1st July 2020, 12:52 PM

kmsmym
Hacker Supreme

Join Date: Jul 2017

Location: Russia

Posts: 232

Reputation: 3306
Rep Power: 128

Points: 7,834, Level: 10

Level up: 22%, 866 Points needed

Activity: 2.0%

Last Achievements

Quote:

> Originally Posted by **lobotomee**
> *that tutorial help me a lot but now I stuck trying to find ObjectManager offset... can you explain how we find this one?*
>
> *plz dont answer: The offset is 0x1C41DB0.. I wanna know how to find it on IDA pro.*
>
> *Thanks to everyone!*

The easiest way to do this is by using the signature pattern, using them in 2 clicks you can find the offset.
https://github.com/ajkhoury/SigMaker-x64

First you get the signaling, knowing the current offset, and then in the new database use it

____

It was nice, but Now it's gone

fvicaria
Active Member
Active Member

Reputation:	29
Join Date:	Jan 2009
Posts:	55
Thanks G/R:	0/0
Trade Feedback:	0 (0%)
Mentioned:	0 Post(s)
Tagged:	0 Thread(s)

How to make signature and search with IDA

Hello all,

I am adding this very simple tutorial on how to create and search for a pattern using IDA. Nothing very complex but useful for those starting.

What you need:
1. IDA
2. Pattern Maker Plugin
3. A pattern search algorithm

I am assuming you have already found a sequence of bytes, that is a region of code that you identified as important for you. Maybe the location of your hack.

Say you want to save it so next time WoW updates you can easily find it.

The main issue here is that you can't simply search for the same sequence of bytes. A signature rarely remains the same across patches due to changes in offsets and other dynamic data.
Putting it in other words your signature will break every time there is a new patch.
What you need is to create a pattern and remove all those bytes that are likely to change.

The solution is to create a mask to go along with your signature where you identify the byte positions that are likely to change.

For example:

A typical signature could be:
"\x56\xE8\xFA\x71\x01\x00\x83\x3D\x5C\x99\x22\x0D\x00\x8B\x35\x64\xA5\x11\x0D\x0 F\x84\x93\x01\x00\x00\x68\xDC\x8C\x13\x0D\xFF\xD6"

And the associated mask would be:
"xx????xx????xxx????xx????x????xx"

Here I identified the static bytes (i.e those opcodes that are unlikely to change with 'X')
and the dynamic data (i.e. those that are more likely to change with '?')

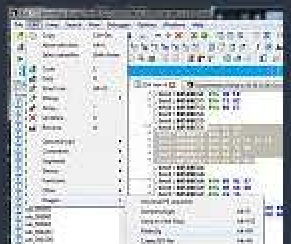
The process of creating a byte signature and a mask can be quite slow, depending on the number of bytes you selected.
To simplify things you could use one of the many pattern maker examples posted in this forum. Just use the good old search.

In this post I decided to use an IDA plugin: SigMaker (credit goes to bobbysing).

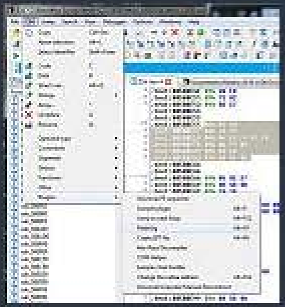
I uploaded it here: Filebeam - Beam up that File Scottie!

Just place it in your IDA Plugins directory.

Once you have done it you just need to select a chunk of bytes in IDA and click F9 hotkey or use the menu Edit -> Plugins -> MakeSig



Once you have done it you just need to select a chunk of bytes in IDA and click F9 hotkey or use the menu Edit -> Plugins -> MakeSig



The result will appear in your output window or can be save to your clipboard. Easy.

OK now you have your Sig+Mask what do you do with it?

Well you keep it until the new patch comes and you need to find the location of your hack again in the new code.

I created a very simple class to help you with that. This is a cut down version of my actual code so you may want to change it.
I allows you to stop at the first match or continue to the next.

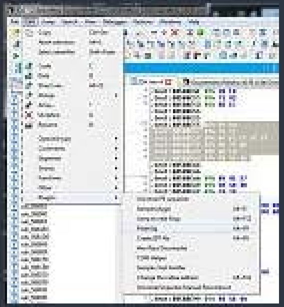
Code:

```
    ///
    /// Finds a byte pattern given a signature and a mask.
    /// It will process the IDA output from SigMaker directly.
    /// The result can be given as an asbsolute address, used in dynamic searches or as an offset (static).
    ///
    public class FindPattern
    {
        private uint _currentIndex;
        private readonly BlackMagic _memory;

        public FindPattern(BlackMagic memory)
        {
            _memory = memory;
        }

        private static bool Check(IList data, IEnumerable sig, string mask)
        {
            // Verifies opcodes match on data and signature if it's static ('x') or ignores if not static ('?').
            // Opcodes don't match so return false.
            return !sig.Where((t, index) => mask[index] == 'x' && data[index] != t).Any();
        }

        private static uint GetHexValueAsUInt(string value)
        {
            if (string.IsNullOrEmpty(value) || value == "0")
                return 0;

            value = value.TrimStart('0');
            value = value.TrimStart('x');
```

Code:

```
    ///
    /// Finds a byte pattern given a signature and a mask.
    /// It will process the IDA output from SigMaker directly.
    /// The result can be given as an asbsolute address, used in dynamic searches or as an offset
(static).
    ///
    public class FindPattern
    {
        private uint _currentIndex;
        private readonly BlackMagic _memory;

        public FindPattern(BlackMagic memory)
        {
            _memory = memory;
        }

        private static bool Check(IList data, IEnumerable sig, string mask)
        {
            // Verifies opcodes match on data and signature if it's static ('x') or ignores if not static ('?').
            // Opcodes don't match so return false.
            return !sig.Where((t, index) => mask[index] == 'x' && data[index] != t).Any();
        }

        private static uint GetHexValueAsUInt(string value)
        {
            if (string.IsNullOrEmpty(value) || value == "0")
                return 0;

            value = value.TrimStart('0');
            value = value.TrimStart('x');

            return uint.Parse(value, System.Globalization.NumberStyles.HexNumber);
        }


        // Given an initial address and a size to search and the byte signature + mask find if
        // we have a match somewhere (will return the first match only).
        // This method will take the string output from SiMaker plugin directly (sig + mask).
        public int SearchForSig(uint startAddress, uint size, string sigStr, string mask, bool
searchAgain = false, bool returnOffset = true)
        {
            // Reset previous search result.
            if (!searchAgain)
                _currentIndex = 0;

            var bytesStr = sigStr.Split(new [] { '\\' }, StringSplitOptions.RemoveEmptyEntries);
            var sig = new byte[bytesStr.Length];

            for (var i = 0; i= size - sig.Length)
                return -1;

            // Search through the full block of memory...
            for (var i = _currentIndex; i < size - sig.Length; i++)
            {
                var data = _memory.ReadBytes(startAddress + i, sig.Length);
                if (!Check(data, sig, mask))
                    continue;

                _currentIndex = i;

                // Return the offset or absolute address.
                return returnOffset
                        ? (int) ((startAddress + i) - (int) _memory.MainModule.BaseAddress)  // offset
                        : (int) (startAddress + i);                                     // absolute
            }

            return -1;
        }
    }
```

Once you have done it you just need to select a chunk of bytes in IDA and click F9 hotkey or use the menu Edit -> Plugins -> MakeSig



The result will appear in your output window or can be save to your clipboard. Easy.

OK now you have your Sig+Mask what do you do with it?

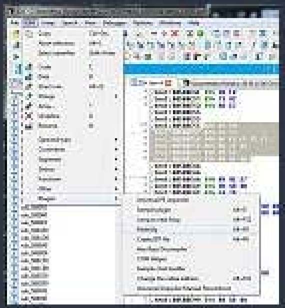
Well you keep it until the new patch comes and you need to find the location of your hack again in the new code.

I created a very simple class to help you with that. This is a cut down version of my actual code so you may want to change it.
I allows you to stop at the first match or continue to the next.

Code:

```
            value = value.TrimStart('0');
            value = value.TrimStart('x');

            return uint.Parse(value, System.Globalization.NumberStyles.HexNumber);
        }


        // Given an initial address and a size to search and the byte signature + mask find if
        // we have a match somewhere (will return the first match only).
        // This method will take the string output from SiMaker plugin directly (sig + mask).
        public int SearchForSig(uint startAddress, uint size, string sigStr, string mask, bool searchAgain = false, bool returnOffset = true)
        {
            // Reset previous search result.
            if (!searchAgain)
                _currentIndex = 0;

            var bytesStr = sigStr.Split(new [] { '\\' }, StringSplitOptions.RemoveEmptyEntries);
            var sig = new byte[bytesStr.Length];

            for (var i = 0; i= size - sig.Length)
                return -1;

            // Search through the full block of memory...
            for (var i = _currentIndex; i < size - sig.Length; i++)
            {
                var data = _memory.ReadBytes(startAddress + i, sig.Length);
                if (!Check(data, sig, mask))
                    continue;

                _currentIndex = i;
```

Once you have done it you just need to select a chunk of bytes in IDA and click F9 hotkey or use the menu Edit -> Plugins -> MakeSig



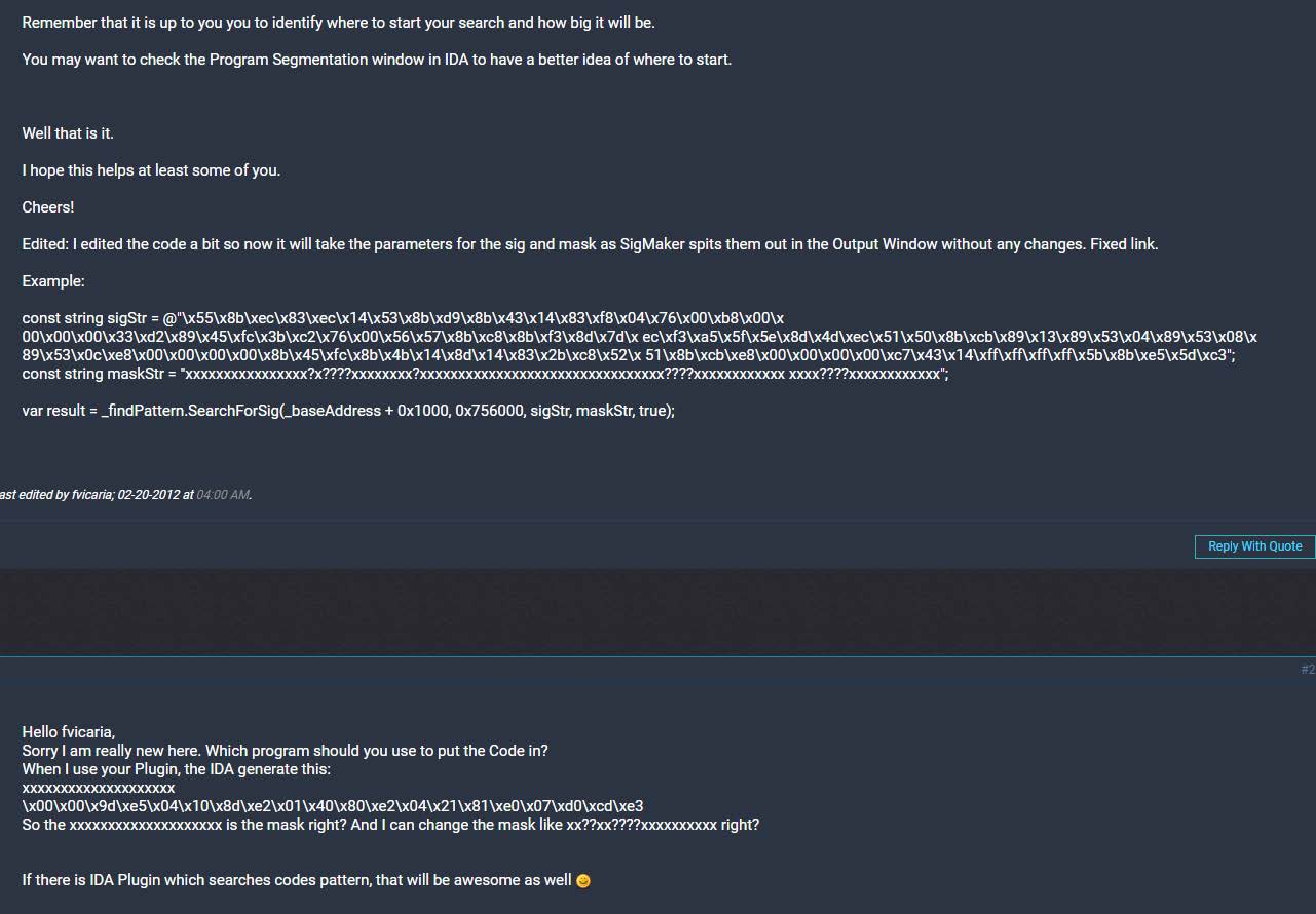
The result will appear in your output window or can be save to your clipboard. Easy.

OK now you have your Sig+Mask what do you do with it?

Well you keep it until the new patch comes and you need to find the location of your hack again in the new code.

I created a very simple class to help you with that. This is a cut down version of my actual code so you may want to change it.
I allows you to stop at the first match or continue to the next.

Code:
```
        public int SearchForSig(uint startAddress, uint size, string sigStr, string mask, bool searchAgain = false, bool returnOffset = true)
        {
            // Reset previous search result.
            if (!searchAgain)
                _currentIndex = 0;

            var bytesStr = sigStr.Split(new [] { '\\' }, StringSplitOptions.RemoveEmptyEntries);
            var sig = new byte[bytesStr.Length];

            for (var i = 0; i= size - sig.Length)
                return -1;

            // Search through the full block of memory...
            for (var i = _currentIndex; i < size - sig.Length; i++)
            {
                var data = _memory.ReadBytes(startAddress + i, sig.Length);
                if (!Check(data, sig, mask))
                    continue;

                _currentIndex = i;

                // Return the offset or absolute address.
                return returnOffset
                           ? (int) ((startAddress + i) - (int) _memory.MainModule.BaseAddress)  // offset
                           : (int) (startAddress + i);                                         // absolute
            }

            return -1;
        }
    }
```



Remember that it is up to you you to identify where to start your search and how big it will be.

You may want to check the Program Segmentation window in IDA to have a better idea of where to start.

Well that is it.

I hope this helps at least some of you.

Cheers!

Edited: I edited the code a bit so now it will take the parameters for the sig and mask as SigMaker spits them out in the Output Window without any changes. Fixed link.

Example:

```
const string sigStr = @"\x55\x8b\xec\x83\xec\x14\x53\x8b\xd9\x8b\x43\x14\x83\xf8\x04\x76\x00\xb8\x00\x
00\x00\x00\x33\xd2\x89\x45\xfc\x3b\xc2\x76\x00\x56\x57\x8b\xc8\x8b\xf3\x8d\x7d\x ec\xf3\xa5\x5f\x5e\x8d\x4d\xec\x51\x50\x8b\xcb\x89\x13\x89\x53\x04\x89\x53\x08\x
89\x53\x0c\xe8\x00\x00\x00\x00\x8b\x45\xfc\x8b\x4b\x14\x8d\x14\x83\x2b\xc8\x52\x 51\x8b\xcb\xe8\x00\x00\x00\x00\xc7\x43\x14\xff\xff\xff\xff\x5b\x8b\xe5\x5d\xc3";
const string maskStr = "xxxxxxxxxxxxxxxx?x????xxxxxxxx?xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx????xxxxxxxxxxxx xxxx????xxxxxxxxxxxx";

var result = _findPattern.SearchForSig(_baseAddress + 0x1000, 0x756000, sigStr, maskStr, true);
```

ast edited by fvicaria; 02-20-2012 at 04:00 AM.

Reply With Quote

#2

Hello fvicaria,
Sorry I am really new here. Which program should you use to put the Code in?
When I use your Plugin, the IDA generate this:
xxxxxxxxxxxxxxxxxxxx
\x00\x00\x9d\xe5\x04\x10\x8d\xe2\x01\x40\x80\xe2\x04\x21\x81\xe0\x07\xd0\xcd\xe3
So the xxxxxxxxxxxxxxxxxxxx is the mask right? And I can change the mask like xx??xx????xxxxxxxxxx right?

If there is IDA Plugin which searches codes pattern, that will be awesome as well 😊

Threk

Member

Lazy Leecher

Reputation:	1
Join Date:	Oct 2010
Posts:	23
Thanks G/R:	0/0
Trade Feedback:	0 (0%)
Mentioned:	0 Post(s)
Tagged:	0 Thread(s)

> Originally Posted by GreatThings
>
> Hello fvicaria,
> Sorry I am really new here. Which program should you use to put the Code in?
> When I use your Plugin, the IDA generate this:
> xxxxxxxxxxxxxxxxxxxx
> \x00\x00\x9d\xe5\x04\x10\x8d\xe2\x01\x40\x80\xe2\x04\x21\x81\xe0\x07\xd0\xcd\xe3
> So the xxxxxxxxxxxxxxxxxxxx is the mask right? And I can change the mask like xx??xx????xxxxxxxxxx right?
>
> If there is IDA Plugin which searches codes pattern, that will be awesome as well 😊

Just convert your code pattern to a sequence of bytes / ida pattern (or how you want to call it) and use the Search for sequence of bytes.

@Topic
You can use the IDA Plugin "sigmaker full" of P47R!CK to generate patterns (both IDA and code patterns).

Last edited by Threk; 05-04-2012 at 05:34 AM.