

كيف احترف اصعب المجالات في الامن السيبراني مثل:

- Malware analysis / Development & Reverse Engineering
- Exploit development
- Security research
- build security solutions for Windows.
- forensic investigations

لا يزال اي كمبيوتر شخصي الذي يعمل بنظام Windows هو جهاز اساسي لا ي المستخدم او حتى في المؤسسات الصغيرة والكبيرة. نظراً لانتشاره في كل مكان، يظل سطح مكتب Windows هو الهدف المفضل للمهاجمين لاختراق المؤسسة، ووصولهم الى الهدف. سواء كنت تقوم بتحليل البرامج الضارة، او إجراء أبحاث أمنية، او إجراء تحقيقات جنائية، او المشاركة في محاكاة الخصوم مثل تطوير او محاكاة برمجية خبيثة او تطوير استغلال لثغرة معينة، او إنشاء حلول أمنية لنظام التشغيل Windows، فإن فهم كيفية عمل Windows داخلياً يعد أمراً بالغ الأهمية لتكون فعالة في مهمتك.

نظراً لأن نظام تشغيل مثل ويندوز شائع جداً، فمهما نوعاً ما ليس سهل، أن شاء الله في هذا المقال رح أوضح ايش ممكن تتعلم في هذا النظام سواء كنت محل او مطور برامج خبيثة او باحث امني او كاتب استغلال او انشاء حلول امنية لنظام الويندوز فإن فهم النظام التشغيل لا بد منه.

اول خطوة حتى تبدأ في أحد من هذه المجالات ان تفهم النظام التشغيل والبنية التحتية لنظام ويندوز. ماذا اقصد في البنية التحتية لنظام ويندوز؟ يا صديقي رکز معی منیج، جیب ورقه و قلم و تعال.

نظام التشغيل Windows يتقسم إلى جزئين أو قسمين. القسم الأول User Mode والقسم الثاني Kernel Mode جميل جداً. هكذا يتبيّن علينا ما الذي يجب أن نتعلّمه أولاً. لذا نأخذ نظره أو فكره عن User Mode و Kernel Mode.

تتخد النواة او Kernel القرارات الأمنية التي تحدد ما يمكن للمستخدم القيام به على النظام. ومع ذلك، فإن معظم التطبيقات التي تستخدمها على جهاز يعمل بنظام التشغيل Windows تعمل في وضع المستخدم User Mode.

يعني التطبيقات مثل المتصفحات او برامج مايكروسوفت تعمل في وضع المستخدم User، وتعمل مكونات نظام التشغيل في وضع kernel. عندما يريد أحد التطبيقات إنجاز مهمة ما، مثل إنشاء ملف، فإنه لا يمكنه القيام بذلك بمفرده. الشيء الوحيد الذي يمكنه إكمال المهمة هو النواة Kernel، لذا بدلاً من ذلك يجب على التطبيقات أن تتصل أو تستدعي وظيفة محددة من Kernel.

جميل جداً، هذا يعني لتشغيل التطبيق او البرنامج لا بد من الاتصال مع Kernel هذا يعني اذا هدفك تحل تطبيق لإيجاد ثغره امنيه او تحليل برمجيه خبيثه يجب عليك أن تفهم التطبيق كيف يعمل على النظام، هذا يعني يجب عليك فهم التطبيق او البرنامج من جانبيين اثنين الجانب الأول وهو User Mode والجانب الثاني Kernel Mode وكلاهما مع بعض ينفذ التطبيق يندرج تحت اسم System Architecture. ومن منظور المهاجم او المخترق يوجد استغلالات على كلا الجانبين أن كان على مستوى User Mode او Kernel Mode.

ما الذي يجب على ان اتعلم؟ لنبدأ مع Windows Internal User Mode او

- فهم هي المبادئ الأساسية وراء تصميم وتنفيذ نظام التشغيل Windows. والوظائف التي يوفرها.
- افهم الوظائف التي يوفرها Windows والتي تحدد التطبيقات والخدمات.
- فهم المراقب الموجودة في النظام والتي يتم إساءة استخدامها بشكل شائع بواسطة البرامج الضارة
- افهم إجراءات الأمان المتوفرة في نظام التشغيل Windows والتي ترفع مستوى الحماية ضد عمليات الاستغلال والبرامج الضارة.

هيك اشياء ازا فهمتها رح تكون قادر او اكثر فعاليه في تحليل البرامج الضارة على أنظمة Windows و أكثر فعالية في التحليل الجنائي. و فهم كيف ممكن يتم إساءة استخدام الوظائف التي يوفرها النظام بواسطة البرامج الضارة.

اهم المواضيع على مستوى User Mode :

- System Architecture
- User Mode Execution
- Memory Management
- PE Files
- Objects and Handles
- Security
- Services Infrastructure
- Security Mitigations

في البداية : System Architecture

في البدايه عند تعلم Kernel Mode وكذلك System Architecture سوف تتعلم في البدايه عن System، كما ذكرنا عندما تريه تنفيذ برنامج معين في وضع المستخدم User يجب اي يتصل مع Kernel لتحديد الوظيفه، عند التكلم على System اي يعني سوف تعرف كيف يتم التواصل بين بعض وماهي المكونات في Kernel Mode و User Mode و Architecture.

ثم : User Mode Execution

من ثم تعرف على كيفية تنفيذ التعليمات البرمجية او البرامج في وضع المستخدم User Mode Execution . عند تشغيل برنامج معين او لعبه معينه يتم إنشاء عملية في النظام، يمكن بدء العملية إما بواسطة المستخدم أو بواسطة النظام نفسه. تستهلك العملية موارد مثل الذاكرة ومساحة القرص المهمة. وكل عملية اسم، وهو اسم البرنامج الذي تم إنشاؤها منه. لتعريف العملية بشكل فريد، يتم منح كل عملية معرفًا فريدًا يسمى معرف العملية، أو PID، بواسطة نظام التشغيل. PID هو معرف/رقم يتم تعبيئه عشوائياً، ويتغير في كل مرة يتم فيها تنفيذ البرنامج حتى على نفس النظام. يتم تخزين البرنامج على القرص الصلب، ولكن لكي تتمكن وحدة المعالجة المركزية او CPU من تنفيذ تعليمات التعليمات البرمجية في البرنامج، يقوم نظام التشغيل أولاً بتحميل البرنامج في ذاكرة الوصول العشوائي (RAM)، وبالتالي إنشاء عملية. تلقط وحدة المعالجة المركزية CPU تعليمات البرنامج من ذاكرة الوصول العشوائي (RAM) وتتفذ هذه التعليمات. عليك اهم مواضيع User Mode Execution .

- Processes
- Threads
- Data structures (PEB & TEB)
- System process
- Process and thread data structures
- Process hierarchy
- Host processes
- Execution vectors
- Jobs

ثم : Memory Management

فهم مساحة العنوان الافتراضية للعملية ومكوناتها. كما ذكرنا تستهلك العمليات موارد مثل الذاكر، حتى يتمكن وحدة معالج المركزية CPU تنفيذ التعليمات البرمجية. وكل عملية يتم إنشاء لها ذاكر افتراضية. اهم المواضيع:

- Process address space

- Address space layout randomization (ASLR)
- Memory protection
- Data execution prevention (DEP)
- Memory-mapped files
- Thread stacks
- Heaps

ث : PE Files

حتى نفهم كيف يتم تحميل البرنامج من قرص الصلب إلى الذاكرة العشوائية لتنفيذ التعليمات البرمجية من قبل CPU يجب عليك دراسة مواضع يجب النظر إلى PE Files .

كيف يقوم Windows loader بتحميل البرنامج من القرص الصلب إلى الذاكرة وتحويله إلى عملية؟ .
كيف يعرف Windows loader تحميل البرنامج كعملية، وكيف يعرف حجم الذاكرة الافتراضية التي يحتاجها، وأين يوجد الكود والبيانات في البرنامج، وأين في الذاكرة الافتراضية لنسخ هذا الكود والبيانات؟ هنا يأتي دور PE File ، يحدد تنسيق ملف PE الـ Headers المختلفة التي تحدد بنية الملف ورمزه وبياناته والموارد المتنوعة التي يحتاجها.

كما أنه يحتوي أيضًا على حقول متنوعة توضح مقدار الذاكرة الافتراضية التي يحتاجها عندما يتم نشرها في عملية وأين يمكن في ذاكرة العملية نسخ التعليمات البرمجية والبيانات والموارد المتنوعة الخاصة بها. بمعنى آخر، ملف PE عبارة عن بنية بيانات تحتوي على المعلومات الضرورية لحمل نظام التشغيل حتى يتمكن من تحميل الملف القابل للتنفيذ في الذاكرة وتنفيذه. اهم المواضيع:

- PE files layout
- Data directories
- Debug directory
- CLR Header
- Export directory
- Import directory
- Relocations
- Loader functionality
- DLL load callbacks

من ثم Object and Handles

فهم الأشياء والتعامل مع الجداول. في أنظمة التشغيل المشابهة لنظام Unix، كل شيء عبارة عن ملف. في نظام التشغيل Windows، كل شيء عبارة عن كائن، مما يعني أن كل File و Process و Thread يتم تمثيله في ذاكرة kernel كبنية كائن. ومن المهم بالنسبة للأمان، أن هذه الكائنات يمكن أن يكون لها واصف أمان معين، والذي يقييد المستخدمين الذين يمكنهم الوصول إلى الكائن ويحدد نوع الوصول الذي لديهم (على سبيل المثال، القراءة أو الكتابة).

كيف يمكننا الوصول إلى الكائنات من تطبيق وضع المستخدم؟ حسناً، إذا كنا في تطبيق وضع المستخدم، فسنحتاج إلى النواة للوصول إلى الكائن، ويمكننا استدعاء كود وضع kernel في تطبيق وضع المستخدم باستخدام واجهة استدعاء النظام system call interface . تقوم معظم استدعاءات النظام بتنفيذ بعض العمليات على نوع معين من كائنات kernel التي يعرضها مدير الكائنات Object Manager . على سبيل المثال، يقوم استدعاء النظام وظيفه NtCreateMutant بإنشاء كائن Mutant. اهم المواضيع:

- Windows Objects
- Object namespace
- Object headers
- Process handle tables
- Handle duplication
- Object callbacks
- Object reference counting

- Type objects

: Security من ثم

تعرف على كيفية قيام النواة بتأمين الوصول إلى الكائنات . كما ذكرنا في نظام التشغيل Windows، كل شيء عبارة عن كائن، مما يعني أن كل File و Process و Thread يتم تمثيله في ذاكرة kernel كبنية كائن. ومن المهم بالنسبة للأمان، أن هذه الكائنات يمكن أن يكون لها واصف أمان معين، والذي يقييد المستخدمين الذين يمكنهم الوصول إلى الكائن ويحدد نوع الوصول الذي لديهم (على سبيل المثال، القراءة أو الكتابة). اهم المواضيع:

- Tokens
- Security identifiers (SID)
- Privileges
- Security descriptors
- Access control lists
- Integrity levels
- Access checks
- Impersonation

: Services Infrastructure من ثم
فهم كيفية عمل الخدمات في Windows واهم المواضيع:

- Service architecture
- Service control manager
- Service configuration
- SVCHost.exe
- Service SIDs
- Trusted Installer
- Logon sessions
- Session isolation

:Security Mitigations من ثم

فهم وسائل التخفيض الأمنية المختلفة التي تمت إضافتها لحماية التطبيقات على مر السنين. واهم المواضيع:

- Process mitigation policies
- Exploit guard
- Arbitrary code guard (ACG)
- Control flow guard (CFG)
- Enhanced control flow guard (xFG)

- Control-flow enforcement technology (CET)
- Win32K API call filtering

اعلم انه كثير ، لا تشعر في الاحتياط يا صديقي انت هنا لكي تتعلم وكل شي مع الوقت كل يلي مطلوب منك فقط التعلم و الوقت.

هذا كان بالنسبة لمستوى User Mode .

الآن مع Kernel Mode اهم المواضيع على مستوى Kernel Mode

النواة هي الجزء الأساسي من نظام التشغيل الذي يدير موارد النظام ويوفر الخدمات للبرامج التي تعمل في وضع المستخدم. يتمتع الكود الذي يتم تشغيله في وضع kernel بـإمكانية الوصول المباشر إلى أجهزة النظام وذاكرته. غالباً ما تتطلب برامج الأمان مستوى عميقاً من الوصول إلى نظام التشغيل لمراقبة التهديدات الأمنية والاستجابة لها بشكل فعال. هذا هو المكان الذي يلعب فيه وضع kernel

ما الذي يجب علي ان اتعلم؟

- فهم المبادئ الأساسية وراء تصميم وتنفيذ Windows kernel .
- فهم المكونات الرئيسية في Windows Kernel والوظائف التي توفرها.
- تكون قادراً على فحص هيكل بيانات النظام باستخدام مصحح أخطاء kernel وتفسير مخرجات أوامر مصحح الأخطاء.
- تكون قادراً على التنقل بين هيكل البيانات المختلفة في النواة باستخدام أوامر مصحح الأخطاء.
- كن قادراً على تحديد مؤشرات الاختراق أثناء البحث عن البرامج الضارة في وضع kernel .
- أن تكون قادراً على إجراء تحليل الطب الشرعي لنواة Windows .
- فهم كيفية تفاعل الجذور الخفية rootkits في وضع kernel والحلول التجارية لمكافحة البرامج الضارة مع النظام.

واهم المواضيع:

- Kernel Architecture
- Processes and Threads
- System Mechanisms
- Execution Contexts
- Synchronization
- Memory Management
- I/O Management
- Kernel Security Mitigations

البرمجة

أين البرمجة من كل هذا؟ أحسنت سؤال جميل ، الفكرة هنا ليس في البرمجة فقط، البرمجة عبارة عن أشياء أو مكونات متاحة من النظام استخدمها لإتمام شيء معين، كما ذكرت في اعلا المقال فهم المراقب الموجود في النظام والتي يتم إساءة استخدامها بشكل شائع بواسطة البرامج الضارة، كيف رح تبرمجها وانت لا تعرف عنها ولا تعرف كيف تستخدمنا، الأن بعد ما تدرست نظام التشغيل من الجانب الأمن فقط يتبقى عليك ان توظف الأشياء في مجالك حتى توضف المراقب الموجود مع النظام التشغيلي. فهمت الفكرة ؟

اهم اللغات:

لعبة C :

تحتاج أولاً إلى تعلم لغة البرمجة C ، لأن معظم البرامج او البرامج الضارة التي ستقوم تحليلها لإيجاد ثغرة او لتحليل سلوكها ان كانت خبيثة، لغة C تستخدم مفهوم المؤشرات كثيراً من خلال كود المصدر الخاص بها، ومن خلال التعرف على المؤشرات، ستكون مهمتك في تحليل البرامج الشرعية او الضارة أسهل بكثير . لذلك، فإن الكثير من وثائق واجهة برمجة التطبيقات (API) تتاح بشكل كبير حول لغة C. سيكون من الأسهل عليك قراءة الوثائق والأدلة إذا كنت تعرف لغة C. لذلك ستكتسب الكثير من خلال تعلم لغة C. إذا كنت تعرف لغة عالية المستوى مثل Java أو C#. لا تزال بحاجة إلى تعلم لغة C ، لأن اللغات عالية المستوى مثل Java و Python وما إلى ذلك، تخفي بعض جوانب البرمجة ذات المستوى المنخفض مثل المؤشرات وتخصيص الذاكرة. وعندما تقوم بتحليل البرمجيات، فإنك تحتاج إلى التعامل مع هذا الجزء من البرمجة ذات المستوى المنخفض. الجانب الأكثر أهمية في تعلم لغة برمجة C هو المؤشرات، والتعامل مع الهيكل، والذكرة، وكيفية عملها. يجب عليك إتقانها، لأنك ستحتاج إليها في "تحليل البرامج او البرامج الضارة".

ـ لغة Python :

تعد لغة Python لغة شائعة لمختلف الثغرات نظراً لبساطتها وتعدد استخداماتها ونظامها البيئي الغني بالمكتبات. ربما تحتاج إلى أنتهت بعض الأشياء البسيطة لاختصار الوقت ، أو التعامل مع البرامج الضارة لتعلم كما تريده، أو تطوير ملحق لإحدى أدوات تحليل البرامج الضارة لديك. بايثون هي اللغة التي تختارها. هناك لغات برمجة نصية أخرى مثل Python أيضاً، ولكن تكامل Python في التطبيقات الأخرى (مثل IDA pro) و العدد الهائل من المكتبات التي تهدف بشكل واضح إلى الهندسة العكسية [1] يجعلها الخيار الأفضل.

يمكن استخدام بايثون لتحليل الملفات الثنائية، مثل الملفات التنفيذية والمكتبات. تحظى مكتبة pwntools بشعبية كبيرة في مجتمع تطوير استغلال البرامج للتفاعل مع الثنائيات، وصياغة الحمولات، والتواصل مع الخدمات عن بعد.

ـ Assembly لغة :

بعد فهم لغة التجميع أمرًا بالغ الأهمية ، خاصة في مجالات مثل الهندسة العكسية وتحليل الثغرات الأمنية وتطوير الاستغلال. بعض الوصاف:

ـ الهندسة العكسية:

لغة التجميع هي تمثيل منخفض المستوى لرمز الآلة وهي ضرورية للملفات التنفيذية الثنائية للهندسة العكسية. يستخدم متخصصو الأمان التجميع لتحليل وظائف البرامج وفهم كيفية عملها وتحديد نقاط الضعف المحتملة .

ـ تحليل البرامج الضارة:

غالباً ما تأتي البرامج الضارة في شكل ملفات تنفيذية ثنائية، ويطلب تحليلها معرفة لغة التجميع. يقوم المهندسون العكسيون بفك رمز التجميع ودراسته للكشف عن سلوك البرامج الضارة، واستخراج مؤشرات التسوية، وتطوير الإجراءات المضادة.

ـ تطوير الاستغلال:

يحتاج مطورو برامج استغلال الثغرات إلى فهم لغة التجميع لتحديد نقاط الضعف في البرامج واستغلالها. يقومون بتحليل كود تجميع البرنامج المستهدف لصياغة الحمولات وإنشاء كود القشرة وتطوير عمليات استغلال تستفيد من العيوب الأمنية.

تحليل الضعف:

يقوم متخصصو الأمان بتحليل البرامج بحثاً عن نقاط الضعف، ويساعد فهم التجميع في تحديد المشكلات الأمنية المحتملة من خلال فحص التعليمات البرمجية للممارسات غير الآمنة أو الخوارزميات الضعيفة أو التكوينات غير الآمنة.

الهندسة العسكرية

- كيف تعرف الشئ كيف يعمل؟ كيف تعرف البنية التحتية لشئ ما؟ ما الطريقة والسلوك الذي يعمل بها؟
- كيف يتم تحليل البرمجيات الخبيثة؟ كيف يتم كشف اكواد البرمجية الخبيثة؟ ما هي اساليب وأنواع وادوات التحليل؟
- كيف نجاوب على هذه الاستلة يجب ان ننطرق الى مجال الهندسة العسكرية

الهندسة العسكرية هي عملية تفكير الشئ لمعرفة كيف يعمل. يتم إجراؤه في المقام الأول لتحليل واكتساب المعرفة حول الطريقة التي يعمل بها شيء ما. يمكن إجراء هندسة عسكرية للعديد من الأشياء ، بما في ذلك البرامج والآلات الفيزيائية والتكنولوجيا العسكرية وحتى الوظائف البيولوجية المتعلقة بكيفية عمل الجينات.

ل الهندسة العسكرية و البرامج الضارة

في تحليل البرمجيات الخبيثة يوجد نوعين من التحليل **static & dynamic**. عندما نتحدث عن تحليل عينات البرامج الضارة بشكل ثابت وдинاميكي. من خلال تقنيات التحليل ، قد نتمكن من الاستنتاج إذا كان ملف العينة عبارة عن برنامج ضار أم لا.

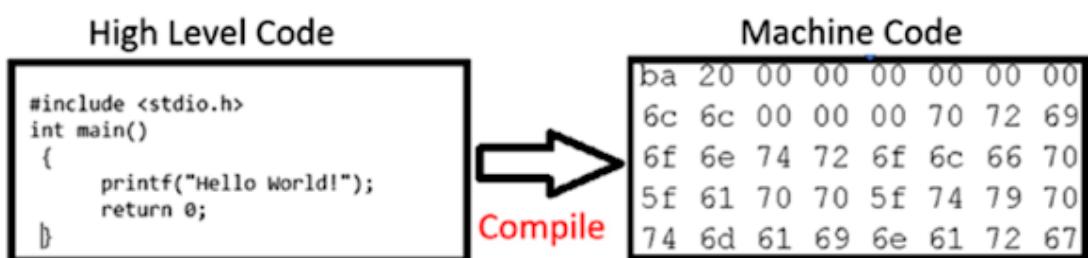
على سبيل المثال ، قد ترغب في تنفيذ برنامج فك تشفير الملفات المشفرة بواسطة برنامج الفدية الضارة. لكن **كيف يمكنك فعل ذلك؟** كيف تقدم خوارزمية فك التشفير أو بعبارة أخرى عكس خوارزمية التشفير المستخدمة بواسطة برامج الفدية؟ مرة أخرى نقف على نفس السؤال. أين نجد الكود الذي تستخدمه البرامج الضارة / برامج الفدية لتفسير الملفات؟

بلطبع لن يسلم مؤلف البرامج الضارة اكواد البرمجية البرامج الضارة إلينا. كل ما لدينا هو ملف ثانوي قابل للتنفيذ. وهنا يأتي دور الهندسة العسكرية ، والتي من خلالها يمكننا تشريح البرامج الضارة وفهم كيفية برمجتها.

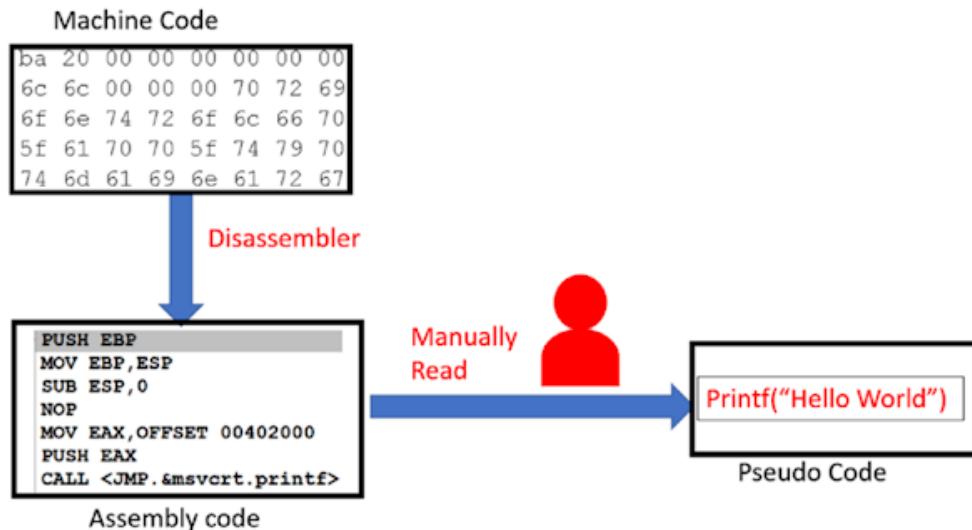
عكس وتفكيرك : (لا ينطبق فقط تحت تحليل الماكوير، كذلك ينطبق لمجالات أخرى مثل تحليل الضعف اي عكس وفك البرنامج حتى موadge فيه نقاط ضعف).

يتم إنشاء الملفات القابلة للتنفيذ كنتيجة لتجميع لغات ذات مستوى أعلى مثل C و VBA وما إلى ذلك ، باستخدام مترجم (Compiler). المبرمجون ، بما في ذلك مبرمجو البرامج الضارة ، يكتبون البرامج والبرامج الضارة في الغالب باستخدام لغة عالية المستوى مثل C و C ++ و Java وما إلى ذلك ، والتي يقومون بعد ذلك بتجميعها باستخدام مترجم (Compiler) لإنشاء ملفات قابلة للتنفيذ.

يحتوي الملف القابل للتنفيذ الذي تم إنشاؤه بواسطة المترجم (Compiler) على رمز الآلة (Machine Code) الذي يمكن للمعالج فهمه. بمعنى آخر ، يحتوي رمز الجهاز على تعليمات يمكن تفسيرها وتنفيذها بواسطة وحدة المعالجة المركزية (CPU).



نظرًا لأنه من الصعب ، إن لم يكن من المستحيل ، فهم وظيفة البرنامج الضار أو الملف القابل للتنفيذ من خلال النظر إلى رمز الجهاز هذا (Machine Code) ، فإننا نستخدم الهندسة العكسية ، لفهم نية الكود. لمساعدتنا في هذه العملية ، لدينا أدوات مختلفة مثل أدوات التفكيك (disassemblers) ، والتي تستهلك رمز الآلة وتحولها إلى تنسيق أكثر قابلية للقراءة من قبل الإنسان في لغة التجميع (Assembly) ، والتي يمكننا بعد ذلك قراءتها لفهم وظائف ونية الملف التنفيذي الذي نعكسه .



يستخدم مهندسو عكس البرامج الضارة أيضًا أدوات أخرى مثل decompilers لتحويل رمز الجهاز إلى تنسيق رمز زائف للغة عالي المستوى يسهل قراءته. ومن الأمثلة الجيدة على برامج التفكيك هذه ، أداة فك تشفير Hex-Rays التي تأتي مع IDA Pro ، و.x64Dbg debuggers ، والتي تأتي مدمجة مع Sandman decompiler

لكن الأداة الرئيسية المشاركة في عملية الانعكاس لا تزال هي المفكك (disassemblers) الذي يحول الكود إلى لغة تجميع (Assembly). لذلك ، لكي تكون مهندسًا عكسياً جيداً ، فإن الفهم الشامل للغة التجميع (Assembly) وتركيباتها المختلفة أمر مهم ، إلى جانب القدرة على استخدام أدوات التفكيك والتصحيح المختلفة (disassemblers & debugging tools) ..

بعد كل هذا، انت جاهز لتأخذ دوره تتكلم في مجال Malware analysis او اي دوره تتكلم في مجال Revese Engineering او اي دوره تتكلم عن Exploit development او اي دوره تتكلم عن Malware development.

لأن كل هذا الان يعتبر فقط تكتيكات وطرق.

اتقان مجال مثل RE او الهندسه العكسيه رح يفتح لك أبواب كثير، مثل إجاد ثغرات ZeroDays لأن انت عندك القدرة على فك وتحليل سلوك البرامج مما تزيد فرصه إجاد الثغرات.

ذلك يفتح لك مجال تحليل البرمجيات الخبيثة، لأن رح تكون بشكل close source، اي يعني ليس لدينا وصول إلى مصدر الكود الأصلي للماлоير، هذا يعني عليك ان تجد طريقه للوصول إلى كود المالوير، لكن كيف؟ هنا يأتي ال reverse engineering اي عكس المالوير حتى نقراء المحتوى، لكن لغه machine هي فقط 01 يعني binary من الصعب على الإنسان أن يقراءها او يفهمها، لهيك في شي اسمو disassemble، يعني لأن 01 مش مفهوم، هيك احنا رح نقراء الكود على شكل assembly ونبدأ نحل سلوكه حتى نجد مؤشرات او سلوك مرrib او خبيث.