Windows 10 IDS – Full Command Cheat Sheet

For: scapy, psutil, pywin32 (Event Log), watchdog, pydivert, pyshark

SCAPY (packet sniffing / crafting)

Command (one line)	What it does
from scapy.all import *	Import scapy core (layers, sniff, sr, send).
conf.iface	Show current capture interface.
conf.iface = 'Ethernet'	Set capture interface (Windows name).
sniff(count=10)	Capture 10 packets and return list.
sniff(filter='tcp', timeout=5)	Capture TCP packets for 5 seconds (BPF filter).
sniff(prn=lambda p:p.summary())	Print one-line summary per captured packet.
sniff(stop_filter=lambda p: IP in p and p[IP].src=='1.2.3.4')	Stop when packet from specific IP arrives.
sniff(store=False)	Stream packets without storing in memory.
sniff(lfilter=lambda p: TCP in p and p[TCP].flags=='S')	Python-level filter: TCP SYN packets.
ls()	List all available layers/fields.
ls(IP)	List fields of IP layer.
lsc()	List Scapy commands/helpers.
pkt.summary()	One-line human summary of a packet.
pkt.show()	Verbose decode of packet layers/fields.
hexdump(pkt)	Hexdump of packet bytes.
Raw(pkt.load)	Access raw payload (if present).
IP(dst='1.1.1.1')/TCP(dport=80,flags='S')	Build TCP SYN packet to 1.1.1.1:80.
send(IP(dst='1.1.1.1')/ICMP())	Send IP/ICMP packet at Layer 3.
sendp(Ether()/IP(dst='1.1.1.1')/ICMP())	Send at Layer 2 (requires iface).
sr(IP(dst='1.1.1.1')/ICMP())	Send/receive; returns answered/unanswered pairs.
sr1(IP(dst='1.1.1.1')/ICMP())	Send and wait for a single response.
srp(Ether(dst='ff:ff:ff:ff:ff:ff')/ARP(pdst='192.168.1.0/24'))	Layer2 send/recv; ARP scan subnet.
srp1(Ether()/ARP(pdst='192.168.1.1'))	Layer2 send and wait one reply.
ARP(pdst='192.168.1.1')	Construct ARP who-has request to target.
DNS(rd=1,qd=DNSQR(qname='example.com'))	Build DNS query record.
IP(dst='8.8.8.8')/UDP(dport=53)/DNS(...)	DNS over UDP packet skeleton.
TCP(flags='S')	TCP SYN flag (use 'SA','PA','FA','R' etc).
IP.ttl=64	Set IP TTL field.
pkt[IP].src, pkt[IP].dst	Read source/destination IP fields.
pkt[TCP].sport, pkt[TCP].dport	Read TCP source/destination ports.
wrpcap('out.pcap', pkts)	Write packets to PCAP file.
rdpcap('in.pcap')	Read packets from PCAP file.

split_layers(IP, TCP)	Disable automatic dissection link between layers.
bind_layers(TCP, TLS)	Bind custom/extra layer after TCP.
Dot1Q(vlan=10)/IP()/UDP()	Add 802.1Q VLAN tag.
IPv6(dst='::1')/ICMPv6EchoRequest()	Basic IPv6 ICMP echo request.
fragment(IP(dst='1.1.1.1')/UDP()/Raw('X'*4000))	Create IP fragments from large payload.
defragment(pkts)	Defragment list of IP fragments (scapy.utils).
sniff(iface='Ethernet', filter='port 53')	Capture DNS traffic on given interface.
PcapWriter('out.pcap', append=True, sync=True)	Stream-write packets to PCAP efficiently.
pkt.time	Epoch timestamp of captured packet.
TCPClient.tcplink(TCP, '1.1.1.1', 80)	Simple TCP client using scapy (optional helper).
traceroute(['1.1.1.1'])	Run basic traceroute (requires privileges).
arping('192.168.1.0/24')	Send ARP who-has to discover hosts.
sniff(offline='file.pcap')	Analyze packets from PCAP file.
sniff(count=0, timeout=10)	Capture for time window without count limit.
sniff(prn=lambda p: p[IP].len if IP in p else None)	Extract a specific field per packet.
NoPayload in pkt	Check if packet has no further payload.
ICMP(type=8, code=0)	Echo request type/code.

PSUTIL (processes / system / network)

Command (one line)	What it does
import psutil as ps	Import psutil.
ps.cpu_percent(interval=1)	CPU usage percent over 1s sample.
ps.cpu_count(logical=True)	Logical CPU count.
ps.cpu_freq().current	Current CPU frequency (MHz).
ps.virtual_memory().percent	RAM usage percent.
ps.swap_memory().percent	Swap usage percent.
ps.disk_partitions()	Mounted disk partitions.
ps.disk_usage('C:\\').percent	Disk usage of C: drive.
ps.disk_io_counters()	Disk I/O counters since boot.
ps.net_if_addrs()	Network interface addresses.
ps.net_if_stats()	Interface stats (up, speed).
ps.net_io_counters(pernic=True)	Per-interface network I/O counters.
ps.net_connections(kind='inet')	Active TCP/UDP connections.
for p in ps.process_iter(['pid','name'])	Iterate processes with selected attrs.
ps.pids()	List all PIDs.
ps.pid_exists(1234)	Check if PID exists.
p = ps.Process(1234)	Get Process handle.
p.name()	Process name.
p.exe()	Executable path.
p.cwd()	Current working directory.
p.cmdline()	Command line list.
p.username()	User name of process owner.
p.create_time()	Process start time (epoch).
p.status()	Process status (running, sleeping...).
p.cpu_percent(interval=None)	CPU percent since last call.
p.memory_info().rss	Resident memory (bytes).
p.nice()	Process priority (Windows returns base priority).
p.threads()	List of threads.
p.connections(kind='inet')	Per-process network connections.
p.open_files()	Files opened by process.
p.children(recursive=True)	Child processes.
p.terminate()	Politely ask process to terminate.
p.kill()	Force kill process.
ps.users()	Logged-in users (Windows: sessions).
ps.boot_time()	System boot time (epoch).

ps.sensors_temperatures()	HW temps (may be empty on Windows).
ps.win_service_iter()	Iterate Windows services.
ps.win_service_get('Dnscache').status()	Get Windows service status.
p.as_dict(attrs=['pid','name','exe'])	Process info as dict with selected attrs.
ps.test()	Quick self-test / info dump (debug).

PYWIN32 – Windows Event Log (Security/System)

Command (one line)	What it does
import win32evtlog, win32con	Import Event Log APIs and constants.
h = win32evtlog.OpenEventLog(None, 'Security')	Open local Security log handle.
win32evtlog.GetNumberOfEventLogRecords(h)	Count records in log.
flags = win32evtlog.EVENTLOG_BACKWARDS_READ|win32evtlog.EVENTLOG_SEQUENTIAL_READ	Set read flags.
events = win32evtlog.ReadEventLog(h, flags, 0)	Read a batch of events.
evt.EventID	Event ID (e.g., 4625 failed logon).
evt.TimeGenerated	Event time.
evt.SourceName	Source provider.
evt.EventCategory	Category code.
evt.EventType	Type (warning, error, info).
evt.StringInserts	List of strings (IP, user, etc.).
win32evtlog.CloseEventLog(h)	Close handle.
# Modern API (Vista+):	
q = win32evtlog.EvtQuery('Security', win32evtlog.EvtQueryReverseDirection, '*')	Query newer Windows Eventing API.
h2 = win32evtlog.EvtNext(q, 10)	Get next 10 results (handles).
win32evtlog.EvtRender(None, h2[0], 1)	Render an event as XML.

WATCHDOG – File/Folder Change Monitoring

Command (one line)	What it does
from watchdog.observers import Observer	Observer engine.
from watchdog.events import FileSystemEventHandler	Base event handler class.
from watchdog.events import PatternMatchingEventHandler	Pattern-based handler helper.
observer = Observer()	Create observer.
observer.schedule(handler, path=r'C:\\path', recursive=True)	Watch a directory recursively.
observer.start()	Start watching.
observer.stop()	Stop watching.
observer.join()	Block until thread exits.

PYDIVERT (WinDivert) – High-speed Packet Capture/Filter

Command (one line)	What it does
import pydivert	Import library.
w = pydivert.WinDivert('true')	Open with filter that matches all packets.
w = pydivert.WinDivert('tcp and tcp.PayloadLength > 0')	Filter TCP packets with payload.
w.open()	Open driver/handle (implicit with context manager).
for pkt in w:	Iterate over captured packets.
pkt.src_addr, pkt.dst_addr	Access IP addresses.
pkt.src_port, pkt.dst_port	Access ports.
pkt.is_outbound	Direction flag.
pkt.tcp, pkt.udp, pkt.icmp	Layer presence booleans/objects.
w.send(pkt)	Re-inject (forward) a packet.
w.close()	Close handle.
pydivert.WinDivert('inbound and tcp.DstPort == 3389')	Example: capture inbound RDP.
pydivert.WinDivert('udp and ip.SrcAddr == 8.8.8.8')	Filter by src IP.

PYSHARK – TShark-powered Capture & Parsing

Command (one line)	What it does
import pyshark	Import library.
cap = pyshark.LiveCapture(interface='Ethernet')	Live capture on interface.
cap.sniff(timeout=10)	Capture for N seconds.
for pkt in cap.sniff_continuously()	Iterate live packets.
cap = pyshark.FileCapture('in.pcap')	Open PCAP file.
cap.set_display_filter('http')	Set Wireshark display filter.
cap.apply_on_packets(func, timeout=30)	Apply callback to captured packets.
pkt.ip.src, pkt.ip.dst	Access IP fields (if present).
pkt.tcp.dstport, pkt.udp.length	Access TCP/UDP fields.
cap.close()	Close capture.
pyshark.LiveCapture(bpf_filter='port 53')	Use BPF filter for capture.