

## Session 4: The Exploitation Process & Using Metasploit

In order to fully understand the cyber threat, it is important to know the vulnerabilities, the tools, and the tactics that may be used during the exploitation process.

During this session, we'll provide an overview of attack planning, the most common tasks performed, and obstacles attackers have to overcome. We'll also cover how Metasploit is used in conjunction with tools like NMap or vulnerability scanners to run attacks.

### LO7: Discuss the three main stages of an attack

Hacking is a professional occupation now. We don't worry much about the script kiddie in his parents' basement, propagation malware like Nimda, or uncoordinated attacks of opportunity. Now, we defend against professional teams that are highly organized, specialized, and focused on achieving specific goals—whatever those might be.

Some high-level ICS attacker goals include:

- Espionage (commercial/geopolitical)
- Hacktivism (politically or socially motivated)
- Monetary/financial
- Retaliation
- Terrorism (including eco & counter)

The main three attack stages an attacker will use:

- Target Development
- Exploitation & Pivoting
- Attack Operations

The table below describes some of the most common tasks an attacker has to complete in order to successfully compromise a network.

Target Development (TD)	Exploitation & Pivoting (E&P)	Attack Operations
Research	Point of entry (PoE) exploits	Bypass & evasion
Map target network	Elevation of privilege (EoP) techniques	C&C infrastructure
Identify potential points of entry (PoE)	Pivoting techniques	Persistent, remote access
Plan attack	Lateral movement on target network	Malware updates
Build tool kit		System-acquisition

### Learning Objective 7

## Target Development (TD)

Good attackers learn about the organization and network they are targeting before they do anything else. We'll refer to this as developing the target.

Why do they develop the target? As with any other work, planning their work ahead of time ensures they can perform their work successfully.

### Developing the Target

How do attackers actually develop a target? TD generally entails:

- Establishing goals for the attack
- Selecting the target (strategic or opportunistic)
- Researching the target organization
- Discovering the target configuration
- Exploring entry points
- Planning the attack
- Building the tool kit.

**Establishing goals for the attack.** Before they get started, attackers decide why they are running the attack and what they want to accomplish.

**Target Selection.** Networks, systems, applications, and data get targeted for one of the following reasons:

- Targets of opportunity – The attacker happens to have the attack technology necessary to exploit the target. For example, your browser happens to be vulnerable to the exploit being served up by the exploit kit on the website you visited.
- Strategic targeting – The attacker wants something from the network being targeted for reasons specific to his strategic goals. A network is tagged because the attacker wants to steal important data, establish a long-term foothold, or cause interruptions.

For the purpose of this class, we'll assume the target has been selected for strategic reasons.

**Researching the Target.** Once a target has been selected, the attacker begins to learn about the people using the network and what work is being done.

At this point, mapping the target technology isn't as important as investigating what is available on the target network. The attacker will gather data that makes it easier to compromise the network and to ensure they don't get caught.



In this stage expect them to:

- Check DNS records and network allocation in Internet records
- Run open source search queries for available organizational information, work flow, or deployed technology
- Review websites, externally available applications, or social media accounts
- Identify and analyze all available files on the target's external DMZ network including raw HTML pages looking for anything to help them success
- Identify employees, vendors, service providers, leadership team
- Explore social media for any information regarding employees, vendors, service providers, and leadership team
- Peruse job postings or contract opportunities
- Use social engineering techniques like calling the help desk to get specific information and sending phishing emails to targeted employees to establish relationships.

**Discovering Target Configuration.** Once an adversary has a good idea about the target organization and what they do, the technical work begins. Attackers need to know:

- What attack paths are available?
- Which systems and services are running?
- What software is being used?
- What privileges and accounts are accessible?
- What configuration information is available?
- What security is in place?

At this point, the attacker will begin to map or probe the target network actively with discovery tools such as NMap and look for other weaknesses like SQL Injection vulnerabilities. Results will be recorded and organized for future use.

In later sections, we'll discuss how to utilize this information to plan and to run attacks with Metasploit.

**Exploring Potential Entry Points.** In this stage, attackers will evaluate all of the information gathered, and then identify and prioritize the best places to run the initial attacks.

Attackers will start utilizing vulnerability discovery tools and techniques. During vulnerability discovery, attackers probe targets for vulnerable services, software, or configurations in order to identify potential points of entry and plan social engineering campaigns.

You have learned to use vulnerability scanners to discover vulnerabilities. In later sections, you'll see how to use Metasploit as well.



**Plan the Attack.** Once the data is gathered, the attackers will plan out the attack using standard project management methods. Assignments need to be made, resources assigned, possible vulnerabilities are selected, along with the appropriate exploits as the project moves towards the next phase of getting the initial point(s) of entry (PoE).

**Building a Tool Kit.** For this class, we'll use the Metasploit Framework (MSF), however professional attackers have their own highly specialized tool kits. Some exploitation tool kits you may have heard of are BlackEnergy, Flame, and Regin.

## Exploitation and Pivoting

**Exploitation** – Attack technology, tools, or techniques used to compromise a target initially. These are also generally used to gain enough privilege to establish long-term access to the target.

**Pivoting** – Attack technology, tools, or techniques used to find other targets of interest on a compromised network and to compromise them. Pivoting techniques are most often used to move between targets with different trust levels.

## Attacker's Trifecta

The attacker's trifecta consists of three components. Each of these must be available in order for an attacker to exploit a system successfully. The attacker's trifecta is any combination of:

- A **Vulnerability** an attacker will use to get Point of Entry (PoE) on a target system
- An **Exploit** used to gain initial entry onto the target via a specific vulnerability with the intent of elevating privilege
- A **Communications (or Attack) path** used to deliver the exploit to the vulnerable target.

## Types of Vulnerabilities

For the purpose of this class, we'll discuss two kinds of vulnerabilities: system and software vulnerabilities.

**System Vulnerabilities.** A system vulnerability is: *"A flaw or weakness in system security procedures, design, implementation, or internal controls that could be exercised (accidentally triggered or intentionally exploited) and result in a security breach or a violation of the system's security policy."*

System or environmental vulnerabilities exist because the owner of the target environment hasn't designed, implemented, or maintained a system resource appropriately.

Examples of this kind of vulnerability include:

This class defines a system or environmental vulnerability per NIST SP800-30, "Risk Management Guide for Information Technology Systems."



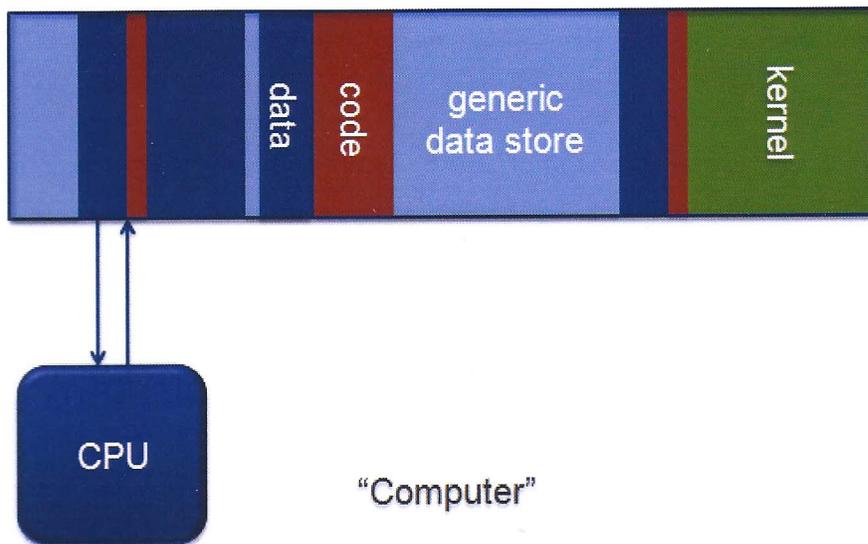
- Default and/or hard coded and/or easily guessed passwords for applications and/or services
- Operating systems that are not (or cannot be) changed after installation
- Failure to upgrade, update, or mitigate software in a timely manner when vulnerabilities are announced
- Critical resources are not adequately segregated on the network with firewalls or network segmentation
- Direct public Internet access to and/or from ICS systems and devices
- Sensitive corporate documents residing unsecured on public facing servers.

**Software Vulnerabilities.** A software vulnerability is “... a mistake in software that can be directly used by a hacker to gain access to a system or network. CVE considers a mistake a vulnerability if it allows an attacker to use it to violate a reasonable security policy for that system (this excludes entirely "open" security policies in which all users are trusted, or where there is no consideration of risk to the system).”

Software vulnerabilities exist because flaws in the software are susceptible to exploitation using common or known attack techniques. Software vulnerabilities may also exist because the software wasn't designed by a vendor to be deployed securely.

## Software Vulnerabilities and Exploits – A Brief History

### Memory Architecture



This class defines a software vulnerability according to the Common Vulnerabilities and Exposures (CVE) terminology.

#CVE

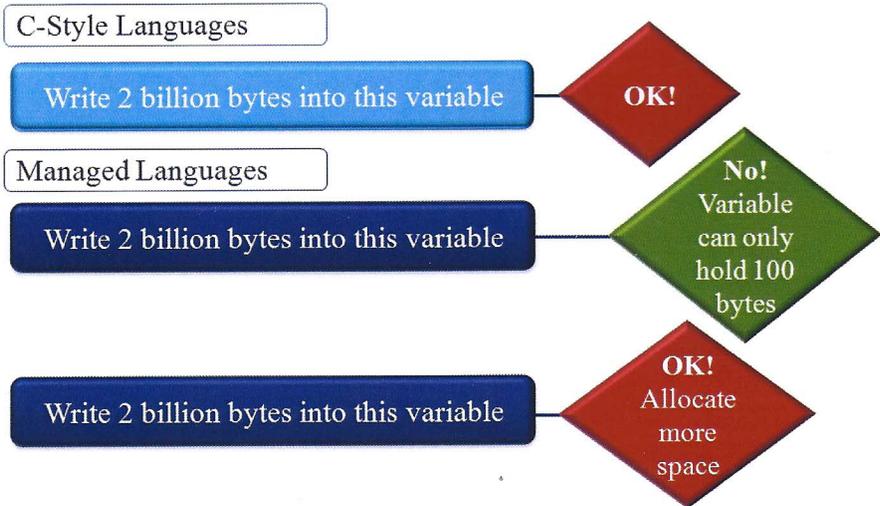
The operating system arbitrarily assigns memory as needed to be used for code, data, kernel, etc. The operating system generally relies on the applications to tell it how much memory is needed for their data. This allows memory to be dynamically assigned on an as-needed basis and freed up for other use when it is no longer required by the program.

Today, most programming languages are modular, using what is typically called a function or an object to define blocks of code that will be loaded into memory as needed and then removed.

### History & Exploitability: Language

Just as the operating system relies on the program to tell it how much memory to allocate, in C-style languages the compiler relies on the programmer to know how to allocate the correct amount of memory and write checks to ensure input is correctly handled. These types of languages assume that the programmer knows what they are doing. So, generally what is written is what happens.

For instance, the programmer makes the request to write two billion bytes into this address, and the system response is “ok!”



Managed languages have been programmed to do some basic checks and provide some safeguards against mistakes. A request that two billion bytes be written into a variable that can only hold 100 bytes, is refused by the system. However, managed languages can still be exploited in other ways. For example, most managed languages are written in a C-style language and can still have flaws in their libraries.

---

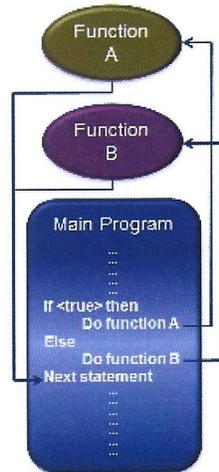
---

---

## Typical Program Flow

This is a simple example showing program flow. In this instance, the program was written with a conditional statement that determines when to use either function.

- Main program:
- If <condition>,
  - Then do function A
    - Return to main
  - Else do function B
    - Return to main



## Exploitability Example 1 (C-Style)

```

void copyBytes(char * source, char * dest, int buffLen){
    for (int i = 0; i < buffLen; i++){
        dest[i] = source[i];
    }
}
  
```

- Denial of service when *buffLen* is user/attacker controlled
- Arbitrary Code Execution (ACE) when *buffLen* and *source* are user/attacker controlled
- What is a shell code?

*control buffLen  
how you can set  
DoS*

*— buffer length —*

## Attacker Controlled Program Flow

- Do function A
  - Run shell code!
  - Then Return to Main.

---

---

---

---

---

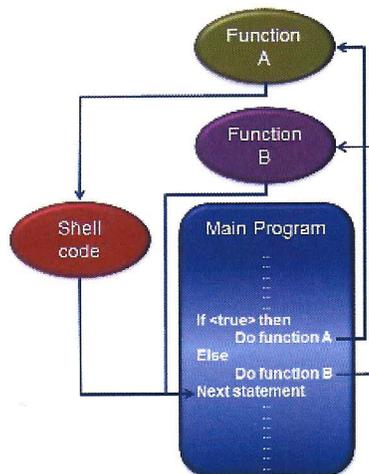
---

---

---

---

---



/ C  
 / C++  
 / Assembly

This kind of bug	Typically results in
Buffer Overflow	Arbitrary Code Execution
Buffer Underflow	Arbitrary Code Execution
Format String	Arbitrary Code Execution
Integer Overflow / Underflow	Arbitrary Code/Denial of Service
Signed Value Bug	Arbitrary Code/Denial of Service
Double Free / Null Pointer	Arbitrary Code/Denial of Service
...	...

This list shows several examples of this kind of vulnerability.

- Buffer Overflow/Underflow** - A stack-based buffer overflow condition is where the buffer being overwritten is allocated on the stack (i.e., a local variable or a parameter to a function). A heap overflow condition is a buffer overflow, where the buffer that can be overwritten is allocated in the heap portion of memory, generally meaning that the buffer was allocated using a routine such as malloc(). A Buffer Underflow typically occurs when a pointer or its index is decremented to a position before the buffer, when pointer arithmetic results in a position before the beginning of the valid memory location, or when a negative index is used.
- Format String** - The software uses unchecked user input as the format string parameter in certain C functions that perform formatting, such as printf(), which can lead to buffer overflows or data representation problems.
- Integer Overflow/Underflow** - An integer overflow condition exists when an integer, which has not been properly validated, is used in the determination of an offset or size for memory allocation, copying, concatenation, etc. If the integer in question is incremented past the maximum possible value, it may wrap to become a very small number, or if the integer is decremented past the lowest possible number (e.g. zero) it may wrap to become a very large number.
- Signed Value Bug** - The software uses a signed integer and performs a cast to an unsigned integer, which can produce an unexpected value if the value of the signed number cannot be represented using an unsigned number. Often, functions will return negative values to indicate a failure. When the result of a function is to be used as a size parameter, using these negative return values can have unexpected results. For example, if negative size values are passed to the standard

memory copy or allocation functions they will be implicitly cast to a large unsigned value. This may lead to an exploitable buffer overflow or underflow condition.

- **Double Free** - When a program calls free() twice with the same argument, the program's memory management data structures can become corrupted. This corruption may cause the program to crash or, in some circumstances, cause later calls to malloc() to return the same pointer. If malloc() returns the same value twice and the program later gives the attacker control over the data that is written into this doubly-allocated memory, the program becomes vulnerable to a buffer overflow attack.
- **Null Pointer** - A NULL pointer dereference occurs when the application dereferences a pointer that it expects to be valid, but is NULL, typically causing a crash or exit.
- **SQL Injection** - Without sufficient removal or quoting of SQL syntax in user-controllable inputs, the generated SQL query can cause those inputs to be interpreted as SQL instead of ordinary user data. This can be used to alter query logic to bypass security checks, or to insert additional statements that modify the back-end database, possibly including execution of system commands.

## Vulnerability Severity

All vulnerabilities, if exploited successfully, result in an unexpected technical impact to a system or software. Vulnerability severity is generally expressed in terms of its Common Vulnerability Scoring System (CVSS) score.

The CVSS score expresses the probability a vulnerability will be exploited based on the ease of exploitability, access vector, access complexity, and authentication required to exploit it. It also addresses the technical impact successful exploitation of the vulnerability would have on the target system in terms of confidentiality, integrity, and availability.

*Note:* A great resource on vulnerabilities and their CVSS score is the National Vulnerability Database <https://nvd.nist.gov/>

*NVD*  
*CVSS*

The screenshot shows the NVD website interface. At the top, it says "Sponsored by DHS/NCCIC/CIS-CERT" and "NIST National Institute of Standards and Technology". The main heading is "National Vulnerability Database" with the subtitle "automating vulnerability management, security measurement, and compliance checking". Below this is a navigation menu with links like "Home", "SCAP", "SCAP Validated Tools", "SCAP Events", "About", "Contact", "Vendor Comments", and "Visualizations". The main content area is titled "National Cyber Awareness System" and "Vulnerability Summary for CVE-2014-9369". It includes the following information:

- Mission and Overview:** NVD is the U.S. government repository of standards-based vulnerability management data. This data enables automation of vulnerability management, security measurement, and compliance (e.g. FISMA).
- Resource Status:** NVD contains: 71636 CVE Vulnerabilities, 302 CPEs, 249 USCERT Alerts, 4380 USCERT Vul Notes, 10286 CWA Queries, 105490 CPE Notes. Last updated: 01/11/2015 12:54:28 PM. CVE Publication rate: 16.9.
- Email List:** NVD provides four mailing lists to the public. For information and subscription instructions please visit NVD.
- Vulnerability Summary for CVE-2014-9369:** Original release date: 03/06/2015, Last revised: 03/09/2015, Source: US-CERT/NIST.
- Overview:** Siemens SPC controllers SPC4000, SPC5000, and SPC6000 before 3.6.0 allow remote attackers to cause a denial of service (device restart) via crafted packets.
- Impact:** CVSS Severity (version 2.0): CVSS v2 Base Score: 7.8 (HIGH) (AV:N/AC:L/Au:N/C:N/N/A/C) (legend), Impact Subscore: 6.9, Exploitability Subscore: 10.0.
- CVSS Version 2 Metrics:** Access Vector: Network exploitable, Access Complexity: Low, Authentication: Not required to exploit, Impact Type: Allows disruption of service.

When you are using Metasploit later in the class, remember the best vulnerabilities from an attacker's perspective:

1. Are remotely exploitable
2. Can be triggered by an unauthorized user
3. Allow arbitrary code execution (ACE) upon exploit.

## Exploits Types

An exploit is an attack technique or technology an attacker uses to take advantage of a vulnerability in order to cause unintended or unanticipated behavior. Exploits are a sub-category of attack technology and are used to perform a specific task (exploitation), which results in Point of Entry (PoE) on a target system.

Historically, exploit or shell code, another term for exploits, might also allow an attacker to elevate privilege and drop a payload on the system. However, that has become increasingly less common as software vendors implement secure code development programs. Today, exploits are more likely to be included in payloads like exploit kits.

### Characterizing Exploits

Exploits can basically be classified into two delivery location categories: **remote exploits** which work over a network and exploits a vulnerability without needing physical access to the vulnerable system and **local exploits** that requires physical access or at least logged onto a vulnerable system. Many 'local exploits' are usually used to increase the privileges of the attacker running the exploit past those granted by the system administrator.

**Client side exploits** are another classification based on origin of the vulnerability and consists of compromised (or designed) servers that send an exploit if accessed with some client application and **server side exploits**. Exploits against client applications may require some interaction with the user and so may be used in combination with a social engineering attacks such as email phishing or enticing the user to click on a link.

In many cases an attacker will use several exploits, first to gain low-level access and then others to escalate privileges and/or create access persistence.

Normally a single exploit can only take advantage of a specific software vulnerability, most times on a particular version of the software. Once the vulnerability is made public the vendor will usually patch the vulnerability, however if an attacker gets persistence before the patch is applied the vulnerability becomes a moot issue. In fact many attackers will patch a system after exploiting it to keep other attackers out.

For the purposes of this class, we'll discuss attack techniques or exploits in terms of:

- Exploit delivery location (remote, local)
- Origin of vulnerability on target system (server side, client side, application).

**Remote attack** – In a remote attack, the attacker tries to exploit any system that:

- Can be run remotely over the Internet or any other network to the vulnerable system
- Doesn't require the attacker to log onto the target system before running the exploit.

Remote exploits are generally much more serious than local ones, but fortunately, remote exploits are much easier to prevent and are generally less common. A remote attack can be launched by any of the hundreds of millions of people on the Internet at any time. The most common remote exploits are buffer overflow and other unchecked input attacks. They are either done against public services (such as HTTP and FTP) or during the logon of protected services (such as SSH and IMAP).

**Local attack** – The attacker has access to an account on the system in question and can use that account to attempt unauthorized tasks. Local exploits are much more common and difficult to prevent.

**Server side attack** – Servers expose a service that clients can interact with where these services include things such as file sharing. As a server provides services to a client, it can expose vulnerabilities which can be attacked. SQL Injection is a server side attack.

**Client side attack** – Client-side attacks target vulnerabilities in client applications interacting with a malicious data. The difference is the client is the one initiating the bad connection.

Client-side attacks are becoming more popular. This is because server side attacks are not as easy as they once were.

Attackers are finding success going after weaknesses in desktop applications such as browsers, media players, common office applications and e-mail clients.

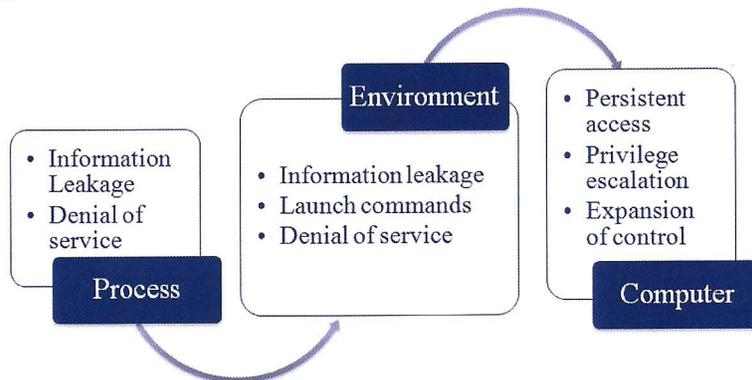
These client applications do not listen for connections on a TCP or UDP port like server applications, they instead call out to servers. Once a client connects to an infected server the server attempts to download an exploit to the client thus bypassing many forward-facing security roadblocks.

*NOTE:* Software vendors with good code security programs will try to eliminate vulnerabilities that are remotely exploitable first. Microsoft began doing this with the release of XP Service Pack 2, resulting in the push to exploit at the application layer rather than the OS.



## Exploitation Process

This is the process of vulnerability discovery and how we get to ACE.



It starts with fuzzing or some other type testing to determine the nature of the vulnerability. The idea is to get something to happen the programmer did not anticipate and protect the application against. This is achieved when a denial of service is created by crashing the application.

Typically, that will result in information leakage about the application and its operating system. The next step is to try and take advantage of this vulnerability by writing an exploit.

Once the exploitation succeeds and understanding how the vulnerability works, it's time to tie it to shell code and see what information and/or access can be gained from the compromised system.

Finally the attacker will want to get elevated privileges. Many organizations are moving towards the "Least User Privilege" paradigm where users have the least number of privileges needed to perform their jobs, however it is probable that attackers will use additional local system exploits to get elevated privileges.

### Pivoting

Pivoting is a process in which an attacker uses a compromised computer system to attack other systems in an effort to avoid firewall rules, which restrict direct access to other portions of a network.

For example, if an attacker compromises a webserver in a company's DMZ, he may then be able to take advantage of the trust relationship between the compromised webserver and the internal company network circumventing the firewall restrictions and thus "pivot" into the internal network. It would then be a simple matter to set up network routing on the compromised DMZ webserver and the attacker would then have access directly into the company's internal network.

In the ICS world there are many instances where getting data out of the ICS network is important to company's financial interests. However, just as in a typical DMZ any communication path between two networks has a potential be to be exploited. Typical types of data into and out of a process control network include:

- Regulatory data
- Raw product usage
- Power/Energy consumption/generation
- Safety and maintenance indicators
- Production status and/or rates
- Historical data collection
- Software/process updates
- Remote access in the event of an emergency.

Many times these connections are done through dual or multi-homed computer systems. In other words, systems with multiple Network Interface Controllers (NIC's) built into them allowing the systems to communicate with two or more networks. Once an attacker gains access to these types of systems the attacker has access to all of the network interfaces.

In other instances these connections are controlled by firewall rules forming trust relationships between two or more networks.

Attackers will attempt to take advantage of these communication paths to achieve their goals and so if these paths cannot be turned off or disconnected they should be carefully monitored for abnormal traffic flows.

### **Lateral Movement**

Lateral movement is simply when an attacker compromises additional hosts on the same network -- no pivoting required. This lateral movement will be on the same subnet or even other subnets if routing has been enabled between the subnets.

Lateral movement allows attackers to own additional systems for hiding their presence, data gathering, finding possible pivot points, and other nefarious acts. If the originally compromised system is removed, shutdown, replaced, etc. the attacker will have other systems from which to operate.

Lateral movement is hard to detect if the attackers are stealthy and try to stay below automated detection levels, however it can be done with internal honeypots, watching for scanning activities coming from hosts that should not be scanning other systems, or by knowing if there is absolutely no reason why two hosts should be talking to each other.

For a really interesting article on a control system attack, read: “*To Kill a Centrifuge*” by Ralph Langner.

<http://www.langner.com/en/wp-content/uploads/2013/11/To-Kill-a-Centrifuge.pdf>

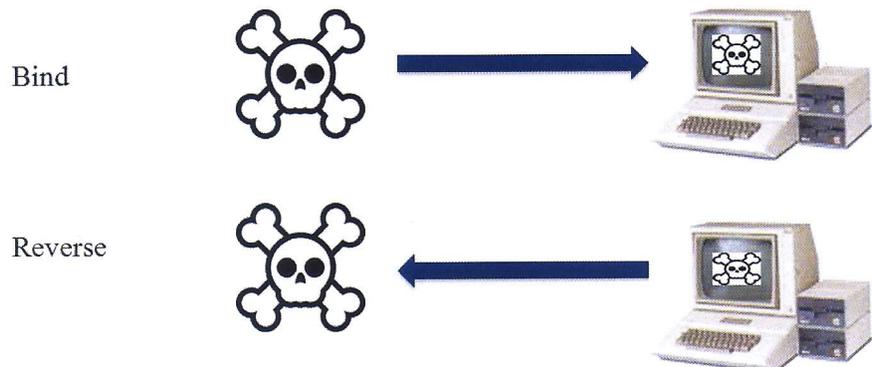
## Attack Operations

Attack operations starts once the attackers have gained access to the target organization’s network. Depending on the operational goals of the attacking organization this could be a long term relationship (hopefully, at least to the attackers, only known to the attacking group) that could perhaps last for years. The goals for this phase might include:

- Bypass network controls
- Get/maintain persistence
- Setup command and control operations
- Evade detection
- Acquire additional systems
- Update tools as needed
- Find/gather/exfiltrate data
- Sabotage processes and/or communications.

## Bypass Network Controls

Bypassing network controls is more than pivoting as explained above. It also includes establishing connections to command and control systems is the most inconspicuous way as possible. There are basically two types of network connections: Bind and Reverse.



**Bind** connections or direct connections is where one system, we’ll call it the “client” system makes a connection directly to a host or “server” on a given service port. The client “binds” its connection to the server. A simple example is a web browser somewhere out in the internet that makes a connection to a webserver on port 80 or 443 in some company’s DMZ. In other words the attacker has a direct path to the target system and can launch attacks directly at the target.

**Reverse** connection attacks can be accomplished in multiple ways. One way is for an attacker to craft an exploit and send it in as an email attachment or on a USB thumb drive that when executed or opened attempts to establish a connection out to an attacker controlled server. The most common is through email spear phishing attacks where the attacker sends a carefully crafted email that

contains a link to a malware server to selected individuals within the targeted organization. When a user clicks on the link the system opens an outbound connection to the infected server and the exploit is delivered back.

It has been reported that in sanctioned email phishing tests “... nearly 50% of users open e-mails and click on phishing links within the first hour ...” and that “... a campaign of just 10 e-mails yields a greater than 90% chance that at least one person will become the criminal’s prey ...”

Many organizations have really good network perimeter defenses; nothing gets in that shouldn’t, however they allow unrestricted outbound connections to just about anywhere.

Okay, how does a defender overcome such staggering odds? The answer is that there is no one catch-all solution to counter phishing attacks. Generally it takes:

- Determined email filtering
- An aggressive and directed security awareness program
- A really good intrusion detection and incident response program with constant vigilance.

## Command and Control

Attackers must be able to communicate with, maintain long-term access to, and remotely administer compromised systems. In order to do so, they must maintain an infrastructure of their own.

They will establish a network of communications and control (C&C) systems, select C&C channels over which compromised systems will phone home, push malware updates to compromised systems, move data between exfiltration points, etc. They will also have to maintain and monitor the attack infrastructure to keep tabs on their progress and to see what else needs to be done.

They employ complex layers of misdirection to hide their C&C systems, literally transmitting commands and data across oceans and countries and hide commands within common network protocols.

Measures that help in detecting and stopping C&C services include:

- Monitor DNS requests for internal devices trying to make connections to known bad addresses
- Monitor outbound network traffic to known bad addresses
- Monitor network traffic against known bad signatures (i.e. keep IDS boxes up-to-date)
- Monitor for traffic between hosts/devices that have no need to and/or should not communicate with each other
- Monitor for large outbound data transmission sessions

Note: See Verizon’s “2015 Data Breach Investigations Report (DBIR)” for additional email phishing statistics.

To Detect  
C&C

- Segment the ICS network(s) from the corporate network
- Baseline the ICS network traffic and then monitor for abnormal behavior
- A new trend is to use some kind of network traffic visualization tool where the human eye can visually see any anomalies.

## Persistent and/or Remote Access

Once a foothold is gained on the target network one of the first things an attacker will want to do is gain persistence. Persistence means that if the compromised system is rebooted and/or patched the malware will automatically execute and re-establish a connection back to the command and control system. The last thing an attacker wants to do is re-attack the same system, especially if the system has been patched and the original vulnerability removed or mitigated. This is where the term Advanced Persistent Threat (APT) comes from. It's applied to groups who are extremely stealthy and maintain persistence in both their presence on the target network and in consistently monitoring, gathering data, and maintaining/updating their tool set.

Attackers want to have remote and consistent access to the victim network for as long as possible. The best part of being an attacker is when you can reach out and touch your victim from half a world away as if you're right there.

The great thing about persistence is if the original vulnerability is patched access continues. In fact, some attackers will apply the patches to fix the vulnerabilities they used get access in the first place to stop competitors from getting in also. Persistence can be accomplished in a variety of ways, from simply adding an entry into startup scripts or replacing some shared library (e.g. DLL) with a specially crafted one.

In more extreme cases the attackers flash the BIOS or some other non-volatile memory that may never really be patched, where the only solution may be to take the device out of service.

One problem with the term persistence is that it doesn't mean a persistent connection, as in the connection remains for days at a time. Malware is known to only "phone home" on occasion according to some schedule in order to remain undetected. On top of that they try to blend in with normal network traffic using standard ports and protocols to hide their acts in the regular "noise" of network traffic.

How do you combat persistence? Like most everything else. Constant vigilance, patching, and malware signature updates.

*APT*  
*Ps 114*  
*"To Kill a Country"*

With remote access an attacker can be anywhere in the world and play with your equipment. With remote access attackers can send in new commands, update their tools, exfiltrate data, manipulate processes, steal money, cause hate and discontent, and generally have a good time.

Normally, attackers use many servers to communicate with a victim. Known bad actors understand that any one IP address is only viable for a couple weeks at best so they constantly acquire and change addresses in order to maintain their remote access. This makes it extremely hard to detect and stop these connections. By the time an IP is identified and mitigations are put in place it changes to something else.

The best way to stop remote access is to completely remove all access to critical network devices and systems. Of course this isn't really practical. Even air gapped systems can be attacked and owned, the STUXNET intrusion proved this point.

A good cyber security program for ICS environments should include policy and procedures (that are enforceable) for the handling and protection of removable media, wireless devices, and hand-held portable devices.

Another option is to employ one-way data diodes when sharing data from an ICS network to the corporate network. Many people believe that they have a one-way only connection between the ICS network and a database server because the data is only "pushed" from the ICS network to the database.

This is a false understanding because the actual data transmission protocol is bi-directional as the sender and receiver must pass information back and forth to keep track of what's happening with the data transmission and any bi-directional communication protocol has a risk of being hijacked and/or manipulated to allow an attacker to gain access to the sending system. One-way data diodes attempt to ensure the connection really is only one way.

## Summary

### Target Development:

- Research, research, and more research on who, what, when, and where
- Map the target network and look for items of interest including servers and what's on them (web, FTP, DNS, email, etc.)
- Identify potential attack points of entry, including phishing
- Plan the attack in an organized way and assigning roles and responsibilities and the command and control infrastructure
- Build/customize attack tools.

STUXNET



## Learning Objective 8



For more information, you can visit:  
[www.metasploit.com](http://www.metasploit.com)  
or  
[offensive-security.com/metasploit-unleashed/Main\\_Page](http://offensive-security.com/metasploit-unleashed/Main_Page)

### Exploitation & Pivoting:

- Execute planned attacks
- Get a point of entry
- Elevate privilege
- Find additional targets and pivot points - repeat
- Be stealthy and don't get caught – low and slow.

### Attack Operations:

- Establish command and control communications
- Establish long term persistence from the inside out
- Find the high value systems and targets
- Exfiltrate data and analyze it looking for passwords, network/PCS information, network connection information (netstat), company documents, design information, banking information, and anything else of interest.

## LO8: Describe Metasploit

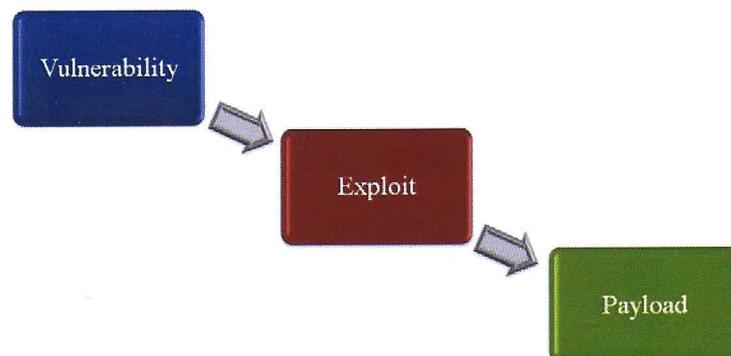
Now that we have discussed the elements of a cyber attack, we will continue the discussion on how attacker tools make life easier for them with an overview of the Metasploit framework.

### MS Framework

- The Metasploit framework is both a **penetration testing system** and a **development platform** for creating **security tools** and **exploits**.
- The framework consists of **tools**, **libraries**, **modules**, and **user interfaces**.
- The basic function of the framework is a **module launcher**, allowing the user to configure an exploit module and launch it at a target system.

### Terminology

There are three terms that are important to understand when working with Metasploit: vulnerability, exploit, and payload.



A **vulnerability** is a software flaw that may be susceptible to exploitation.

An **exploit** is an instance of an attacker taking advantage of a vulnerability.

A **payload** is the module the chosen exploit will run when a machine is compromised.

## The Basic Exploit Process

The exploit process follows ten basic steps:

- Show exploits
- Use exploit *<full exploit name>*
- Show options
- Set *<option name> <value>*
- Show payloads
- Set PAYLOAD *<full payload name>*
- Show options
- Set *<option name> <value>*
- Set TARGET *<value>*
- Exploit.

## MSF: Useful Commands

Some of the common commands used with Metasploit are listed below.

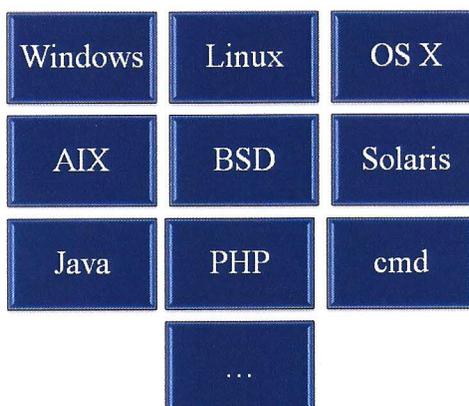
Command	Description
help	Help menu. "help<command>" or "<command>-h" prints information about <command>
search	Searches module names and descriptions
info	Displays information about one or more modules
use	Selects a module by name Modules can be anything in modules directory (includes exploits, auxiliary, encoders, payloads*, nops)
set, unset, setg, unsetg	Sets/unsets one or more variables Typically used to select a payload
sessions	List, interact with existing sessions

## MSF Payloads Overview

- OS Targets
- Classes
- Communication Options
- Action Types
- Modifiers
- Meterpreter (will be covered in depth later in session)



## MSF Payloads: OS Targets



## MSF Payloads: Actions

MSF payloads use the following codes to execute actions:

- Shell gives a shell
- adduser, adds a user to the system
- chmod, changes permissions on a file
- inetd lets inetd listen for connection
- VNCInject gives a GUI
- Upexec, downexec should upload/download & execute a file
- Meterpreter replacement shell.

## MSF Payloads: Modifiers

In addition, payloads use modifiers.

- nonx will try to bypass DEP
- reflective dll injection method
- dllinject, dll injection method
- patchup dll injection method
- ordinal (ord) dll injection method
- ipv6 run using ipv6 instead of ipv4
- dns do dns lookup
- allports tries all ports looking for one that will work

## Meterpreter: Useful Commands

**help, <command> -h**

- lists commands, displays usage about a command

**execute**

- launch a command on the host system
- example: execute -f cmd.exe -i

use

- loads extension

shell

- launches a shell on host system, and interacts with it

channels

- list, interact with existing channels

Other useful meterpreter commands include:

ps

- list process

migrate pid#

- get a pid from ps, then migrate

run persistence-h

- shows options for persistence

portforwarding in meterpreter

- will forward a port from the victim to the attacker or vice versa

run hashdump

- dump hashes

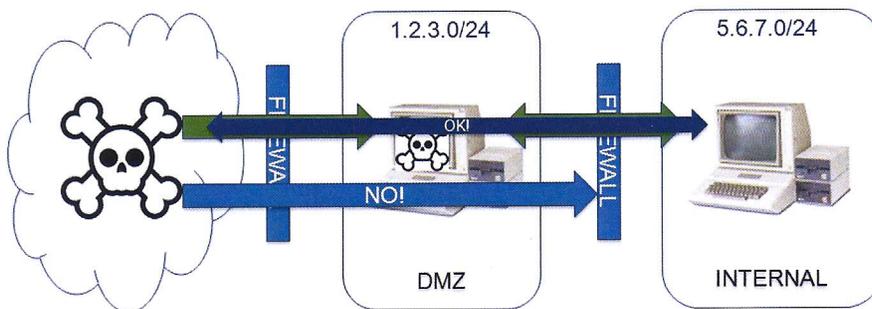
run autoroute-h

- sets up routing

getuid

- shows your current privilege level

### Meterpreter Routing



This graphic illustrates pivoting. The attacker is able to talk to the host in the DMZ directly, but not the internal host. However, by pivoting, the attacker can send packets to the DMZ host who forwards them. The DMZ machine has permissions on the firewall allowing it to talk to the internal machine.

Pivoting can be accomplished using meterpreter by setting up routing by using **autoroute** within meterpreter, or the **route** command within msfconsole.

Some interesting things to do with routing and the meterpreter

*shell  
 payload /windows/shellbind #cp  
 ↓  
 /meterpreter/*

*meterpreter  
 Trust level  
 DMZ ↔ Internal  
 &  
 Firewall ↔ DMZ  
 As a defender you can find it in  
 Routing Tables*

session include using:

- The **portfwd** command through meterpreter to reach a specific port on a remote host machine
- The **socks4a** module and **proxychains** to use external commands, such as nmap and rdesktop.

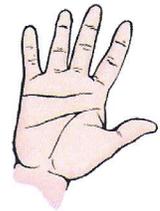
## LO9: Use the Metasploit framework

In order to understand the exploitation process, you will complete a hands-on exercise. This exercise will cover how to:

1. Use different exploitation techniques to gain an initial point of entry on the target network and how to elevate privilege;
2. Consider the technical parameters within which your attack must be run; and
3. Maintain long-term access to a compromised system by installing payloads.

## Class Customizations to Metasploit and Kali Hands-on Exercise

For the purposes of this class, some aspects of the Kali image and Metasploit modules have been expanded to demonstrate attack planning and execution. The target systems, identified vulnerabilities, and exploits in specific were developed for this course to ensure ease of instruction.



This means that you will have to run the Metasploit tools and payloads included in the Kali DVD you received at the beginning of class. A standard Metasploit or Kali installation will not include the information and files necessary to perform some of the exercises.

### MSF and *msfconsole*

There are a number of different interfaces for MSF. In this class, we will be accessing the Metasploit Framework through the *msfconsole*. *Msfconsole* is a command line interface that provides centralized, supported access to almost all of the MSF features.

### Miscellaneous *msfconsole* and MSF features

- *Msfconsole* will relay any commands it doesn't understand to the underlying operating system. That just means you can use any OS commands, e.g. **ifconfig eth0**, within *msfconsole*.
- Metasploit is NOT case sensitive.
- MSF naming convention uses underscores not hyphens, which is important when you are searching for modules or trying to use autocomplete.
- A session refers to the interaction between your instance of *msfconsole* and the instance of Meterpreter running on a compromised system.
- *Msfconsole* supports tab completion.