

RSA® Conference 2019

San Francisco | March 4–8 | Moscone Center



BETTER.

SESSION ID: SEM-M03E

Lessons Learned Running High Stakes Cybercriminal Arms Races

Michael J.J. Tiffany

President, Co-Founder

Tamer Hassan

CTO, Co-Founder

#RSAC

Ad fraud pays astonishing well

It makes the most money with the least risk:

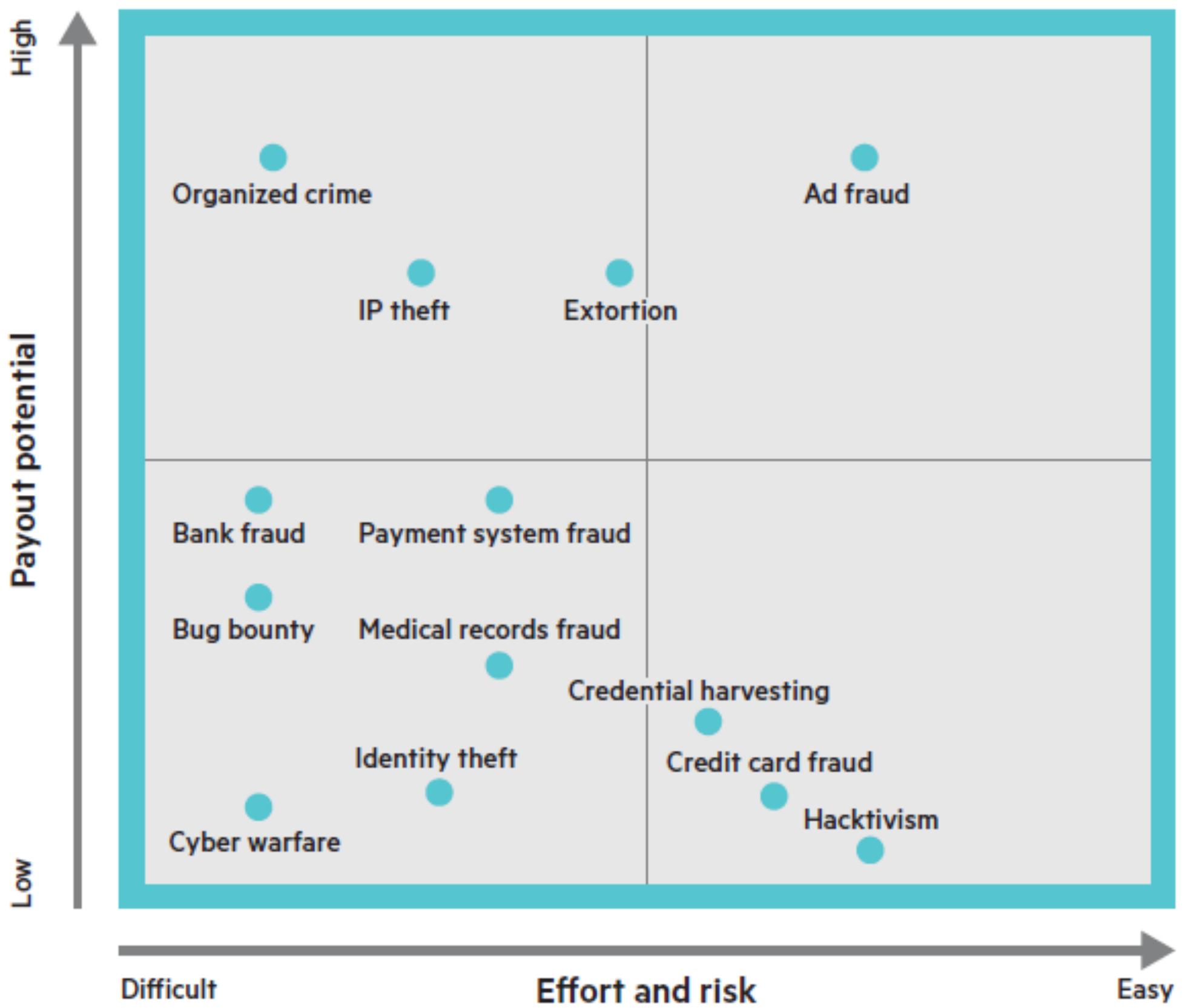


Figure 1: Attractiveness of hacking based on financial gain and effort
Source: HP Enterprise Solutions. The Business of Hacking, May 2016

So botnet operators doing ad fraud are elite

- If you make millions of dollars, you can afford a multimillion dollar counterintelligence budget to reverse-engineer ad fraud detection
- As a result, the battlefield is strewn with dead detection techniques that worked for a time, but then were defeated

Most defense doesn't solve the problem

- It just changes who the victims are
- The Club doesn't decrease the number of cars stolen
- It just changes which cars are chosen
- Most defenders are only paid to do that

What do you have to do to take out a whole category of cybercrime?

- Something different happens when you aim for categorical defense: the bad guys play back at you
- Remember the words of Sun Tsu: if you don't leave an avenue for your enemy to retreat, he will fight to the death
- That's what we've been doing with ad fraud

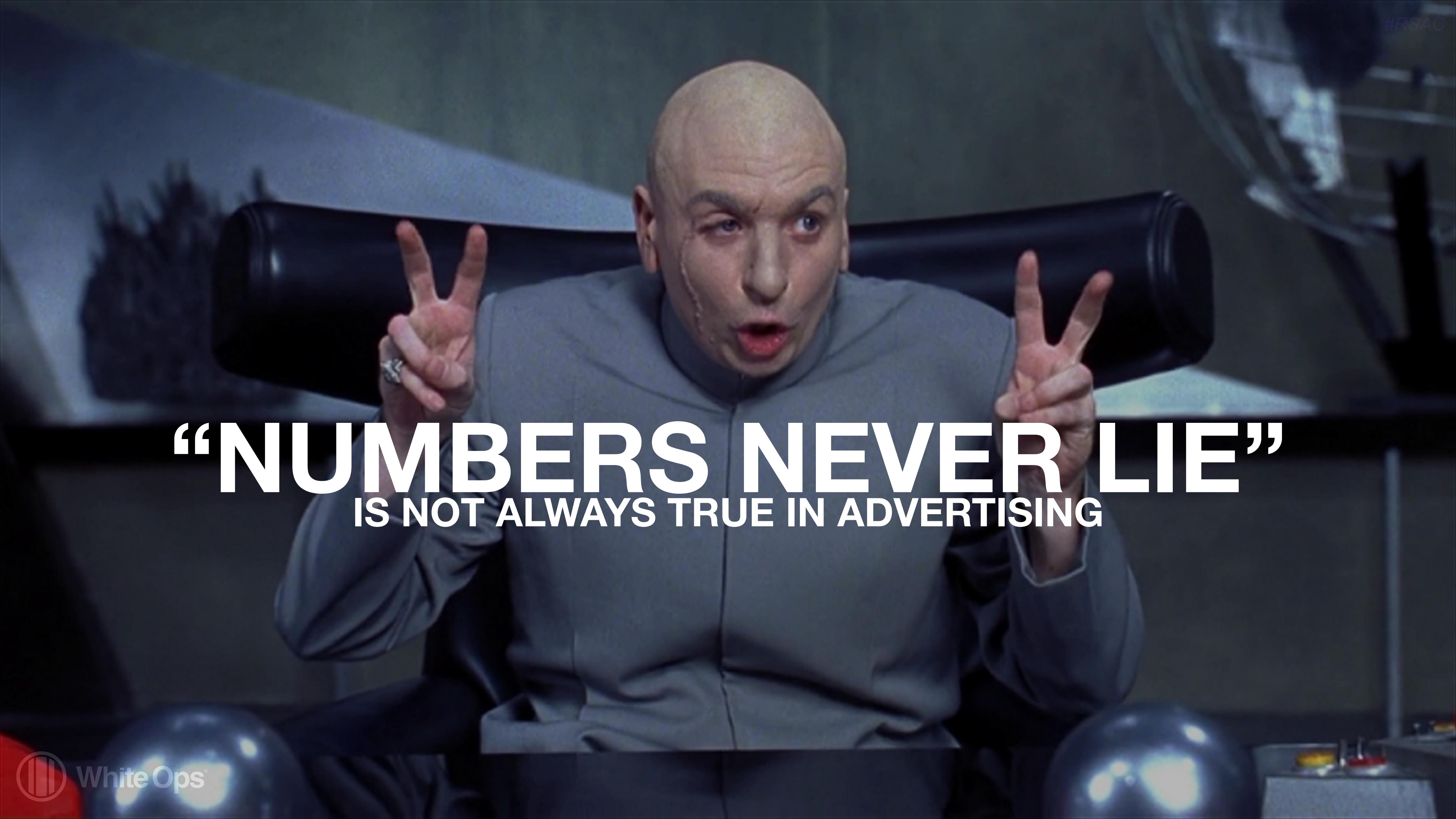
The Arms Race is On

- In an arms race, victory isn't about what you bring to the fight today.
- It's what you can bring to the fight tomorrow and 1,000 tomorrows from now
- Against an enemy who's saying
 - the
 - exact
 - same
 - thing.

We're going to share with you our techniques for winning arms races

- Against adversaries who:
 - are at least as smart as us
 - are quite possibly richer than us
 - have great incentives to win
- So we'll start by telling you more about what we're up against

But wait:
Almost all digital advertising is targeted.
And bots don't buy anything.
So why doesn't this problem solve itself?

A bald man in a grey suit is sitting in a dark, futuristic-looking vehicle or cockpit. He is looking directly at the camera with a surprised expression, his mouth slightly open. He is making a peace sign with both hands, one on each side of his head. The background is dark and blurred, suggesting a high-speed vehicle.

“NUMBERS NEVER LIE”

IS NOT ALWAYS TRUE IN ADVERTISING



A bot on Grandma's computer looks as real as she does



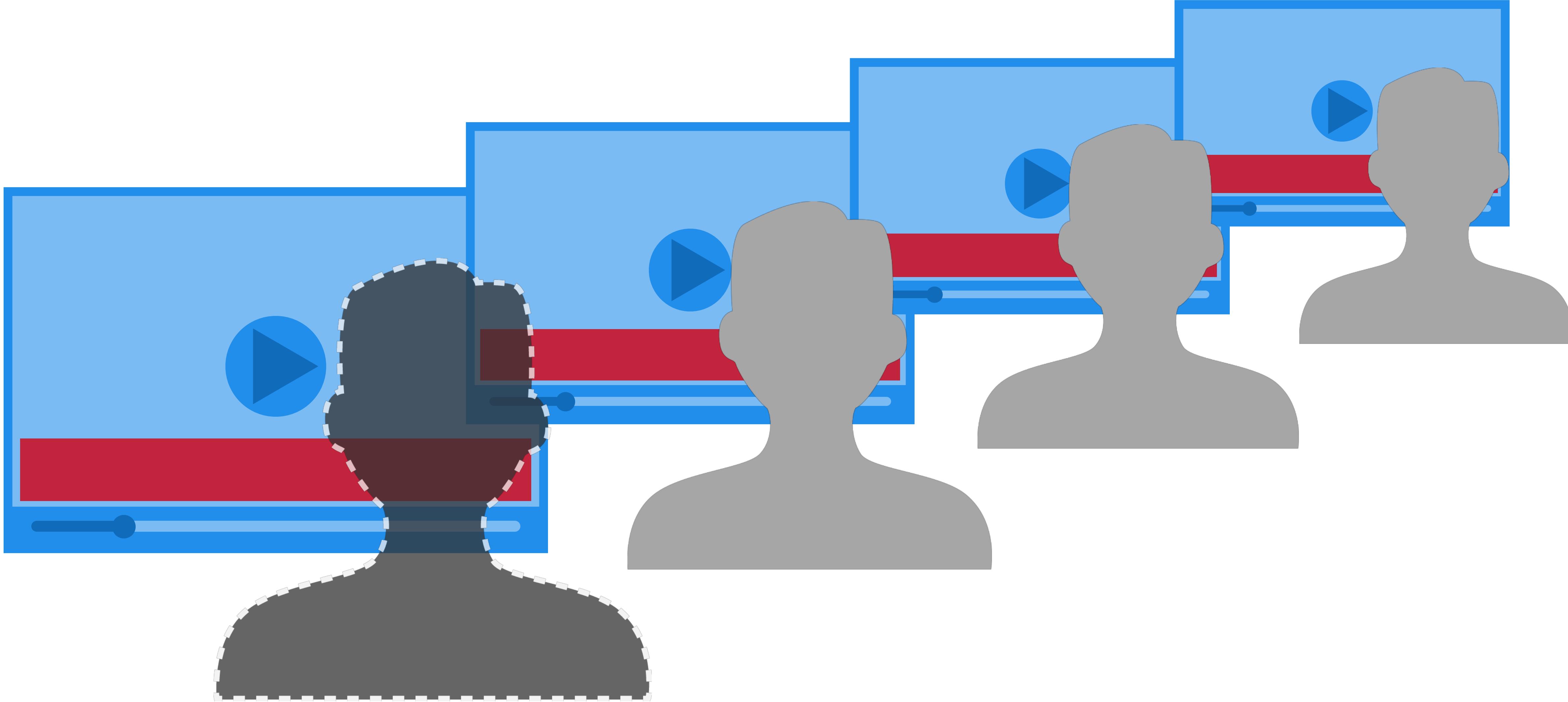
If Grandma is:

logged into a social network, checking her email, buying things on the web...

and there is a **bot** on her computer...

the **bot** looks like it's doing those things, too.

With bots on millions of infected computers “watching” ads in the background...



Bots can
inflate the
size of any
audience



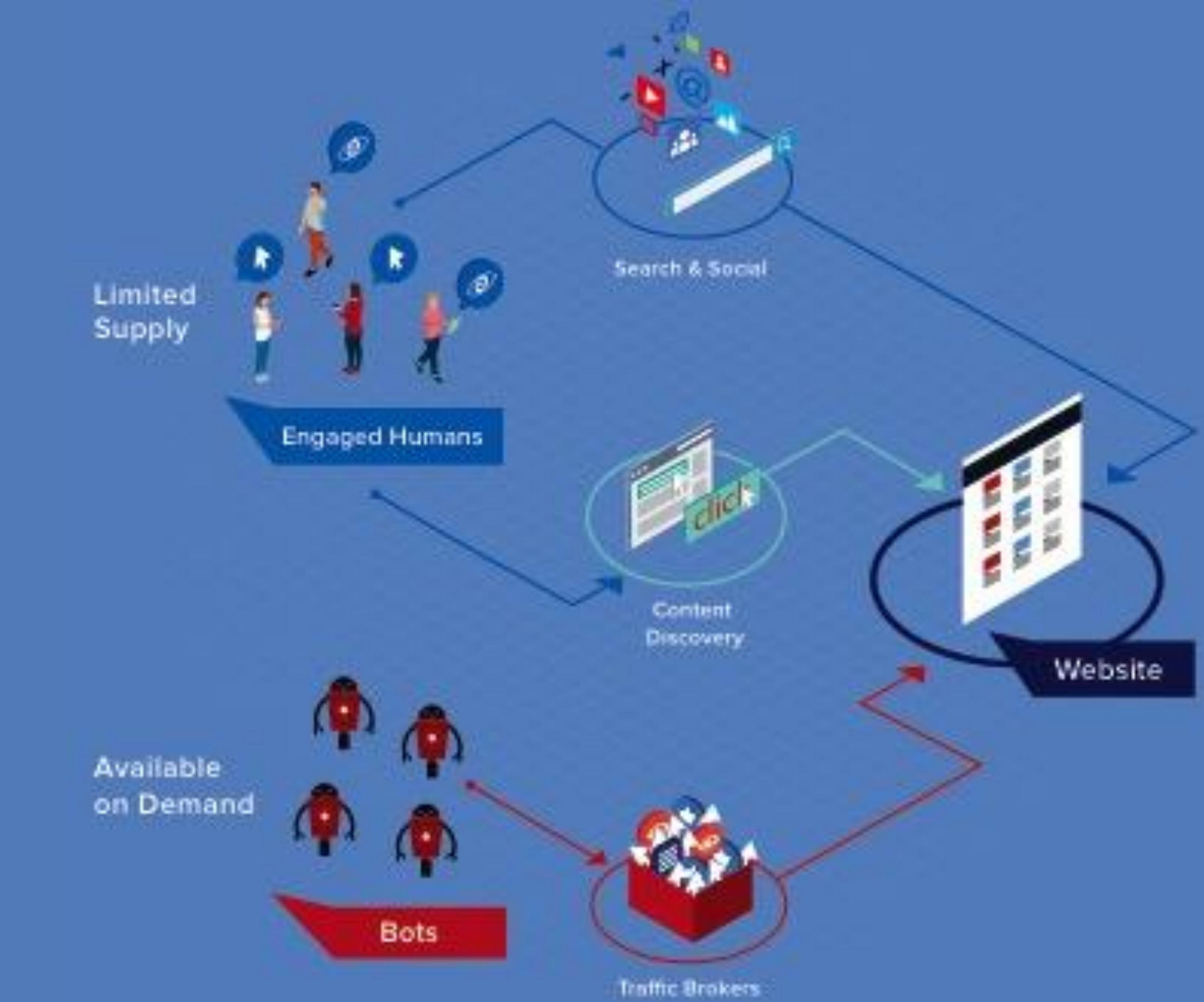
Need to serve more ads?
The market price for bot
traffic that passes as human
is \$0.01 or \$0.02 per visit

A 50,000 node botnet is big
enough to pull in \$150,000/mo.

Botnets sell their traffic to an enabling layer of CPC traffic brokers, who use cover stories about placing affiliate links and pay-per-click text ads for their customers' websites to drive traffic to them

Traffic acquisition programs provide visitors on demand

Publishers paying handsomely for legitimate search traffic are competing against publishers paying much less for bot traffic, and the tools used by most marketers cannot tell the difference.



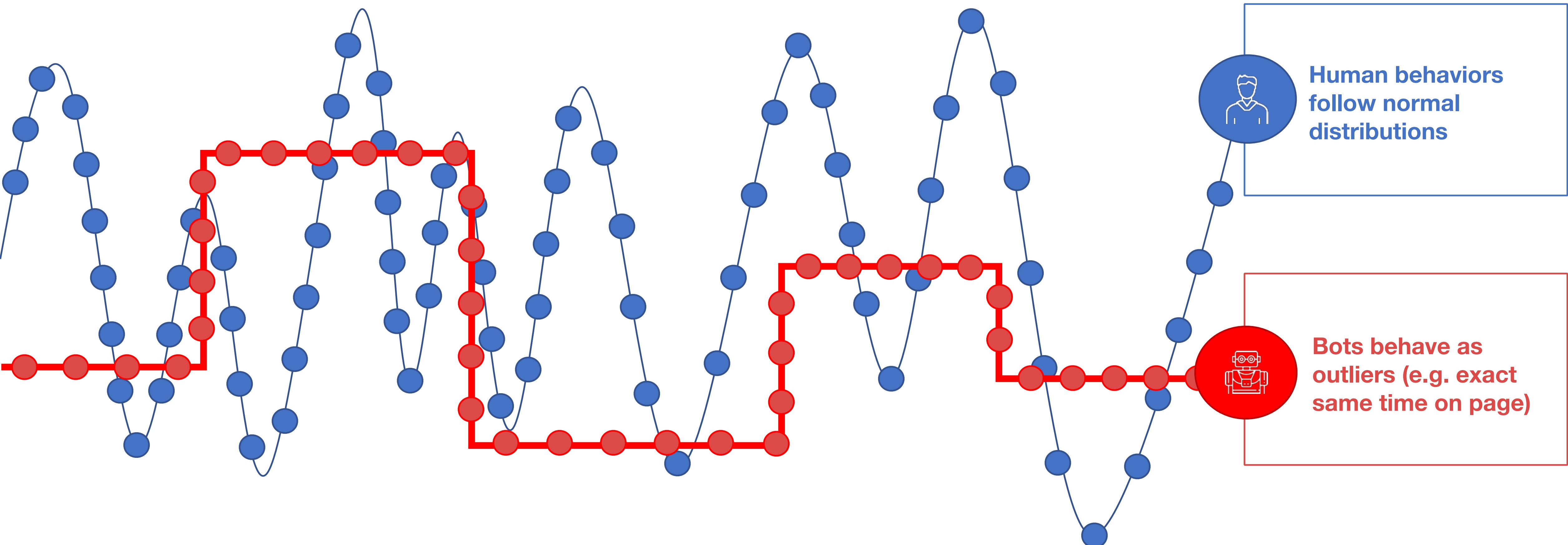
The result on the market:

**Infinite reach seems possible against
even the most niche audiences**

**The price of advertising stays the same
or even goes down during peak
demand**

In a market with the
biggest of Big Data, how
does this happen?

Past bot defense relied on the idea of finding anomalous behaviors: patterns that simply look “non-human”



The challenge is that bots have evolved and can now perfectly imitate human behaviors.

Yesterday's Bots

Lived in data centers

→ 76% live on residential machines

Lacked cookies, etc.

→ Have cookies, browser history

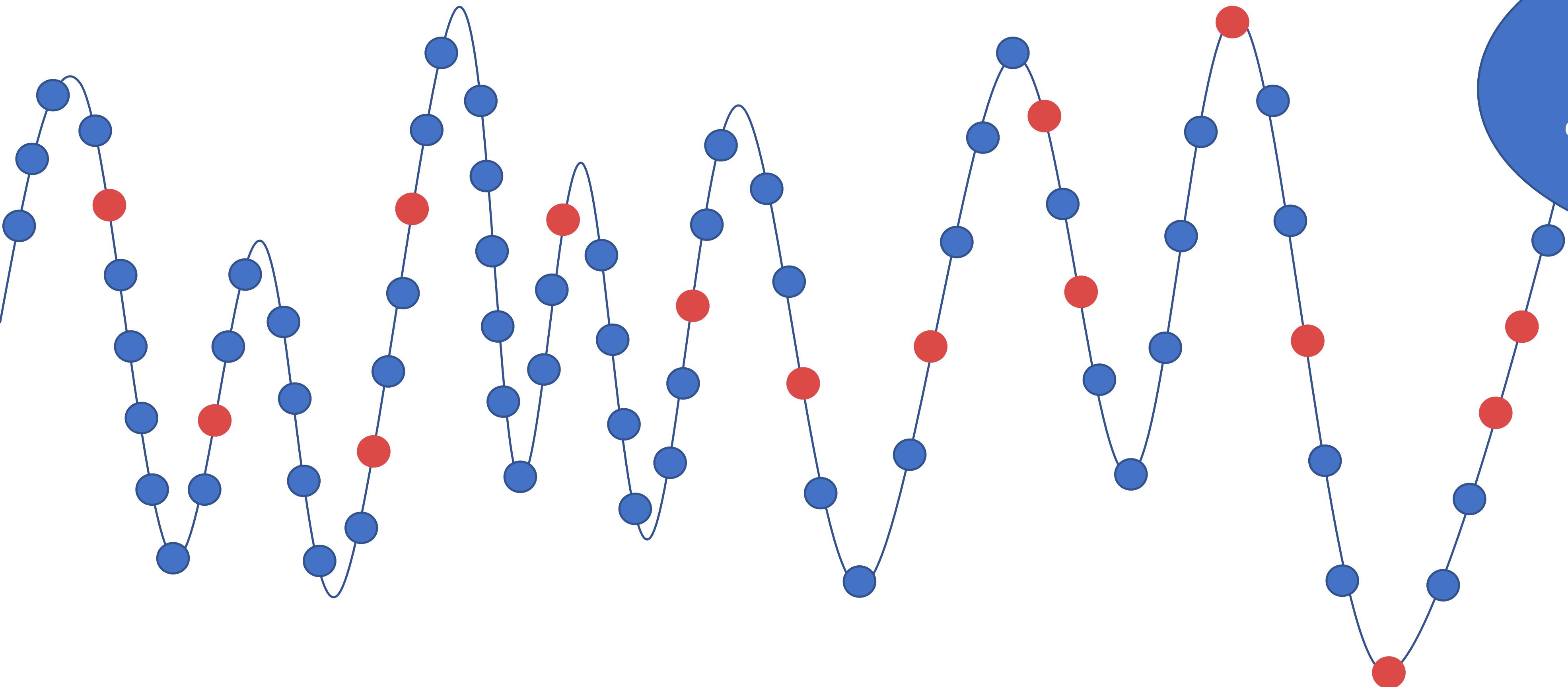
Behaved “like bots”

→ Behave like humans

Never bought things

→ Get credit for what their pet humans buy

So now instead of being clear outliers, the bots simply blend right in with the humans *and get credit for their conversions*



If bots can now impersonate human behaviors, how effective can looking for non-human behaviors possibly be?

Just how sophisticated are we talking?

All the fraud models

- Be a “Content Creator”, hire “traffic”, get paid by publisher
- Be a “Publisher”, hire “traffic”, get paid by SSPs
 - “cashout sites” or “ghost sites”
- Be an “SSP”, create “inventory”, get paid by DSPs
 - Ad injection (malware, evil proxies/vpns, dnschanger, etc)
 - Vertically integrated bots that create “traffic” and “inventory” (Methbot)
- Have bots, sell “traffic”
 - Own end users
 - Run a “bot farm”
- Have shady site, disguising source of traffic (“traffic laundering”)
- Have site, push affiliate cookies (“cookie stuffing”)
- Have site, run lots of invisible ads (“ad stacking”)

Fraud detection forms the selection pressure for the ecosystem. Let's look at the past several generations of evolution.

Bot Designs: curl/wget

- A very small shell script
- Figure out the URL that triggers a billing event, hit it
- Great for your IoT botnet
- Very easy to catch by anyone who cares

Bot Designs: Scripts

- Basic web scraper/crawler type code
- Usually written in something like python, node, perl, php, ruby, etc
- Can parse HTML
- Doesn't execute JavaScript
- Fairly easy to catch by anyone who cares

Bot Designs: Embedded

- A rendering engine is embedded in another application
- Internet Explorer, Chromium, and Webkit all have supported embedding tools
 - IE WebBrowser control, MSHTML
 - Chromium Embedded Framework
 - WebKit
 - Official support for embedding Gecko was dropped in 2011. Can still be done.
- Usually intended for rendering trusted content
- May have security controls disabled
- Range widely in detection difficulty

Bot Designs: Off-the-shelf headless browsers

- Repurposed tools designed for scraping or QA
- Runs without displaying anything
- PhantomJS, SlimerJS, Zombie.js, HtmlUnit, etc.
- Headless Chrome
- Not usually suitable for compromised end user systems
- Large payloads, requiring frequent updates
- Unmodified, detectable with a little effort
- Minor modifications for stealth make detection tricky

Bot Designs: Off-the-shelf automation tools

- Repurposed tools designed for scraping or QA
- Hooks into a real web browser and automates it
- Selenium, Webdriver, and their various wrappers
 - There's currently a draft W3C spec for webdriver, supposed to set navigator.webdriver = true
- Not usually suitable for compromised end user systems
- Large payloads, requiring frequent updates
- Can be difficult to detect

Bot Designs: System Emulators

- Primarily done for bots wanting to run mobile traffic
- Usually combined with off-the-shelf automation tools
- Also done to run “bot farms”
- Tricky to detect

Bot Designs: Custom Browser

- Implement enough of a browser to make verification vendors happy
- Large development effort
- High maintenance
- Deep control of behaviour
- Didn't expect anyone to actually do this, but in Methbot, we found one

Bot Designs: Hybrid

- Browser farm in the cloud with SOCKS through infected residential machines
- An improvement on the Methbot model:
 - Enabling programmatic spoofing
 - Real residential IPs
- No suspicious CPU load on the infected endpoints to spin up the fans and attract attention
- Dramatically simpler management and update deployment
- Hides infrastructure from researchers

Just how good can custom browsers
get?

Methbot, a custom browser with modular monkeypatching

- At peak, 300M video ad impressions per day, for millions of dollars
- Hundreds of thousands of IPs falsely registered as US ISPs
 - No, not BGP hijacks, large block allocations and small leased blocks
- Custom HTTP library (buggy)
- DOM support via Cheerio
- CSS support (library unknown)
- Fully custom implementations of many browser APIs
- Flash support via custom NPAPI implementation and Fresh Player
- NodeJS runtime
- A “bot farm” running on dedicated servers
- Extensive fraud detection countermeasures

HOWTO: Live remote malware inspection

```
var ary = Object.keys(window), dumpf, dumpt, dumpc;  
// grab a random object from the global namespace  
var rndObj = window[ary[(Math.random()*ary.length)|  
0]];  
// wrap a hopefully untampered toString function  
var str = function(o){return function()  
{}.toString.apply(o)};  
// try to dump some code  
try{ dumpf = str(rndObj) }catch(e){}  
try{ dumpt = str(rndObj.toString) }catch(e){}  
try{ dumpc = str(rndObj.constructor) }catch(e){}
```

And their response in the arms race:

```
text = text.split('function()\n{} .toString().apply()\n            .join('window.__MethFakedFuncToString\n(');\n\ntext = text.split('function()\n{} .toString().apply()\n            .join('window.__MethFakedFuncToString\n(');\n\ntext = text.split(' {} .toString().apply()\n            .join('window.__MethFakedToString(');
```

Methbot - __MethFakedFuncToString

```
__MethFakedFuncToString = function(el){  
    try {  
        if (el.hasOwnProperty('toString'))  
            return el.toString()  
    } catch (e) {}  
    var t = null;  
    t = function() {}.toString.apply(el)  
    return t;  
}
```

Other JavaScript Dumping Countermeasures

```
function toString() {
    // An if-else chain is used here because a "switch" block or an Object lookup
    // would coerce these functions into strings.
    if (this === _functionToStringShim) {
        var target = _functionToStringOrig;
    } else if (this === _alertShim) {
        target = _alertOrig;
    } else if (this === _confirmShim) {
        target = _confirmOrig;
    /* This code has been modified from its original version. It has been formatted to fit this slide. */
    } else if (this == _getCurrentPositionShim) {
        target = _getCurrentPositionOrig;
    } else if (this === _onmessageDelegate && _onmessageFormatted != null) {
        return _onmessageFormatted;
    } else {
        target = this;
    }
    return sandbox('Function', 'toString')(target);
}
```

Escalation: Very specific countermeasures

```
// return loadLocalFile(this.link, this.callback,  
// '../for_whiteops/load.src.4.16.6.js')  
//}  
// wo flash  
if (this.link.indexOf('viz11.swf') !== -1) {  
var res = {  
  url: this.link, statusCode: 200, status: '200 OK',  
  rawHeaders:'HTTP/1.1 200 OK\nServer: nginx/1.4.6 (Ubuntu)\n',  
  headers: {}, $:cheerio.load(''), body:new Buffer('')  
};  
return this.callback(false, res)  
}
```

Now let's talk IP address diversity

Methbot's approach: IP Registration Forgery

inetnum:	196.62.0.0 - 196.62.31.255	person:	IP Admin
netname:	COMCAST-CABLE	address:	IP Admin
descr:	Comcast Cable Communications, Inc	phone:	+2482534202
country:	US	e-mail:	adw0rd.yandex.ru@gmail.com
admin-c:	IP9-AFRINIC	nic-hdl:	IP9-AFRINIC
tech-c:	IP9-AFRINIC	changed:	adw0rd.yandex.ru@gmail.com
status:	ASSIGNED PA	20151014	
mnt-by:	IP-ADMIN	source:	AFRINIC
mnt-lower:	IP-ADMIN		
mnt-domains:	IP-ADMIN		
mnt-routes:	IP-ADMIN		
changed:	adw0rd.yandex.ru@gmail.com 20151014		
source:	AFRINIC		
parent:	196.62.0.0 - 196.62.255.255		

Methbot - IP Registration Forgery

```
inetnum:          196.62.32.0 - 196.62.63.255
netname:         TIME-WARNER
descr:          Time Warner Cable Inc.
country:        US
admin-c:         IP9-AFRINIC
tech-c:          IP9-AFRINIC
status:          ASSIGNED PA
mnt-by:          IP-ADMIN
mnt-lower:       IP-ADMIN
mnt-domains:    IP-ADMIN
mnt-routes:     IP-ADMIN
source:          AFRINIC # Filtered
parent:          196.62.0.0 - 196.62.255.255
```

Methbot - IP Registration Forgery

inetnum: 196.62.64.0 - 196.62.95.255
netname: VERIZON
descr: Verizon Trademark Services LLC
country: US
admin-c: IP9-AFRINIC
tech-c: IP9-AFRINIC
status: ASSIGNED PA
mnt-by: IP-ADMIN
mnt-lower: IP-ADMIN
mnt-domains: IP-ADMIN
mnt-routes: IP-ADMIN
source: AFRINIC # Filtered
parent: 196.62.0.0 - 196.62.255.255

Methbot - IP Registration Forgery

```
inetnum:          196.62.96.0 - 196.62.127.255
netname:         ATT
descr:          AT&T Services, Inc.
country:        US
admin-c:         IP9-AFRINIC
tech-c:          IP9-AFRINIC
status:          ASSIGNED PA
mnt-by:          IP-ADMIN
mnt-lower:       IP-ADMIN
mnt-domains:    IP-ADMIN
mnt-routes:     IP-ADMIN
source:          AFRINIC # Filtered
parent:          196.62.0.0 - 196.62.255.255
```

Methbot - IP Registration Forgery

```
inetnum:          196.62.128.0 - 196.62.159.255
netname:          COX
descr:           Cox Communications Inc
country:          US
admin-c:          IP9-AFRINIC
tech-c:           IP9-AFRINIC
status:           ASSIGNED PA
mnt-by:           IP-ADMIN
mnt-lower:        IP-ADMIN
mnt-domains:     IP-ADMIN
mnt-routes:       IP-ADMIN
source:           AFRINIC # Filtered
parent:          196.62.0.0 - 196.62.255.255
```

Methbot - IP Registration Forgery

```
inetnum:          196.62.160.0 - 196.62.191.255
netname:         CHARTER
descr:          Charter Communications Operating, LLC
country:        US
admin-c:        IP9-AFRINIC
tech-c:         IP9-AFRINIC
status:         ASSIGNED PA
mnt-by:         IP-ADMIN
mnt-lower:      IP-ADMIN
mnt-domains:   IP-ADMIN
mnt-routes:    IP-ADMIN
source:         AFRINIC # Filtered
parent:         196.62.0.0 - 196.62.255.255
```

Methbot - IP Registration Forgery

```
inetnum:          196.62.192.0 - 196.62.223.255
netname:         Cequel
descr:          Cequel Communications Holdings
country:        US
admin-c:         IP9-AFRINIC
tech-c:          IP9-AFRINIC
status:          ASSIGNED PA
mnt-by:          IP-ADMIN
mnt-lower:       IP-ADMIN
mnt-domains:    IP-ADMIN
mnt-routes:     IP-ADMIN
source:          AFRINIC # Filtered
parent:          196.62.0.0 - 196.62.255.255
```

Methbot - IP Registration Forgery

```
inetnum:          196.62.224.0 - 196.62.255.255
netname:         CenturyLink
descr:          CenturyLink, Inc.
country:        US
admin-c:         IP9-AFRINIC
tech-c:          IP9-AFRINIC
status:          ASSIGNED PA
mnt-by:          IP-ADMIN
mnt-lower:       IP-ADMIN
mnt-domains:    IP-ADMIN
mnt-routes:     IP-ADMIN
source:          AFRINIC # Filtered
parent:          196.62.0.0 - 196.62.255.255
```

Methbot - IP Registration Forgery

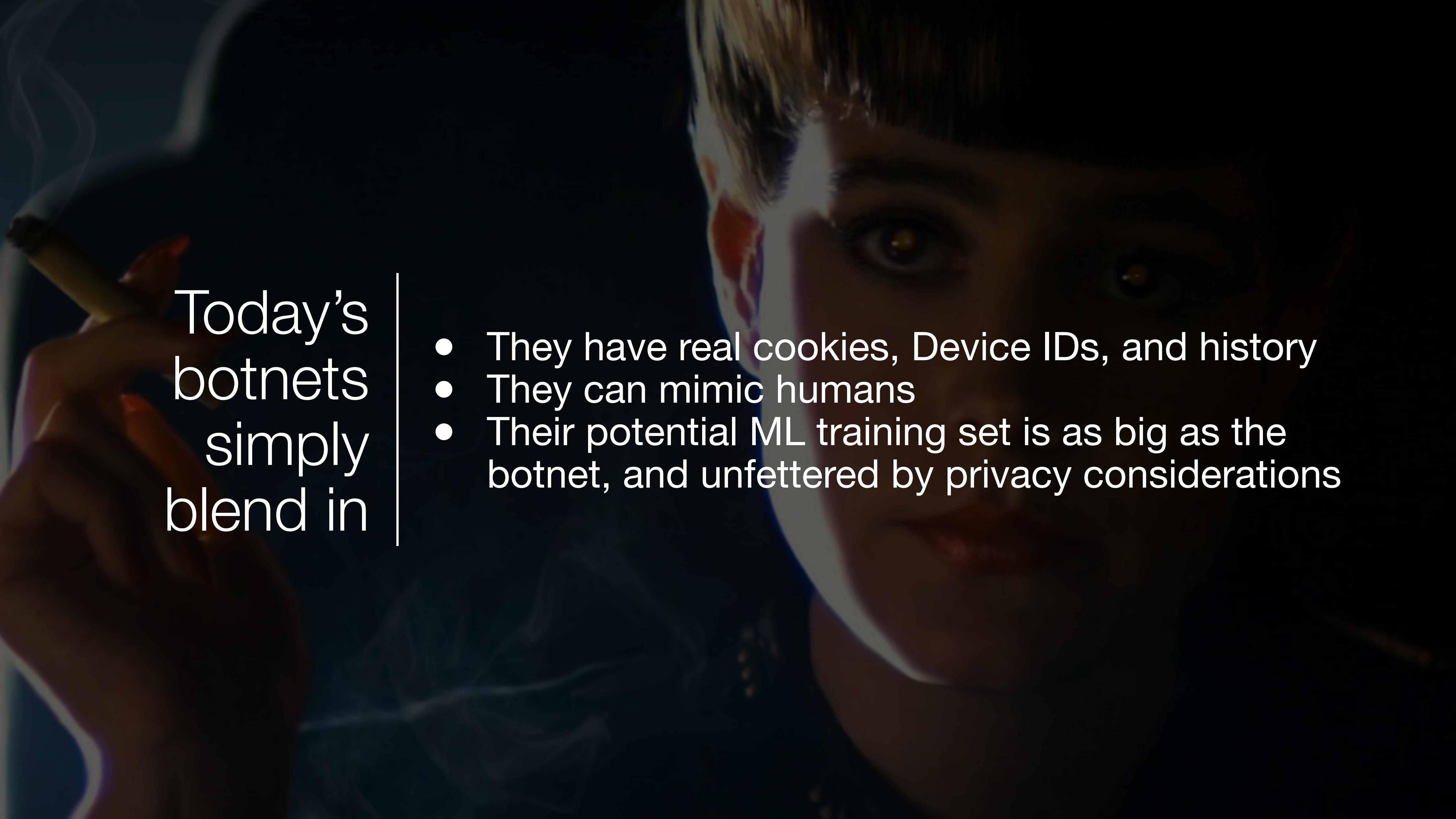
% Abuse contact for '161.8.192.0 - 161.8.223.255' is 'stepanenko.aa@mmk.ru'

inetnum:	161.8.192.0 - 161.8.223.255		
netname:	Verizon_Trademark_Services_LLC-19		
descr:	Verizon Trademark Services LLC	person:	NetBComm LLC
country:	US	address:	USA, Texas, Dallas ,
admin-c:	SOV68-RIPE	Verizon Trademark Services LLC	
tech-c:	SOV68-RIPE	phone:	+12191278854
status:	LEGACY	nic-hdl:	SOV68-RIPE
mnt-by:	MMKMGN-MNT	mnt-by:	NetBC
mnt-by:	NetBC	created:	2015-07-20T07:15:59Z
created:	2015-10-13T14:47:56Z	last-modified:	2015-12-25T08:57:55Z
last-modified:	2015-10-13T14:47:56Z	source:	RIPE # Filtered
source:	RIPE		

The ultimate option (for now)

Today's hybrid approach

- Use real browsers!
- Completely control the operating environment of your real browsers for performance, management, scalability... in server farms under your control
- Then use a residential botnet *just to proxy your traffic*



Today's
botnets
simply
blend in

- They have real cookies, Device IDs, and history
- They can mimic humans
- Their potential ML training set is as big as the botnet, and unfettered by privacy considerations

The lesson:

**It doesn't matter what fraud detection
technique you have today.**

**What matters is how you run the arms
race.**

All arms races are resource depletion games

- The side that wins is the side that can run the arms race longer.
- There are only two winning patterns:
 1. Start richer and stay richer
 2. Tilt the arms race so that your side can keep up at a much lower cost than the other side

What to do when your enemy is smarter and richer?

We must model our adversaries as:

- at least as smart as us
- quite possibly better resourced than us

Each day they win, they get fraud profits; each day we win, we get to keep our stressful infosec jobs

Looking human is easier than you think

The game we play is out in the open

The adversary:

- has access to our payload
- can monkey-patch our payload to selectively execute it or not execute it at all
- can monitor and manipulate what is sent back to our servers
- can analyze how normal browsers execute our payload, and what data is sent back to our servers
- can operate their fraud operations off of real computers using real identity tokens (e.g. cookies) and real behavior patterns (real mouse movements, matching the times of day of real visitors, visiting the same sites as real people as well as the sites that earn them money)

But they have to play it

They adversary:

- must accept our payload
- must download it from us
- must send data back to us

because all real browsers and apps do.

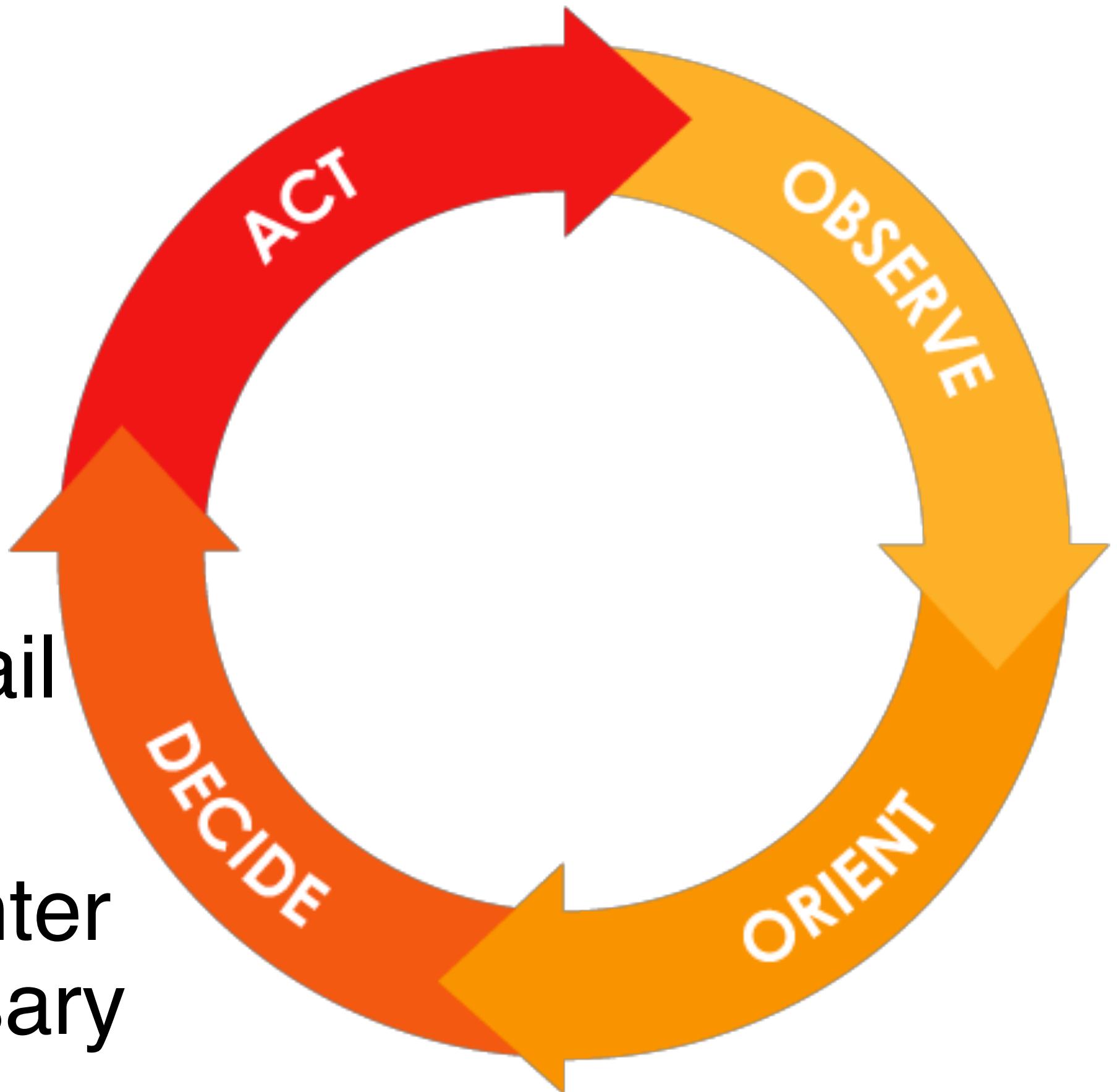
Our adversaries can try to beat our game, but they can't refuse to play our game.

The OODA Loop

There's already a formal framework for heads-up competition between equally capable adaptive adversaries: it's called the OODA Loop.

The trick here is to get inside the adversary's OODA loop by:

- presenting a dynamic challenge
- denying the adversary immediate success/fail feedback
- changing the challenge on a periodicity slightly shorter than the feedback loop to the adversary



Result:

The adversary is forced to play the next round of the game before they can tell if they won the last round or not.

To win like this:

You need a sufficiently large parameter space of possible fresh dynamic challenges to last until [victory, the life of your project, the heat death of the universe... choose what fits your threat model]

Can you apply these principles in your own arms races?

Yes, if:

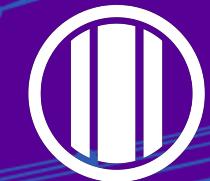
- You can trigger a silent alarm
- You can stop the adversary from benefitting from her action without interrupting that action

Denying immediate success/fail feedback is key.

Harder locks add only time to the lockpicker's attack. The silent alarm completely changes his risk:reward calculation.

RSA® Conference 2019

Thank you!



White Ops®