**Greeting From Reserver :**

**How I Crack Your Code With .NET**

**By Satria Ady Pradana**

**21 Oktober 2017**

Oke, selamat malam semuanya, terima kasih sudah hadir di kulwap ini.

Ada pepatah mengatakan, tak kenal maka kenalan, jadi kita kenalan dulu.

Nama saya Satria Ady Pradana, kalo cewek boleh panggil Mas Ganteng. Kalo cowok cukup panggil aja Satria.

Saat ini sebagai Cyber Security Consultant di Mitra Integrasi Informatika, di komunitas lain aktif sebagai Community Lead di Reversing.ID, komunitas yang membahas reverse engineering untuk segala macam, mulai dari software, hardware, protocol, dsb pokoknya sebuah system.

Sebagai seorang konsultan keamanan sistem, lebih sering melakukan kegiatan penetration testing ke suatu sistem. Mostly web application dan web service. Tapi kadang beberapa kali ada suatu kebutuhan untuk melakukan reverse engineering terhadap aplikasi client

Kalo disuruh menceritakan apa yang sudah pernah dibuat, gak ada something yang gede yang bisa ditunjukkan. Karena saya tukang bongkar. Kalo disuruh menyebutkan apa yang sudah dibongkar, gak bisa sebut spesifik karena NDA, tapi yang pasti beberapa ada mobile app, desktop app, web app (terutama basis .net)

**So, Greeting From Reverser:**

Reverser adalah satu pihak yang cukup ditakuti oleh pengembang aplikasi komersial.

Sesi ini akan sedikit membahas tentang reverse engineering, terhadap aplikasi .net.

***Apa itu reversing?***

***Tools apa yang diperlukan?***

***Apa yang dilakukan ketika reversing?***

***dan case yang pernah dihadapi.***

As said before, karena aku gak bikin sample code sebelumnya, jadi kita bikin secara live aja :D

**Jadi, apa itu reverse engineering?**

Secara sederhana Reverse Engineering adalah proses untuk mengekstrak informasi terutama desain cara kerja dari suatu sistem. Orang-orang lebih mengenal crack daripada RE, tapi sebenarnya RE tidak terbatas crack saja. RE lebih luas bahasannya, dan lagi-lagi, RE lebih banyak dikenal buat tindakan yang gak bener, seperti nyolong dan crack. Padahal ada beberapa hal legitimate yang bisa dilakukan dengan RE. Salah dua yang paling utama adalah menjinakkan malware, dan mencari celah keamanan.

**Di dunia Computer Science, ada 2 representasi program: native dan interpreted.**

Aplikasi native adalah aplikasi yang jalan langsung di atas OS maupun baremetal, menggunakan instruksi prosesor langsung, tanpa adanya suatu layer tambahan.

Aplikasi interpreted adalah aplikasi yang jalan di atas sebuah process VM alias membutuhkan suatu layer lagi.

Contoh aplikasi interpreted bisa scripting seperti Python, Ruby, dsb, ada lagi yang mengalami compilation menjadi bytecode seperti Java.

**Berbeda dengan keyakinan banyak orang, aplikasi .NET adalah aplikasi yang interpreted, seperti Java. Format dan ekstensi .exe tidak membuat aplikasi .NET menjadi aplikasi Native.**

Aplikasi .NET, entah dibuat dari C# maupun VB, pada akhirnya akan di-compile menjadi sebuah bytecode yang disebut MSIL (MicroSoft Intermediate Language) ato CIL (Common Intermediate Language). Kode .NET CIL akan berjalan di atas CLR (Common Languare Runtime) bagian dari .Net Framework.

Itulah kenapa jika tidak menginstall .Net framework, aplikasi .net tidak akan bisa dijalankan.

**Dan ini yang menarik, sebuah mantra yang perlu diingat: most interpreted system dapat dikembalikan menjadi bentuk source code-nya, mungkin mirip (beda variable) tapi tidak harus sama persis.**

Ini karena bagi .net (dan java), membubuhkan semacam metadata yang memberitahu informasi-informasi tentang kelas, hubungan antar kelas, serta hal-hal lain. Ini memang dari sananya, untuk mendukung yang namanya reflection

Bagi saya, reverser, mendapatkan aplikasi .NET itu menyenangkan, karena 99% developer nggak paham bahwa kode mereka bisa ditelanjangi, dilihat bebas. Dengan kata lain kalo developer .net deliver aplikasi, dia sebenarnya memberikan source code juga.

Nah, aplikasi .net ini range-nya cukup lebar. Untuk .net lawas yang sudah mature (sebelum .net core), kita bisa membuat aplikasi desktop dalam bentuk .exe atau .dll, maupun web application (pendekatan webform dkk). Alternatif dari .net microsoft adalah mono, cross platform CLR yang dikembangkan berdasarkan spesifikasi yang sama dengan .net microsoft. Jika pernah memprogram dengan Unity, dia menggunakan mono. Mono ada di linux, windows, mac os x. Namun terakhir si mono dibeli oleh Microsoft sehingga ke depan kemungkinan baik mono dan .net microsoft akan dijadikan satu, bisa dalam bentuk .net core dsb. IDK.

**Untuk reverse .net biasanya ada 2 tools yang kupake:**

**- Telerik JustDecompile dan**

**- dnSpy**

Telerik JustDecompile: https://www.telerik.com/products/decompiler.aspx

dnSpy https://github.com/0xd4d/dnSpy

Sebagai contoh, file berikut :

/\*

Simple C# program Skeleton.

Console.

Compile:

$ csc skeleton.cs

Run:

$ skeleton

\*/

using System;

namespace LearnNetCode

{

class Skeleton

{

static void Main(string[] args)

{

Console.WriteLine("Hello World");

Console.ReadKey();

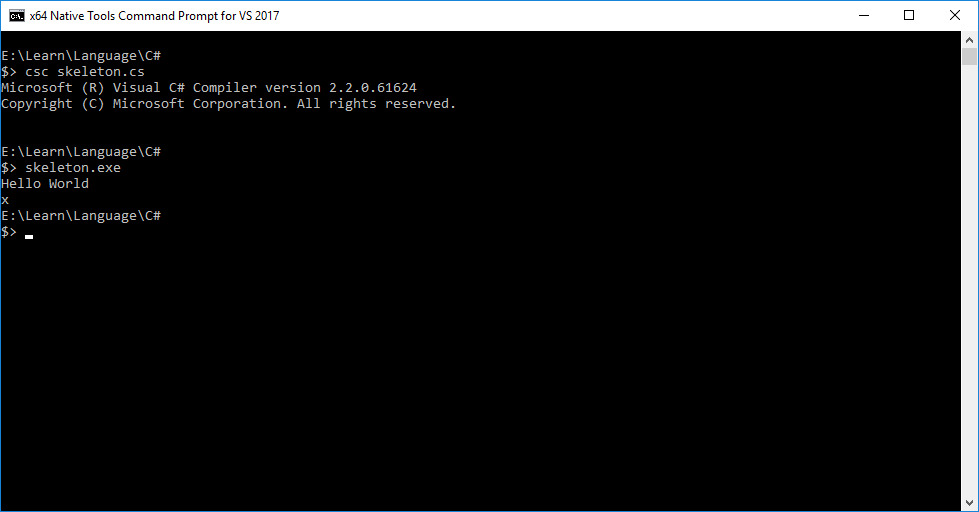
}

}

}

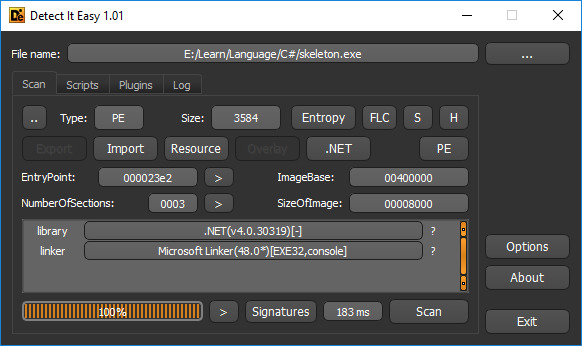
Aku compile dengan command berikut (Visual C# 2017):

csc skeleton.cs

Kemudian dijalankan, akan menghasilkan seperti ini

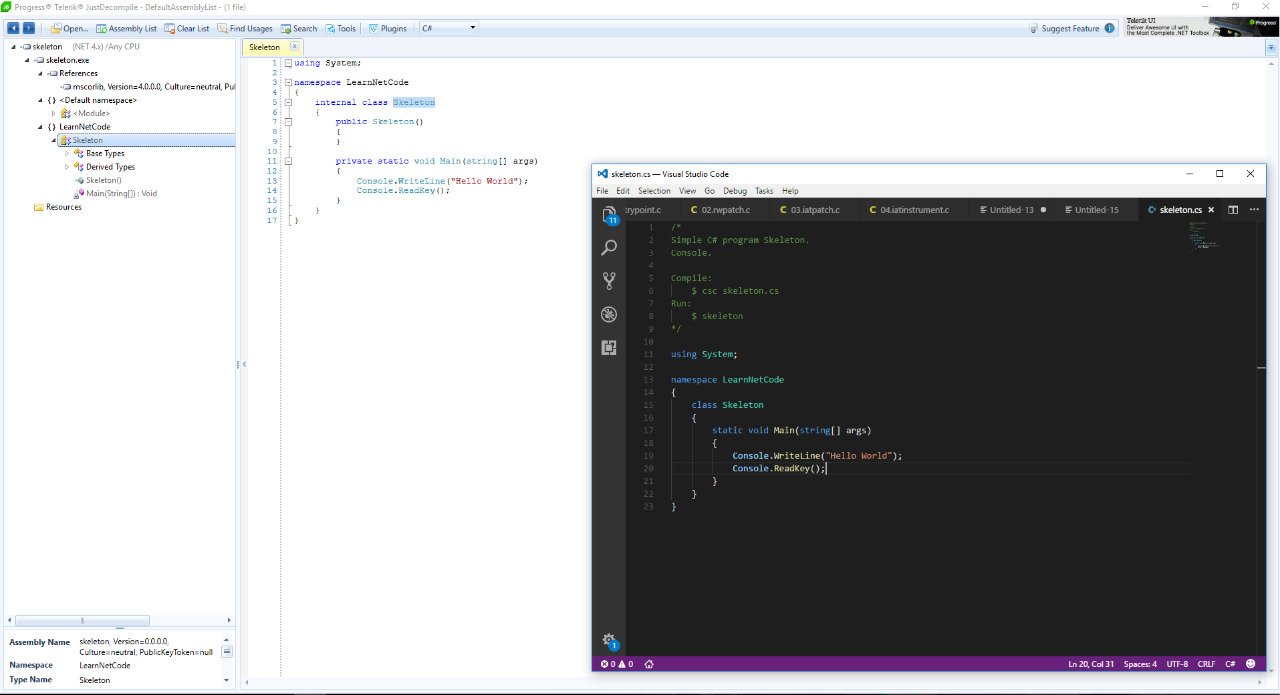
Biasanya, ketika dihadapkan dengan sebuah file executable maupun dll yang belum pernah kuketahui sebelumnya, langkah pertama dalam reversing adalah information gathering terlebih dahulu.

Disebutkan bahwa aplikasi .net memiliki metadata, dia juga punya signature yang membedakan antara .net dengan aplikasi native. Untuk mengetahui hal ini, kita bisa menggunakan aplikasi khusus. Yang menjadi favorit saya adalah DIE (Detect It Easy)



Di sana terlihat jelas bahwa aplikasi ini adalah aplikasi .net

Kita tinggal menelanjangi kodenya.

Kita gunakan Telerik JustDecompile, semudah drag n drop aplikasi .exe ke sana

Itu adalah tampilan hasil decompilation dengan Telerik JustDecomple.

Kita bandingkan dengan source code asli. Kita bisa dapatkan dengan persis, lebih verbose sedikit, kode yang kita tulis, dan ya, informasi yang dibawa oleh metadata ini sampai ke level nama kelas beserta nama method.

**Nah, sebenarnya RE menitikberatkan kepada analisis, dan kemampuan yang paling fundamental adalah kemampuan membaca.**

**Syukurnya** **.net lebih gampang dibaca ,Jadi itu sedikti insight.**

Sekarang kita ke hal yang lebih menantang.

**"Abis itu ngapain?”**

Pertanyaan ini memiliki banyak sekali variasi jawaban. Karena "apa yang kita lakukan selanjutnya" akan bergantung kepada "apa tujuan kita sebenarnya"

Ada beragam motivasi untuk melakukan RE, misalnya:

- mendapatkan cara kerja suatu proses. Misalnya software buatan kompetitor bisa melakukan hal yang jauh dibandingkan hal yang kita capai.

Bagaimana caranya? curi saja kodenya.

- memecahkan proteksi yang ada, supaya bisa kita pake dengan bebas. License checking? serial number? crack saja.

- memodifikasi software. Butuh fitur baru tapi pengembangnya sudah gulung tikar? Entah bikin baru atau tambahkan modul ke aplikasi itu. Jika cara kedua dilakukan, kita melakukan modifikasi si software dengan cara menambahkan kode di suatu tempat yang masih kosong kemudian kita bikin sebuah shortcut sedemikian hingga kita bisa memanggil modul kita tadi.

Motivasi pertama itu so obvious.

Yang menarik adalah motivasi kedua dan ketiga.

Banyak sekali aplikasi komersil menggunakan serial number untuk memproteksi penggunaan. Serial number invalid = nggak bisa install atau nggak bisa pake.

**Tapi apa sebenarnya serial number?**

Cuman sekedar deretan huruf yang akan di-feed ke sebuah / beberapa fungsi pengecekan. Hasil akhirnya adalah: valid ato invalid

**Jadi, bagaimana aku mengakali sebuah pengecekan serial number?**

So far ada 2 kategori besar: patch dan keygenning

**Patching** artinya adalah mengubah satu bagian dari kode dengan hal yang kita inginkan.

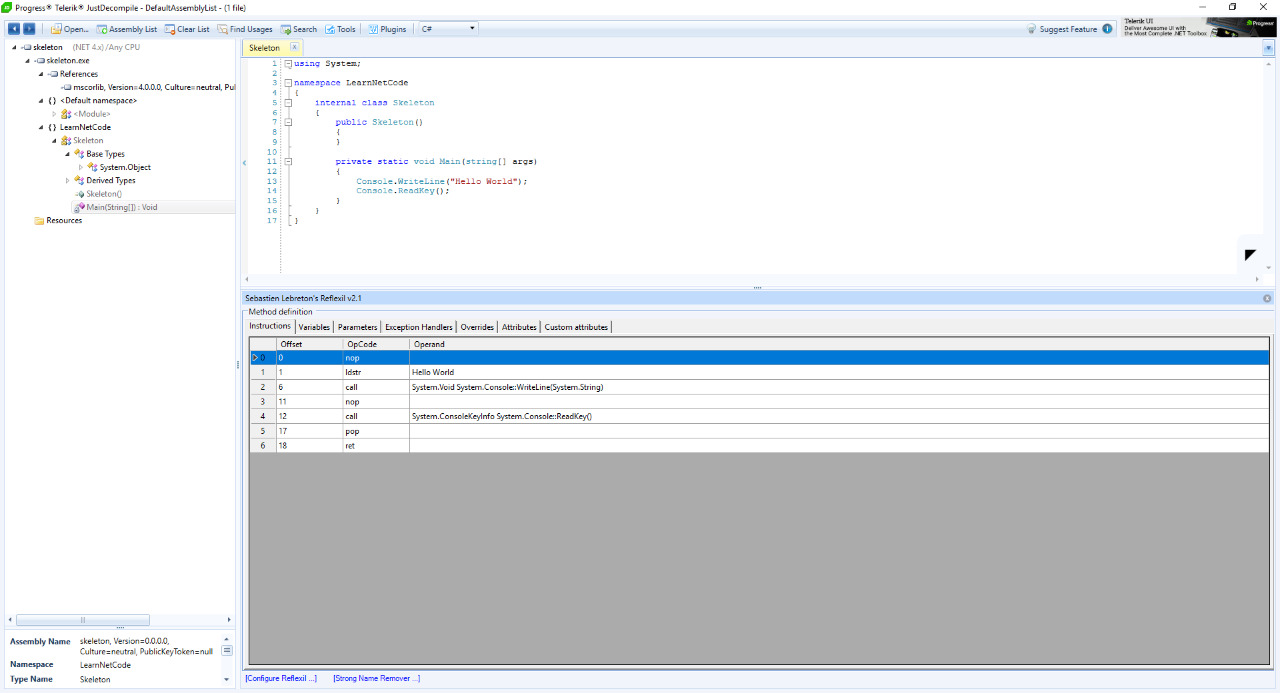
To be short, tujuan kita adalah mengubah fungsi pengecekan untuk menghasilkan keputusan "valid" untuk apapun serial yang kita berikan.

**Bagaimana caranya?** Bisa dengan meng-overwrite keseluruhan fungsi, atau hanya mengoverwrite bagian jump saja.

Untuk overwrite ini kita harus bermain di level yang lebih rendah dibandingkan kode C# hasil decompilasi tadi. Kita mengubah bytecode alias CIL dari aplikasi

Pembahasan ini cukup panjang karena kita harus mengetahui dulu representasi CIL alias bahasa assembly-nya .NET

Kira-kira seperti inilah representasi dari CIL



Itu adalah representasi CIL untuk fungsi Main yang menampilkan Hello World.

Jika dibandingkan antara CIL dan assembly, bahasa CIL lebih mudah dipahami karena CIL sangat mirip dengan representasi tingkat tinggi di C#

Patching nanti "semudah" mengubah opcode dari return "false" menjadi return "true" (contohnya), tapi tidak semua menggunakan satu pengecekan saja.

**Yang lebih menantang adalah, bisakah kita membuat keygen dari serial itu?**

Bisa saja, caranya dengan entah mencomot algoritma itu untuk dibalikkan logic-nya, ato kita bikin suatu algoritma baru yang memecahkan algoritma pengecekan.

Pengecekan serbuah serial number dapat kita anggap sebagai sebuah sistem persamaan yang besar.

Dari persamaan itu kita bisa mencari nilai-nilai yang cocok sehingga serial itu dapat dianggap valid.

**That's, kita bisa membuat sendiri algoritmanya, ato menggunakan sebuah alat bantu bernama theorem prover**

Ada salah satu theorem prover yang menarik dan powerful bernama Z3, dikembangkan oleh Microsoft. Z3 ini dipakai untuk memecahkan sistem persamaan yang sangat besar dan kompleks dengan pendekatan Constraint. Intinya, dikasih constraint, apakah ada sesuatu yang bisa memenuhi ini? jika ada apa contohnya

Contoh yang sederhana ini:

/\*

Simple C# program Skeleton.

Console.

Compile:

$ csc skeleton.cs

Run:

$ skeleton

\*/

using System;

namespace LearnNetCode

{

class Skeleton

{

static bool check(string order, string serial)

{

if (serial.Length != 15)

return false;

if (order.Length != 6)

return false;

if (serial.Substring(0, 3) != "SAP")

return false;

if (Int32.Parse(serial.Substring(3, 4)) < Int32.Parse(serial.Substring(5,4)))

return false;

if (Int32.Parse(serial.Substring(7, 4)) != 1337)

return false;

if (Int32.Parse(order.Substring(2,4)) + 15 != Int32.Parse(serial.Substring(11,4)))

return false;

return true;

}

static void Main(string[] args)

{

Console.Write("Masukkan kode order: ");

string order = Console.ReadLine();

Console.Write("Masukkan serial: ");

string serial = Console.ReadLine();

if (check(order, serial))

{

Console.WriteLine("Serial diterima");

} else {

Console.WriteLine("Serial ditolak");

}

}

}

}

Ini contoh pengecekan yang sangat sederhana, kita bisa membalikkan logika pengecekan ini dengan mudah. Tapi yang penting adalah esensinya, bahwa untuk membuat keygen sebenarnya kita perlu tau bagaimana cara key di-generate dan dicocokkan sehingga kita bisa membuat algoritma untuk membuat key-nya sendiri.

Nah halangan ketika reverse engineering sebuah aplikasi pasti ada. Tapi biasanya hanya mereka yang sudah aware saja menghadapi sebuah aplikasi yang cukup besar, meskipun bisa kita decompile, tetap membutuhkan waktu. Karena itu kita perlu strict ke tujuan kita semula.

**Apa yang ingin kita capai?**

Biasanya dalam pentesting, ada beberapa hal yang ingin kulihat:

- apakah ada hardcoded information. Komponen connection string, misalnya. Atau mungkin credentials.

- apakah ada mekanisme rahasia ketika hendak berhubungan? Misal konek ke port tertentu, kirimkan data, kemudian kirimkan data asli ke port lain.

- bagaimana cara dia menghandle sesuatu? oh ternyata dia nggak pake parameterized buat bikin SQL query, bisa nih di SQL Injection. dsb

Sebagai contoh:

Ada seorang developer yang menyatakan bahwa data di database telah terlindungi dengan baik, tidak akan ada yang bisa dapatkan data itu. Ternyata dia membuat suatu algoritma enkripsi sendiri dengan kunci yang di-hardcoded. Easy.

**Lalu bagaimana cara melindungi dari reverse engineering?**

Not so much. Pada akhirnya Reverse Engineering akan menang, cuman tinggal menunggu waktu saja.

Yang bisa dilakukan adalah mempersulit dan menghambat kerja reverser.

Caranya adalah dengan melakukan obfuscation

**Obfuscation** adalah seni mengaburkan kode dengan tetap membiarkan kode akhir secara semantik melakukan hal yang sama seperti sebelum di-obfuscated.

Karena .net sangat baik dengan memberitahu metadatanya, yang kita lakukan adalah mengubah penamaan kelas atau method di metadata itu sehingga menjadi sesuatu yang tampak random.

Misal kelas FormAbout menjadi aAbcd

Obfuscation ini dilakukan di fase akhir, bukan di source code.

**Salah satu tool untuk obfuscation di .net adalah ConfuserEX**

Tapi ya... dengan waktu dan motivasi yang cukup pada akhirnya Reverser akan tetap menang 😆

Sebenarnya ada banyak sekali hal yang ingin dibahas, tapi kita sudahi pembahasan sampai di sini dulu.

**Kita sudah membahas properti utama dari .net yaitu**

- interpreted language

- mencantumkan metadata

- bisa di-decompile

**RE itu luas. Dalam RE kita bisa**

- mengekstrak informasi dari program (curi kode)

- memecahkan proteksi

- memodifikasi program

Karena .NET bisa di-decompile dengan mudah, maka lakukan obfuscation untuk menghambat kerja reverser

Sekian, dan kalo ada program silahkan kasih ke saya untuk saya bongkar 😂

--------------------------------SESSI TANYA JAWAB-----------------------------------

R 🡪 Satria Ady Pradana : Back to moderator, atau mulai tanya jawab?

R🡪 Moderator : Yoo...Kalo ada yang mau ditanya silahkan

R 🡪 Satria Ady Pradana : Btw, mungkin tahun depan saya bikin buku Reverse Engineering dengan .Net

eh salah, Reverse Engineering .NET Application

Kalo ada sesi lanjutan, mungkin enaknya dibahas tentang hal-hal spesifik misal RE terhadap web application yang dibikin dengan .NET

Bagaimana mencari sesuatu dengan cepat, apa yang dicari, tools yang digunakan, cara menghindari proteksi yang ada, dsb

R 🡪 Hunaepi : Mantapp bukunya langsung saya pesan yahh Om...

R 🡪Satria Ady Pradana : Pertanyan akan dilayani sampe sengantukku wkwkwk

T 🡪Fredy : Ini kalo habis di decompile bisa compile balik ngga ya?

Terus bisa disisipin sesuatu dalam bentuk apapun juga ngga pada saat ngompile balik nya?

J 🡪 Satria Ady Pradana : Most of them bisa

Cek juga di sini: https://www.slideshare.net/satriapradana1/workshop-reverse-engineering-protecting-and-breaking-the-software

Kalo project yang gak terlalu besar, decompile enak, tapi kalo kompleks akan banyak dependensi pas compile dsb jadi akan bergantung kepada situasi yang ada. Tapi theoretically sangat bisa

R 🡪Fredy : Wah.. makin gede makin susah di decompile ya berarti..

Haha.. orang-orang bakalan lari dari .net dunk klo tau gitu?

Trims buat jawaban nya

J 🡪Satria Ady Pradana : Decompile mah gampang, compile ulang itu masalahnya, kan kudu resolve beberapa hal :D

Makanya daripada decompile-modify-compile mending modifikasi di tempat

Jadi si dnSpy atau Telerik JustDecompile bisa dipake buat itu.

T 🡪Jonathan : Hallo, baru sempet baca tadi di jalan.. Mau tanya dong

J 🡪Satria Ady Pradana : Sok atuh

T 🡪 Jonathan : Untuk code yang berpotensi di SQL Inject bukannya tergantung dari gimana cara kita menulis codenya juga ya ?

J 🡪Satria Ady Pradana : Yep. Misal, gak pake filter, gak pake prosedur standard buat sanitasi, dsb.

Kita baca kode hasil dekompilasi buat ngeliat ada apa nggak keteledoran macam itu

R 🡪Jonathan : Berarti sebenernya dari sisi si developernya dong yang harus paham how to writes secure code

J 🡪Satria Ady Pradana : In this case, untuk melindungi dari SQL Injection, ya

T 🡪 Fatur : Mas satria mau tanya dong, ada kemungkinan gak sih, kita naro semacam monitoring di koding kita. Jadi kalo ada yang coba penetrasi. Itu bisa ketahuan.

J 🡪 Satria Ady Pradana : Maksudnya penetrasi ini SQL injection dkk gitu?

Baiknya sih terapkan secure coding dan secure SDLC yang baik. Trus di depan web app pasangin WAF (Web Application Firewall) buat ngecek isi dari request yang dikirimkan apakah possibly sebuah serangan

R 🡪 Fatur : Ok Mas, makasih

J 🡪 Ady Pradana : Sama-sama

R 🡪Juan Caesar : Di nginx ada tuh firewall bisa dterapin

T 🡪 Eko Junaidi Salam : Os yang biasa dipake untuk reversing pake apa ?

J 🡪Satria Ady Pradana : Saya pake windows

T 🡪Eko Junaidi Salam : Btw Om Satria sudah bergelut sebagai reverser berapa lama ?

J 🡪Satria Ady Pradana : Nggak lama-lama amat sih, tingkat akhir kuliah, which is sekitar 3 tahun yang lalu. Tapi buat ctf doang

T 🡪Eko Junaidi Salam : Ctf ? Apa itu ?

J 🡪Om Ronald Simanjuntak : Capture the flag

Lomba cepat cari akses

J 🡪Satria Ady Pradana : Iya, capture the flag

Ada challenge, kudu solve, titik akhirnya nanti dapat token yang disebut flag kalo udah solve. Submit buat point. Paling banyak poin yang menang (agregasi dari banyak challenge)

Satria Ady Pradana : fb ada, tapi aku lebih sering ke telegram

<https://t.me/ReversingID>

Moderator : Thanks kulwapnya semalem yang super sekali by professor of LOLI

#eeehhh

@Satria Ady Pradana ...

Semoga bisa bermanfaat untuk semua