1. отслеживание всех запускаемых процессов, каждый запущенный и запускаемый процесс должен проверяться;

*procs = [proc for* proc *in* psutil.process\_iter()

1. проверка цифровой подписи всех исполняемых файлов, используемых процессами;

*def* check\_sign(path):  
 *try*:  
 *with* open(path, "rb") *as* f:  
 pefile = signify.authenticode.signed\_pe.SignedPEFile(f)  
 status, err = pefile.explain\_verify()  
  
 *return* status == signify.authenticode.signed\_pe.AuthenticodeVerificationResult.OK, err  
 *except* FileNotFoundError:  
 *return False*, "Unknown"  
 *except* Exception *as* err:  
 *return False*, err

1. проверка исполняемых файлов на наличие признаков упаковки (например, с использованием утилиты Peid);

packer = pypackerdetect.PyPackerDetect()  
  
  
*def* check\_packer(path):  
 *if not* os.path.exists(path):  
 *return* ""  
  
 *try*:  
 result = packer.detect(path)  
 *return* result["detections"]  
 *except* Exception *as* err:  
 *return* str(err)

1. проверка атрибутов секций, выявление секций одновременно доступных для записи и исполнения (WX);

*def* \_\_analyze\_section(section):  
 *"""  
 https://blog.kowalczyk.info/articles/pefileformat.html  
 0x00000020 Code section  
 0x00000040 Initialized data section  
 0x00000080 Uninitialized data section  
 0x04000000 Section cannot be cached  
 0x08000000 Section is not pageable  
 0x10000000 Section is shared  
 0x20000000 Executable section  
 0x40000000 Readable section  
 0x80000000 Writable section  
 """* r, w, x = *False*, *False*, *False  
 if* (section.Characteristics & 0x00000020) *or* (section.Characteristics & 0x20000000):  
 x = *True  
 if* section.Characteristics & 0x80000000:  
 w = *True  
 if* section.Characteristics & 0x40000000:  
 r = *True* result = ""  
 result += "R" *if* r *else* "-"  
 result += "W" *if* w *else* "-"  
 result += "X" *if* x *else* "-"  
  
 *return* result  
  
  
*def* analyze\_pe\_file(filepath):  
 *try*:  
 pe = pefile.PE(filepath)  
 result\_struct = {}  
 *for* section *in* pe.sections:  
 result\_struct[section.Misc\_PhysicalAddress] = \_\_analyze\_section(section)  
  
 *return* result\_struct  
 *except* Exception *as* err:  
 *return* {}

1. сопоставление кода в памяти процессов с соответствующими исполняемыми файлами, выявление участков исполняемого кода, отличных от исходных файлов в файловой системе;

??????

1. для подозрительных исполняемых файлов, с отсутствующей цифровой подписью осуществляется поиск сигнатур техник из MITRE ATT&CK.

??????

1. для всех процессов отслеживать сетевые соединения и отображать информацию об IP-адресе (IP ASN, ассоциированное доменное имя, информация о провайдере и др.) стороннего узла, с которым происходит взаимодействие;

net\_connections= psutil.net\_connections(kind='inet')

1. вывод для каждого процесса значения, отображающего вероятность его потенциальной вредоносности на основе собранной информации.

\*аналитика предыдущих пунктов\*