安某客滑块逆向

逆向目标

- 目标：安某客滑动验证码逆向分析
- 主页：

  aHR0cHM6Ly93d3cuYW5qdWtlLmNvbS9jYXB0Y2hhLXZlcmlmeS8/Y2FsbGJhY2s9c2hpZWxkJmZyb209YW50aXNwYW0=

抓包分析

滑动滑块 -》

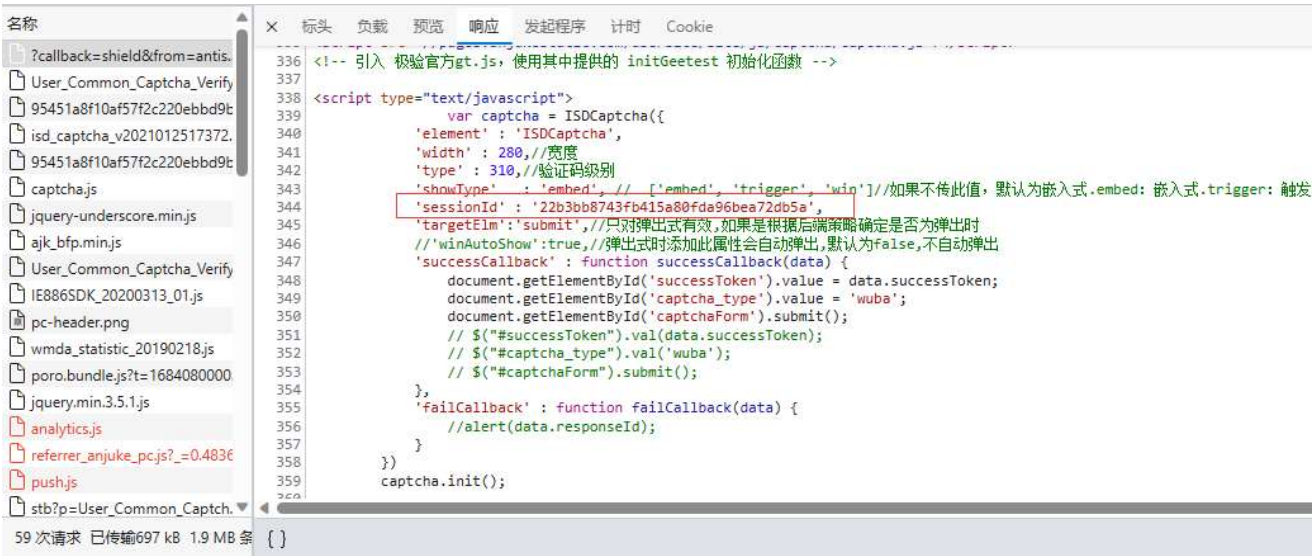
请求checkInfoTp，Form Data 里有以下参数**sessionId**、**responseId**、**dInfo**、**language**、**data** -》

响应"message":"校验失败"

接着往上抓包分析、找参数生成逻辑

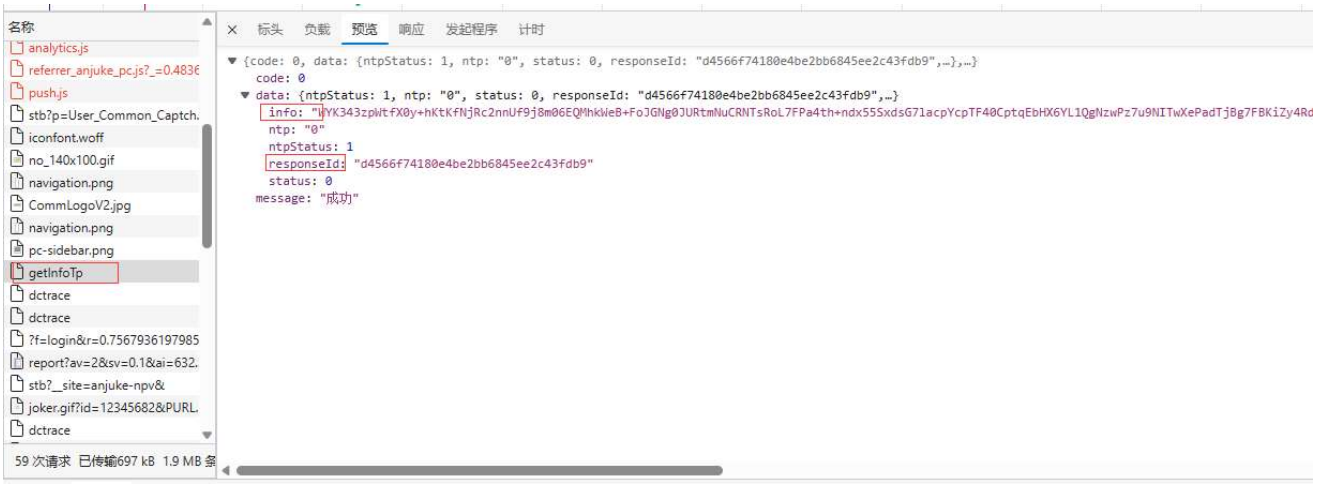
1、sessionId

首页请求，其中有个 sessionId



2、dInfo

然后有个 getInfoTp 的请求，Form Data 里有 dInfo 是加密参数，返回值里 info 也是加密的，包含了图片信息，返回值 responseId 在后续的请求也会用到。



XHR断点，找到dInfo参数。大致就可以看出是 AES 加密，传入了 sessionId 和一个 *taN() 函数的返回值*，taN() 函数是一些 URL，UA 之类的信息，可以写死

往里跟就可以看到 AES 算法了，我们js还原一下，还原代码为js文件里的function AESencrypt{}方法

```
,
_EPy[_0x90o[67]] = function(_cRV, _2undefinedp) {
    if (false) {
        var _0xisab = 778 >> 124 % 940 >> 724 % 64;
    } else
        _2undefinedp = _2undefinedp[_0x90o[115]](_0x90o[17])[_0x90o[164]](function(_PUi, _JrX, _JP9) {
            if (true + 1 + undefined) {
                var _0x18pp = 87 & 393 << 237;
            } else
                return _JP9 % 2 == 0 ? _PUi + _0x90o[17] : _PUi + _JrX;
        }, _0x90o[17]),
        _2undefinedp = _jWF[_0x90o[54]](_2undefinedp),
        _cRV = _0x90o[53] == typeof  _cRV : JSON[_0x90o[108]](_cRV),
        _cRV = _WMx[_0x90o[165]](_( "CBC" ndefinedp, {
            iv: _2undefinedp,
            mode: _mjv[_0x90o[166]][_0x90o[167]],
            padding: _GrF
        }),
        _2undefinedp = _9kJ[_0x90o[108]](_cRV[_0x90o[168]]);
    return (0,
    encodeURIComponent)(_2undefinedp);
}
```

3、info、sessionId

getInfoTp 这个接口返回的 info 的值是加密的，前面我们已经知道用到了 AES 加密算法，这里可以直接猜测也是用的的 AES 来解密的，找到 AESDecrypt 这个方法，此方法与上面的加密方法写在了一起，刷新发现断下之后传入了两个参数，第一个正是 info 的内容，第二个则是 sessionId。

还原代码为js文件里的function AESdecrypt{}方法

```
                    ,
                    _EPy[_0x90o[61]] = function(_jzC, _73M) {  _jzC = "u9ENbjdx+t4C8Gt+gj4BYZfb3zm4kZdEQ1q3iTL4xE+L/gWZJTB0h5NrgnQQDQmD2Ppv2A11DERgm5xvzib
                        if ((415 % 8 / 7 == 1) - !![]) {
                            var _0x35kt = 113 + 56 * 873;
                        } else
                            _73M = _73M[_0x90o[115]](_0x90o[17])[_0x90o[164]](function(_Szd, _PgX, _3fK) {
                                if (!![])
                                    return _3fK % 2 == 0 ? _Szd + _0x90o[17] : _Szd + _PgX;
                                else {
                                    var _0xer816u = {
                                        _0x2s3oqp: function(_0xj5qbaj, _0xpoiq6j) {
                                            return _0xj5qbaj | _0xpoiq6j;
                                        }
                                    };
                                    var _0xo2vz = _0xer816u._0x2s3oqp(136 | 101, 750);
                                }
                            }, _0x90o[17]),
                            _73M = _jWF[_0x90o[54]](_73M),
                            _jzC = _WMx[_0x90o[169]](_jzC, _73M, {
                                iv: _73M,
                                mode: _mjv[_0x90o[166]][_0x90o[167]],
                                padding: _GrF
                            })[_0x90o[170]](_jWF);
                        if (false - 1)
                            return JSON[_0x90o[54]](_jzC);
                        else {
                            var _0xs41d = 759 % 645 ^ 99 * 77;
                        }
                    }
```

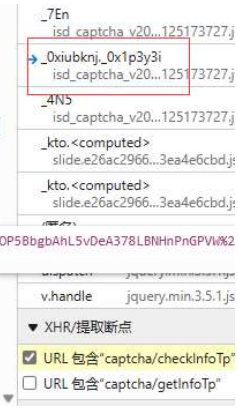
解密结果可以看到滑块的图片地址等信息：

```
                        return _0xj5qbaj | _0xpoiq6j;
                    }
                };
                var _0xo2vz = _0xer816u._0x2s3oqp(136 | 101, 750);
            }
        }, _0x90o[17]),
        _73M = _jWF[_0x90o[54]](_73M),  _73M = init {words: Array(4), sigBytes: 16}
        _jzC = _WMx[_0x90o[169]](_jzC, _73M, {  _jzC = "{\"level\":310,\"isimgUrls\":1,\"bgImgUrl\":\"https://wos3.58cdn.com.cn/Him
            iv: _73M,
1,"bgImgUrl":"https://wos3.58cdn.com.cn/HimeDcBazHkN/captchaimg/cb241c40-5691-465a-8180-3334bf2df050.png","width":"280","tip":"请点击并
    if (false - 1)
        return JSON[_0x90o[54]](_jzC);
    else {
        var _0xs41d = 759 % 645 ^ 99 * 77;
    }
}
```

4、data

XHR断点，找到堆栈，发现_5DD 就是 data 值

```
    var _0xr5t9 = 995 * 578 | 699 >> 659 + 553;
}
_4N5 = function(_Rub, _5DD, _Lnn) {
if (true + (8 * 454 >> 5 == 1))
    return new Promise(function(_drs, _undefinednx) {  _drs = ƒ (), _undefinednx = ƒ ()
        if (1 + false * (8 * 564 << 5 > 0))
            (0,
            _7En)({
                type: "\u0050\u004f\u0053\u0054",
                url: _dhN[_0x90o[76]],
                dataType: "\u006a\u0073\u006f\u006e",
                data: {
                    sessionId: _Lnn,
%2BID2U5Gf9i8BO159L1pDX2WJ12uOoOJikba64NwWs5I3jF0KsBhIboiW2xUHQcBw9CYRUqgxC5hOG8yY5v0ENE9rmppSTP%2BDy8jm8jba5bBFw4gSKRHkKNCXvzA9S%2FaGGPVU6h3OP5BbgbAhL5vDeA378LBNHnPnGPVW%2
                    data: _5DD
                }
            })[_0x90o[51]](function(_mXY) {
                if ((415 % 8 / 7 == 1) - null)
                    _mXY && _mXY[_0x90o[70]] === _Baj[_0x90o[62]] ? (0,
                    _drs)(_mXY) : (0,
                    _undefinednx)(_mXY[_0x90o[72]] || _0x90o[17]),
```



往上跟栈，可以看到 _Ug0 里面有个 track 参数，这明显就是轨迹了，同样最后的结果经过了 AES 加密。

然后往上找，可以看到 _Ug0 由三个参数组成，x 是水平滑动的距离，track 是轨迹，p 是定值。

```
        _rrA += _1kW[_vmT][_0x9pl[238]] + _0x9pl[221] + _1kW[_vmT][_0x9pl[239]] + _0x9pl[240];
if ((415 % 8 / 7 == 1) * (8 * 564 << 5 > 0))
    _Ug0 = {  _Ug0 = {x: 136, track: '29,17,0|30,17,51|31,17,55|31,18,58|32,18,58|33,18,…,224|163,24,228|164,24,234|165,24,4
        x: _Ug0 || 0,
        track: _rrA,  _rrA = "29,17,0|30,17,51|31,17,55|31,18,58|32,18,58|33,18,61|34,18,63|35,18,65|35,19,66|36,19,66|37,19
        p: _mGq[_0x9pl[21]][_0x9pl[241]]
    };
else {
    var _0xx0xutx = {  _0xx0xutx = undefined
        _0xnysmcr: function(_0x88uh2f, _0x717c67) {
            return _0x88uh2f & _0x717c67;
        }
    };
    var _0xci78 = _0xx0xutx._0xnysmcr(409, 557);  _0xci78 = undefined, _0xx0xutx = undefined
}
```

4.1 轨迹track: _rrA处理

轨迹生成前，得先识别缺口得到要滑动的距离，这里选用OpenCV库。注意的是图片是有缩放的，原始尺寸 480 × 270 px 渲染后的尺寸 280 × 158 px，比例大概是 1:0.5833333，我们先识别距离后再将距离进行缩放。

轨迹的处理，该站点校验并不太严格，所以可以自己写一下。本例中可以使用缩放法，先采集一条正常的，手动滑出来的轨迹，然后根据识别出的实际距离（缩放后的）和样本轨迹中的距离相比，得到一个比值，然后将样本中的 x 值和实际值都做一个对应的缩放，生成新的轨迹

总结

负载dinfo、sessionId --》 请求getinfotp --》 响应info、responseId --》

负载sessionId、responseId、dInfo、data --》 请求checkinfotp --》校验是否成功

```
2023-05-15 23:50:10.836 | INFO     | __main__:get_sessionId:31 - sessionId ==> b8588410c9b6461a99370f3f94ac0807
2023-05-15 23:50:12.672 | INFO     | __main__:get_dInfo:42 - dInfo ==> t%2BKcu0tA7a4F4oX7J10u%2Bajy26%2FM8rOCV0zoSNdn50Ud7auUnkI4dZMJRCLU0Q8%2FMdVWU7fqHJ
2023-05-15 23:50:12.829 | INFO     | __main__:req_getinfotp:52 - info ==> AuPlr1uPlN3rvdRXGPrlp0LxPfLr5ykWRvtus7oDwMzBACPJMbZZdEnQAr5tqh3oVdvIvVB9jHE0IJI
2023-05-15 23:50:12.830 | INFO     | __main__:req_getinfotp:53 - responseId ==> bc4cdc84666d47dead41bcb1f39ea482
2023-05-15 23:50:13.199 | INFO     | __main__:identify_gap:94 - 缺口位置==> 149
2023-05-15 23:50:13.199 | INFO     | __main__:identify_gap:95 - 缩放后==> 86
2023-05-15 23:50:13.199 | INFO     | __main__:generate_track:106 - 生成轨迹 ==> 15,23,0|16,23,43|16,23,45|17,23,47|17,23,49|18,23,50|18,23,52|19,23,53|19,2
2023-05-15 23:50:13.199 | INFO     | __main__:generate_track:107 - 轨迹长度 ==> 1639
2023-05-15 23:50:13.407 | SUCCESS  | __main__:req_checkInfoTp:124 - 校验成功
```