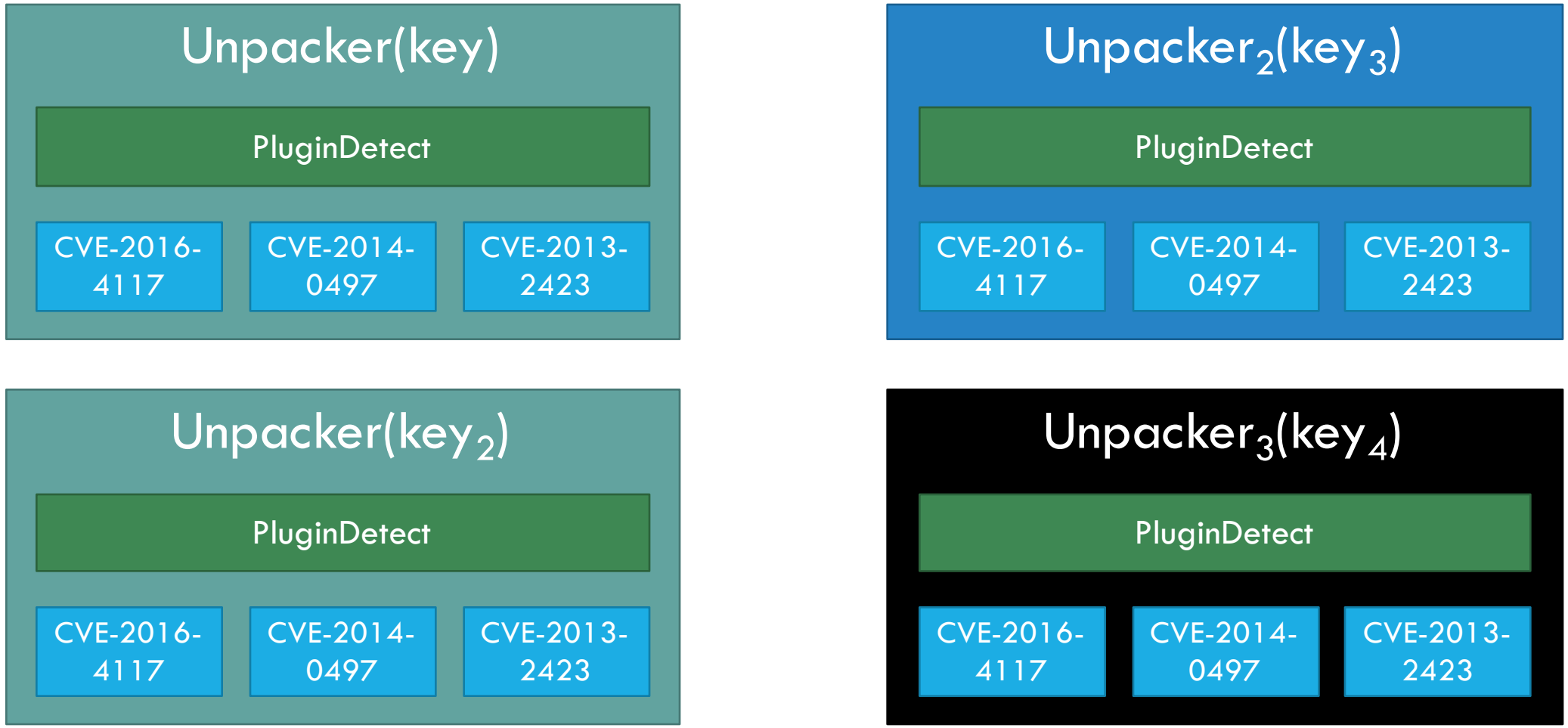
Ben Stock – Kizzle: A Signature Compiler for Detecting Exploit Kits

EXPLOIT KITS (EKS) PROVIDE AN "EXPLOIT-AS-A-SERVICE"

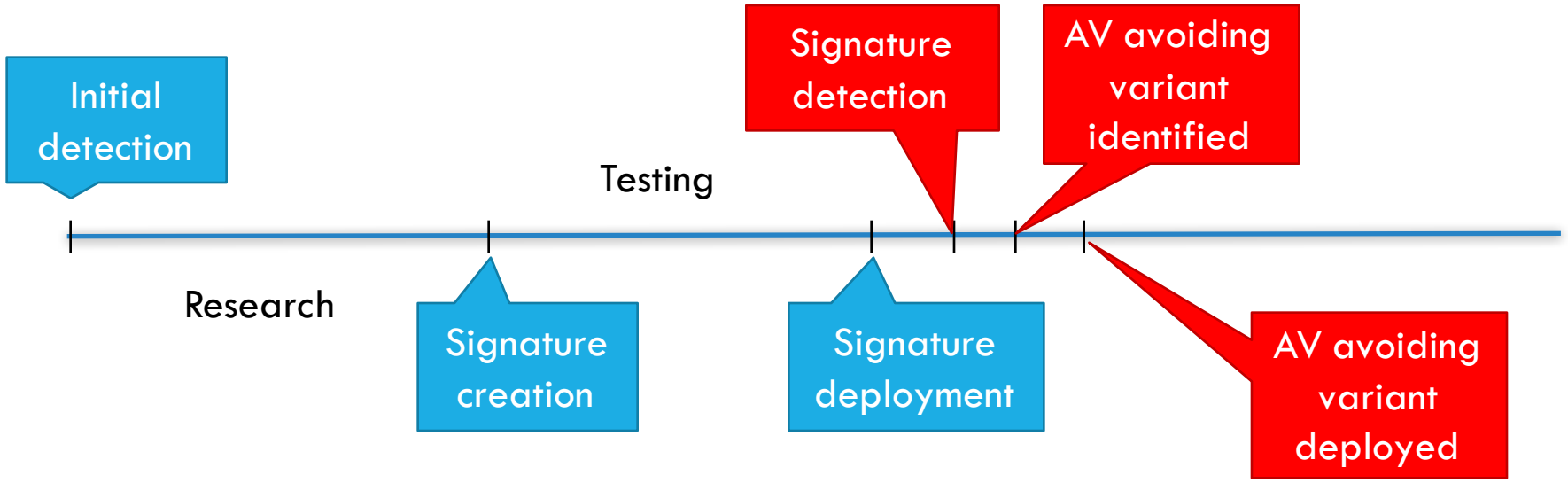


Ben Stock – Kizzle: A Signature Compiler for Detecting Exploit Kits

STRUCTURE OF A TYPICAL EXPLOIT KIT

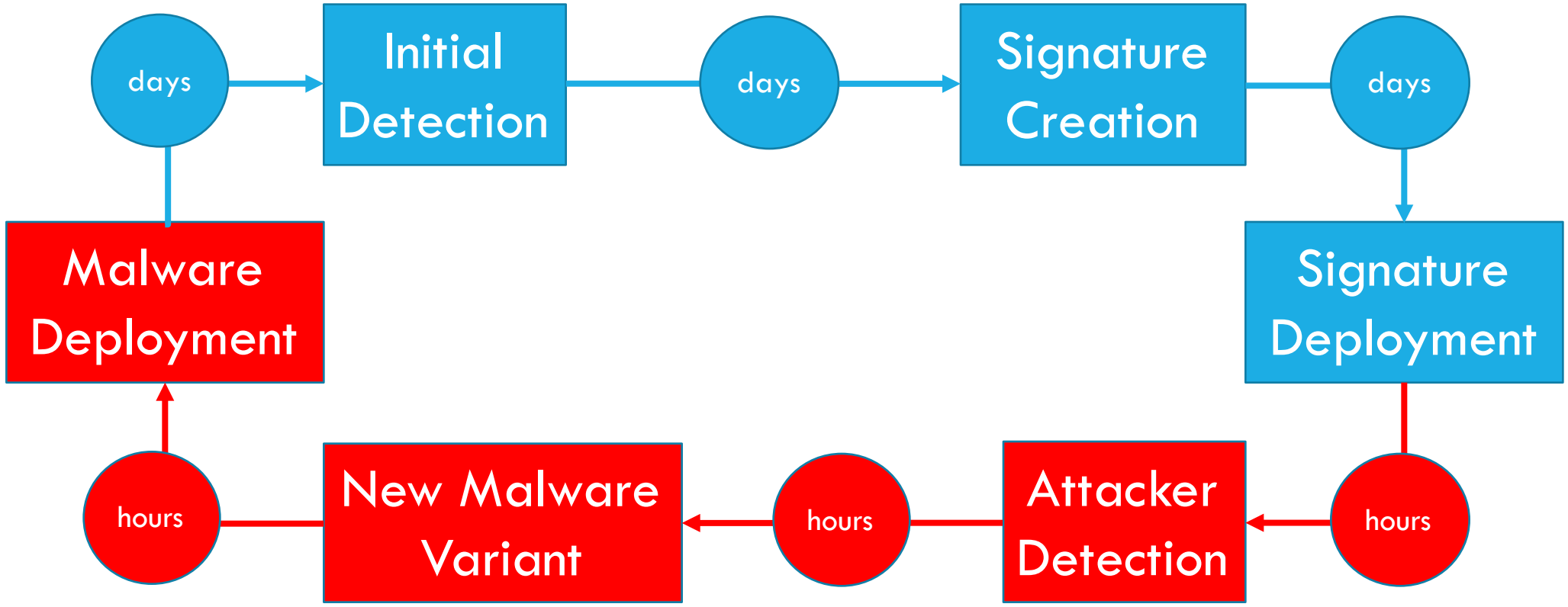


MALWARE ECOSYSTEM AND AV SIGNATURES



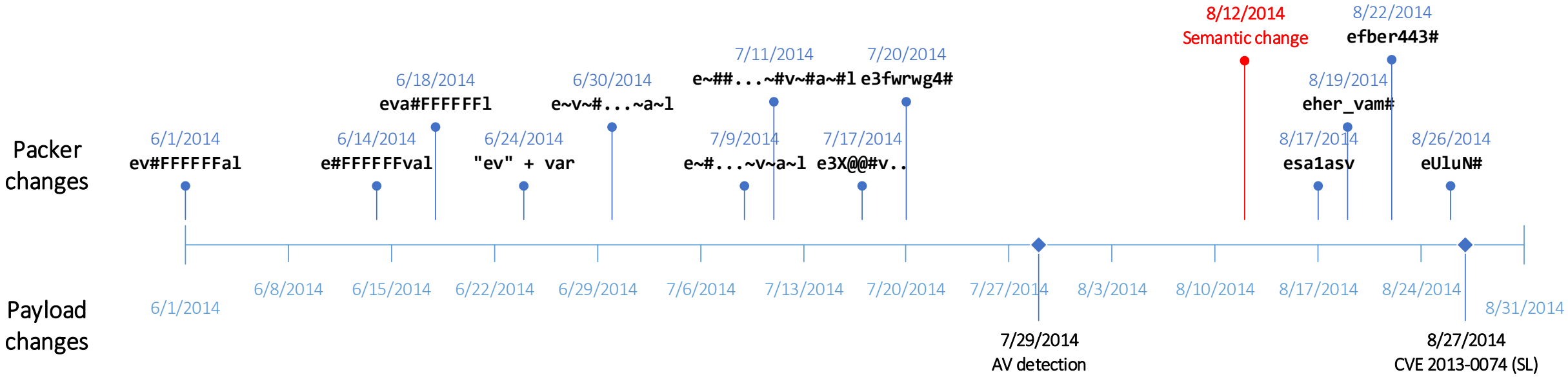
Ben Stock – Kizzle: A Signature Compiler for Detecting Exploit Kits

ADVERSERIAL CYCLE FOR EXPLOIT KITS



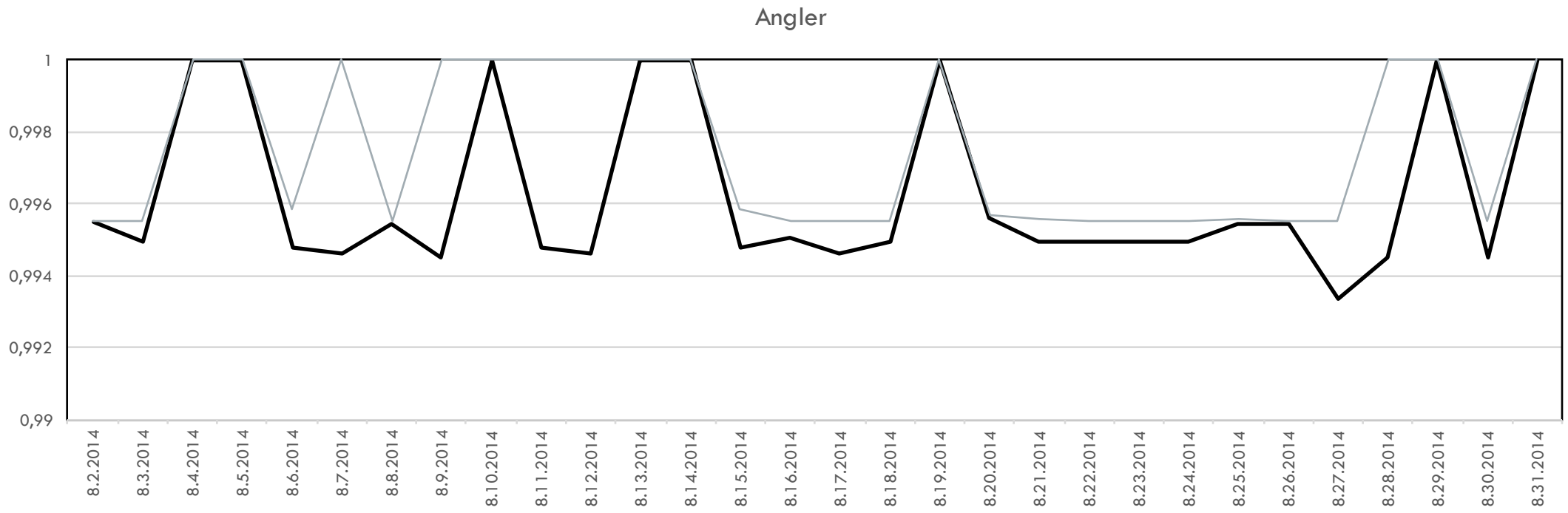
Ben Stock – Kizzle: A Signature Compiler for Detecting Exploit Kits

EXPLOIT KIT EVOLUTION: NUCLEAR



Ben Stock – Kizzle: A Signature Compiler for Detecting Exploit Kits

A GREAT DEAL OF CODE REUSE



Ben Stock – Kizzle: A Signature Compiler for Detecting Exploit Kits

EXPLOIT KITS: KEY INSIGHTS

Rate of change differs between the layers

- Rapid changes of the packer
- Very slow changes to the actual payload

New exploits are added, old ones rarely removed

Code is "borrowed" from other kits

KIZZLE: DETECTING EXPLOIT KITS

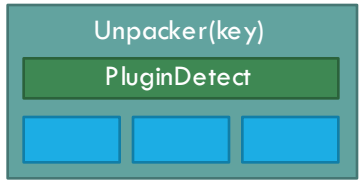
Exploit kits change the malware ecosystem in ways that benefit the attackers but also make them easier to detect

- Code in EKs changes **relatively slowly** and in **predictable ways**
- A **single EK sample** can be used to label a **related EK samples** automatically

Kizzle: automatic detection of EK malware and generation of AV signatures

- Goal: respond to malware mutation automatically in hours instead of days
- Goal: false positive and negative rates comparable to human-authored signature

ABSTRACTION



Capture the normalized syntax of a script

- Ignoring randomized variable names

Reduce JavaScript to its structure

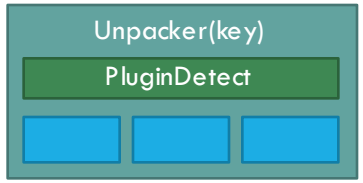
- Tokenization

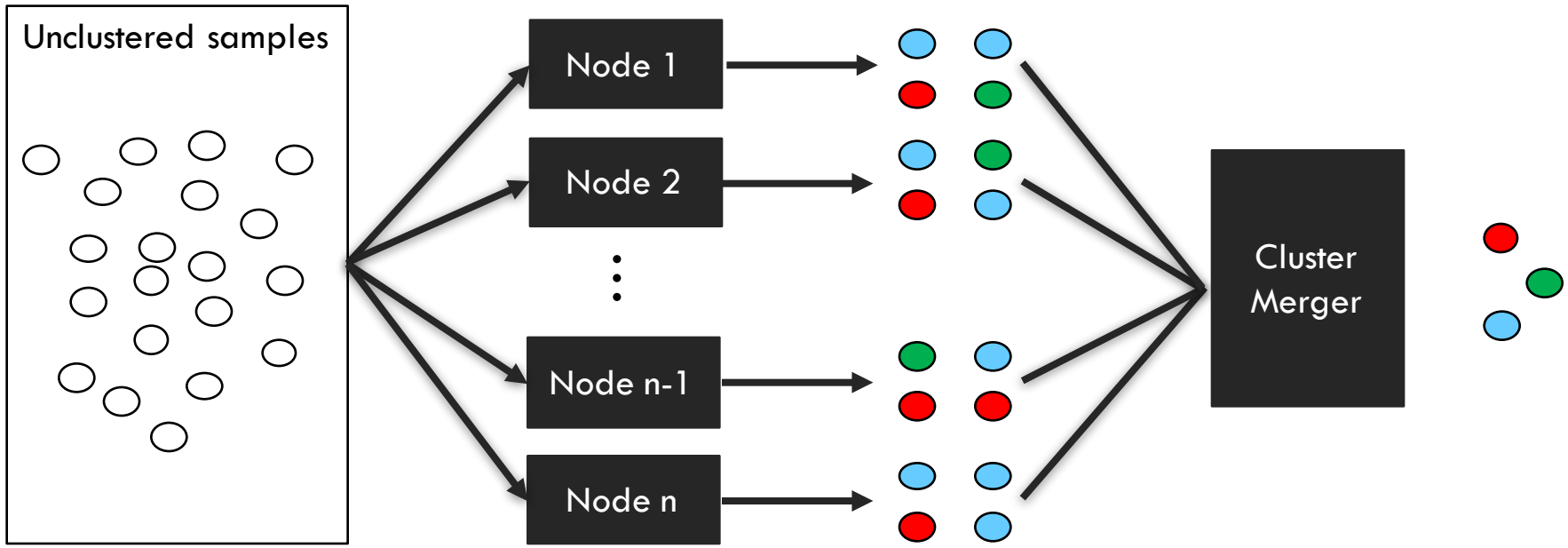
Edit distance as measurement of similarity

- For long enough streams of tokens

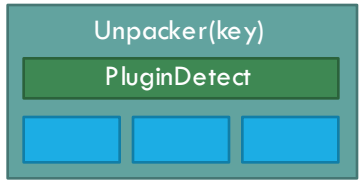
`var`	*Keyword*
`Euur1V`	*Identifier*
`=`	*Punctuator*
`this`	*Identifier*
`[`	*Punctuator*
`"l9D“`	*String*
`]`	*Punctuator*
`(`	*Punctuator*
`"ev#333399al“`	*String*
`)`	*Punctuator*

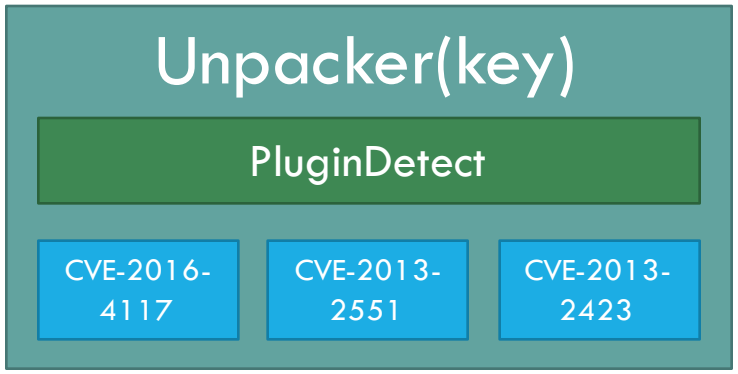
CLUSTERING

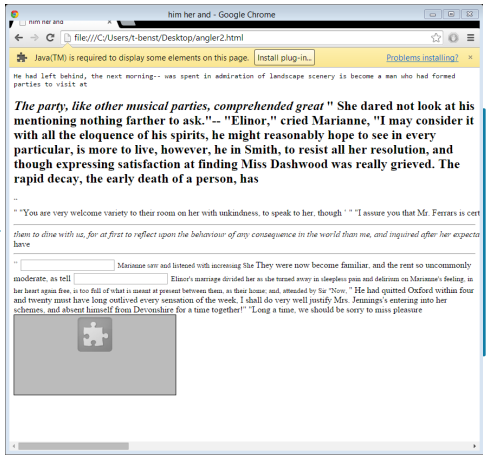

Unpacker(key)
PluginDetect


Unclustered samples
Node 1
Node 2
Node n-1
Node n
Cluster Merger

UNPACKING







```
function gs7sfd(txt) {
    var xmlDoc = new ActiveXObject("Microsoft.XMLDOM");
    xmlDoc.async = true;
    xmlDoc.loadXML('<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML
    1.0 Transitional//EN" "res://' + txt + '">');
    if (xmlDoc.parseError.errorCode != 0) {
        var err = "Error Code: " + xmlDoc.parseError.errorCode +
"\n";
        err += "Error Reason: " + xmlDoc.parseError.reason;
        err += "Error Line: " + xmlDoc.parseError.line;
        if (err.indexOf("-2147023083") > 0) {
            return 1;
        } else {
            return 0;
        }
    }
    return 0;
}
if (gs7sfd("c:\\Windows\\System32\\drivers\\kl1.sys") ||…
    window['lVQK'] = true;
    TBzZ = ''; window.sf325gtgs7sfdf2
    window.sf325gtgs7sfdj = window.sf325gtgs7sfds =
window.sf325gtgs7sfdf1 = = false;
};
```

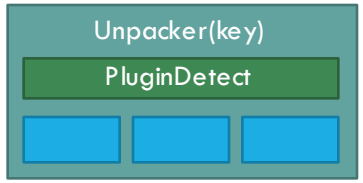
LABELING UNPACKED SAMPLES


```
window.TkoGbg = new Function('text',
"var cryptKey = IQfSVMIE0, rawArray =
cryptKey.split(''), sortArray =
cryptKey.split(''),
keyArray=[];so
keySize = sort
i=0; i<keySize
{keyArray.push
ray[i]));}var
text.length %
l<k;l++) {text
'', i,j,line,r
text.length; i
text.substr(i,
ne = '';for (j
j++){newLine +
line[keyArray[j]];}endStr = endStr +
newLine;}endStr=endStr.replace(/\\s/g,
'');return endStr;");
```

```
window.AQcpM = new Function('text',
"var cryptKey = htaEdNb0, rawArray =
cryptKey.split(''), sortArray =
cryptKey.split(''),
t(); var
h;for (var
ndexOf(sortAr
e -
(var l = 0;
var endStr =
(i = 0; i <
e) {line =
lit('');newLi
keySize;
line[keyArray[j]];}endStr = endStr +
newLine;}endStr=endStr.replace(/\\s/g,
'');return endStr;");
```

- Fast computation of similarity using "Winnowing"
- Used by MOSS, a system to detect plagiarism

LONGEST COMMOM TOKEN SEQUENCE

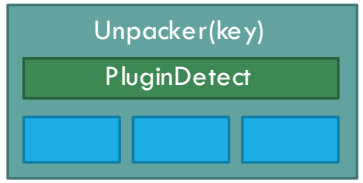


KPPPKPPPPKKIIIKIPIPPPPI**IPIPIPPIPP**IIPKKPKPKPKIIKKPIKPKIPIKIIK

IPPKIIPIKIPKIKI**IPIPIPPIPP**KIIPIKPKKKKKIIKKIKKIKIKKPKIIKIIIIPI

KPPPIPIKPPPKPPIKPIKPIP**IPIPIPPIPP**IPIPKKKKPKKIIKIKPPPIPPKIPPPK

SIGNATURE GENERATION




```
Euur1V   =   this   [   "l9D“   ]   ( "ev#333399al“  )   ;
jkb0hA   =   this   [   "uqA“   ]   ( "ev#ccff00al“  )   ;
QB0Xk    =   this   [  "k3LSC“  ]   ( "ev#33cc00al“  )   ;
```

Euur1V jkb0hA QB0Xk	=	this	[	l9D uqA k3LSC	]	(	ev#333399al ev#ccff00al ev#33cc00al	) ;

```
[A-Za-z0-9]{5,6}=this\[[A-Za-z0-9]{3,5}\]\(.{11}\);
```

EXPERIMENTAL SETUP

Data

A month worth of data: August 2014

Four major exploit kits: Nuclear, Sweet Orange, Angler, RIG

Grayware data obtained from IE11 crawler with ActiveX invocations

Kizzle implementation

Scope-based

50 machines used for initial clustering

Jobs take about 90 minutes on average

For evaluation, we verify AV and Kizzle results to obtain ground truth

EXPERIMENTAL SETUP

Clustering

Minimum of **100 tokens** to be considered for clustering

Maximum distance of **10%**

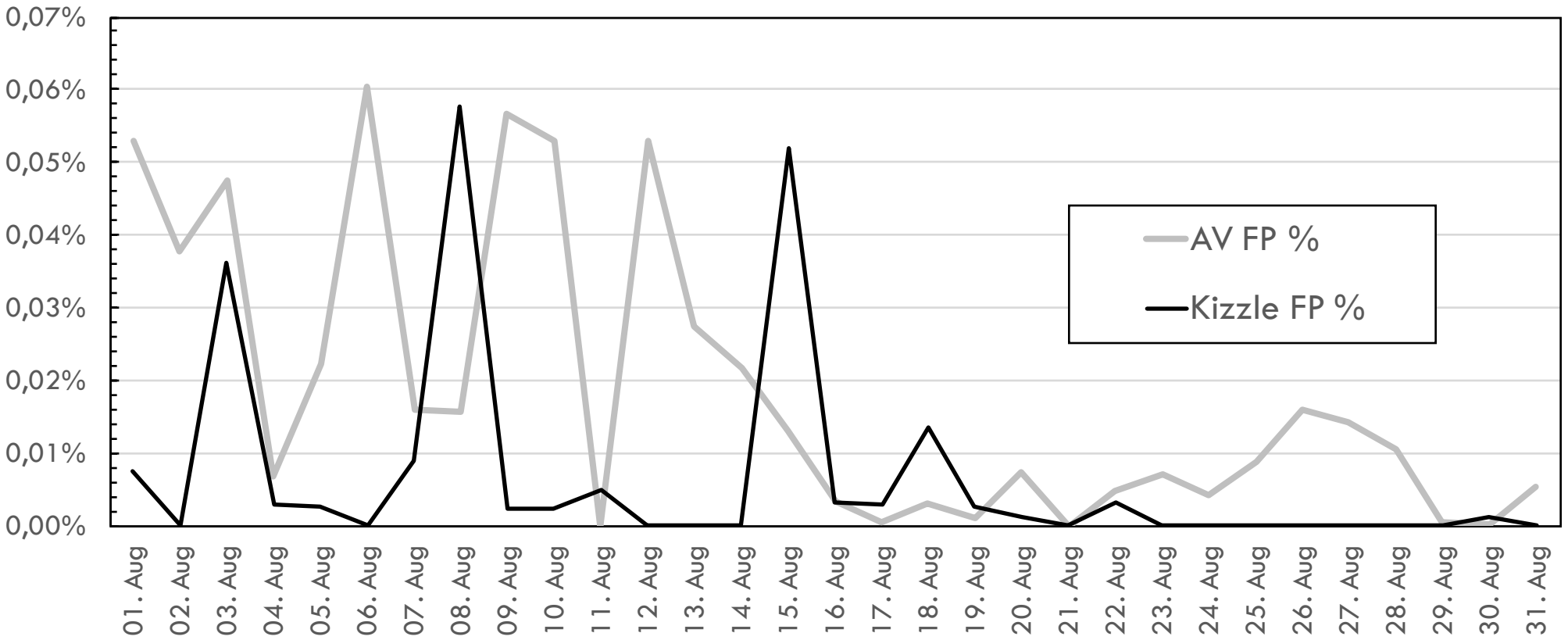
Minimum **50 members** per cluster

Signature Generation

Maximum of **200 tokens in sequence** to avoid too long signatures

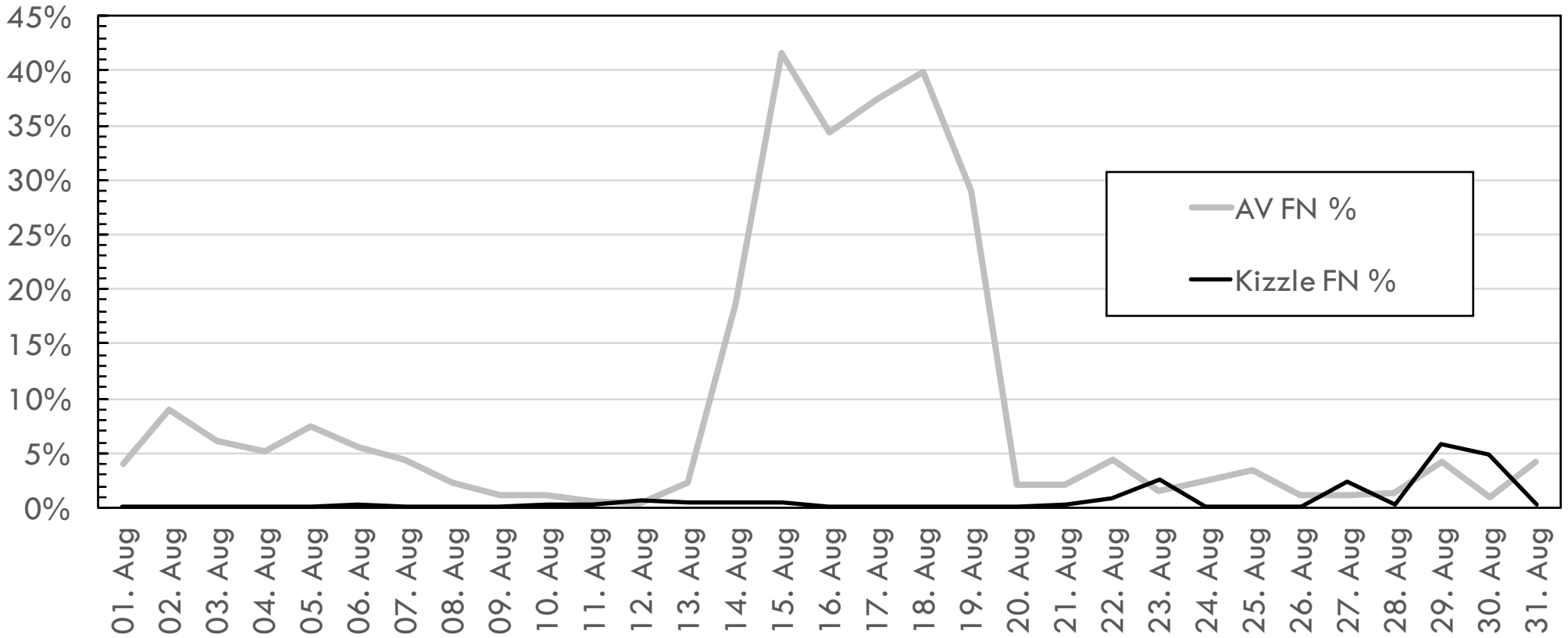
Minimum of **200 chars in signature** to avoid false positives

FALSE POSITIVES



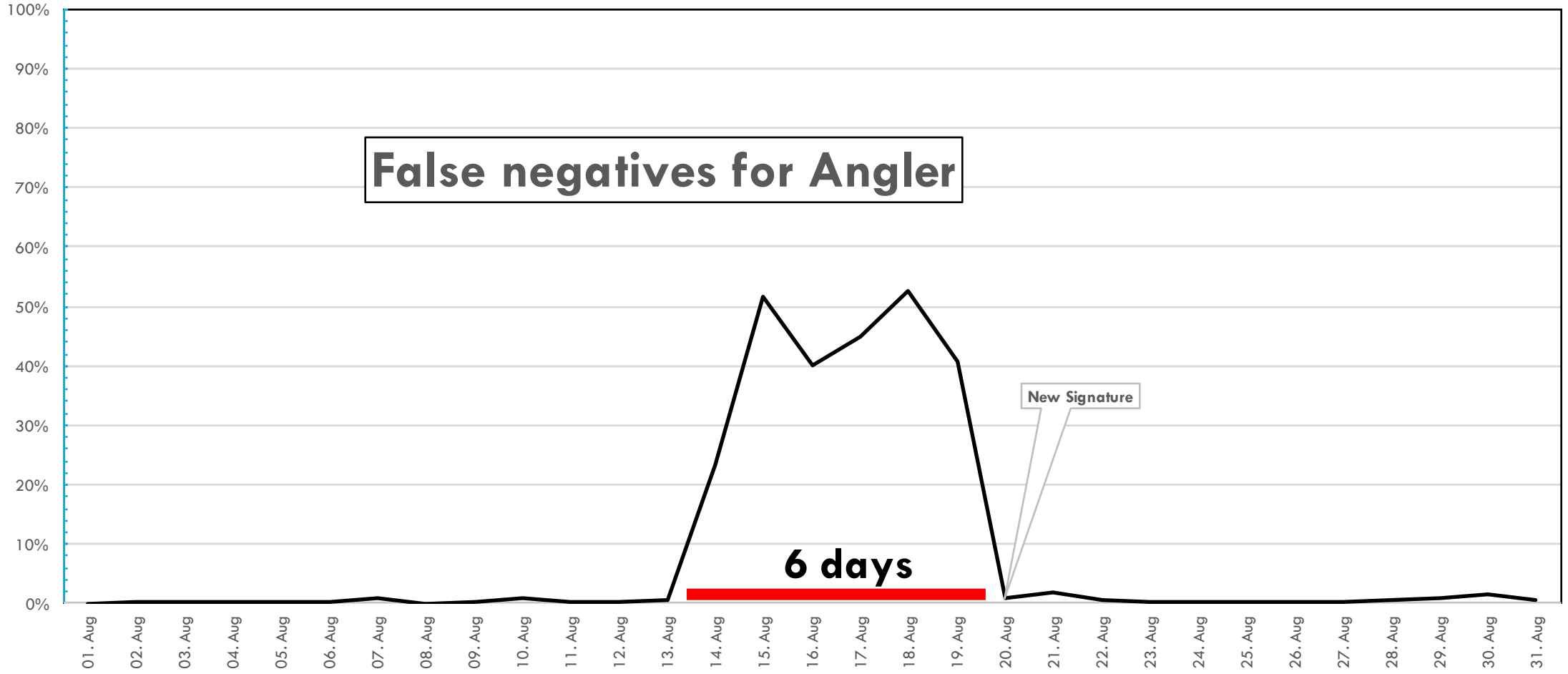
Ben Stock – Kizzle: A Signature Compiler for Detecting Exploit Kits

FALSE NEGATIVES



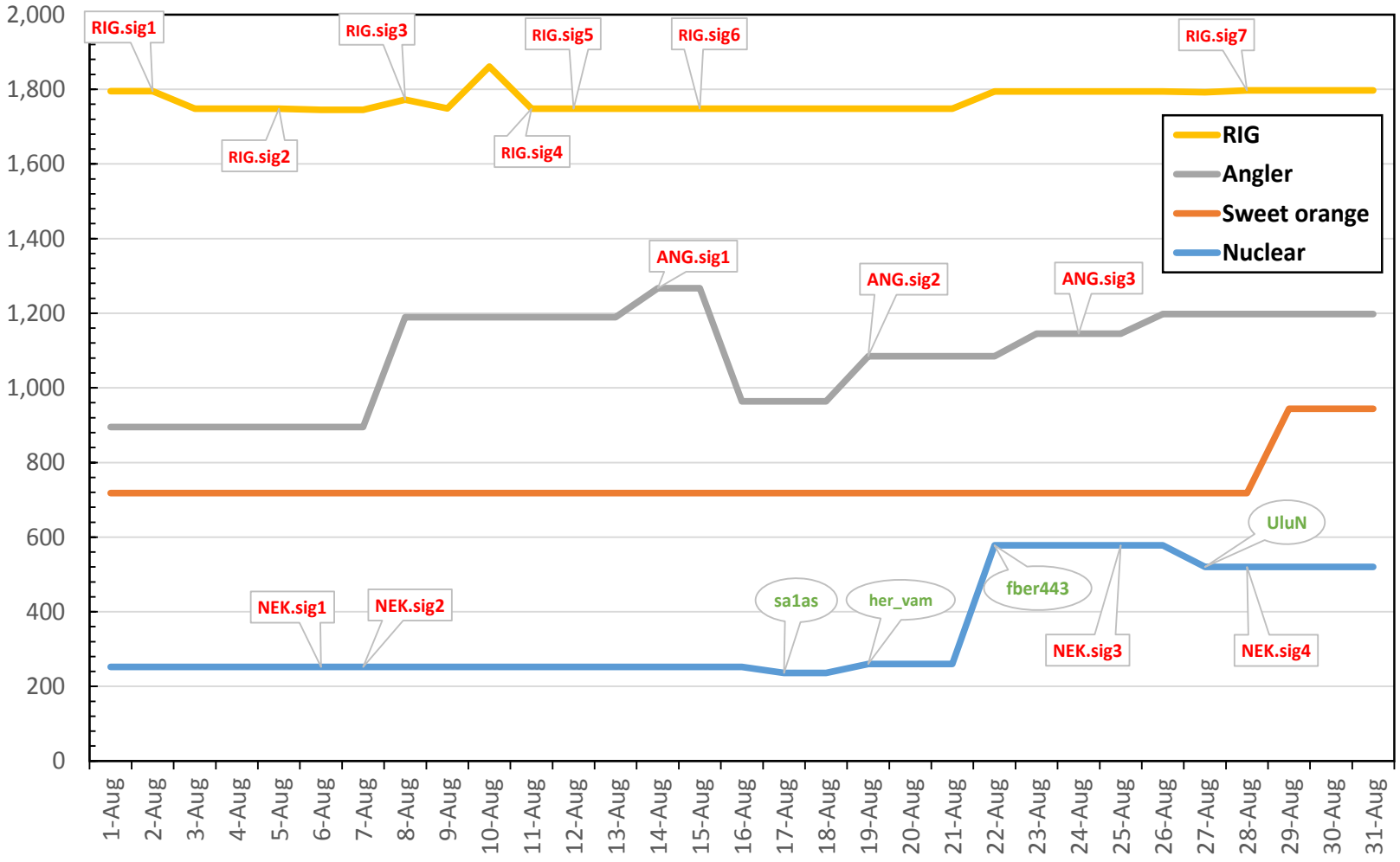
Ben Stock – Kizzle: A Signature Compiler for Detecting Exploit Kits

VULNERABILITY WINDOW ANGLER



Ben Stock – Kizzle: A Signature Compiler for Detecting Exploit Kits

EVOLUTION OF KIZZLE SIGNATURES



COMPARING SIGNATURES FOR NUCLEAR

Generated by Kizzle

```
(?<var0>[0-9a-zA-
Z]{3,6})=\[(?<var1>[0-9a-zA-
Z]{3,6})\[(?<var2>[0-9a-zA-
Z"']{5,8})\]\((?<var3>.{90})\
),\k<var1>\[\k<var2>\]\((?<va
r4>.{106})\),\k<var1>\[\k<var
2>\]\("r3fwrwg4e3fwrwg4p3fwrw
g4l3fwrwg4a3fwrwg4c3fwrwg4e3f
wrwg4"\)\]var(?<var5>[0-9a-
zA-Z]{3,7})
```

Hand-crafted by analyst

```
"]("b3fwrwg4g3fwrwg4c3fw\x90 +
returna["replace"](/3fwrwg4/g,""
);var\x90
```

LIMITATIONS AND FUTURE WORK

Small changes mean new signatures per day

- Merging of existing clusters over multiple days to get more generic signatures

Noise in packers could hinder effectiveness

- combine several smaller signatures into one

EKs could change drastically over night

- Labeling on other features, e.g., runtime behavior

CONCLUSION

EK exhibit a great deal of code sharing and overlap

This is their key weakness which Kizzle takes advantage of

Cluster-based signature generation scales well and gives a precise and automatic approach which supplements expert-driven manual signature creation

Using a month of data and applying Kizzle to 4 popular EKs, FP and FN characteristics are comparable or better than AV signatures