On the Security of PPPoE Network

Fanbao Liu[1,2], Yumeng Feng[2], Yuan Cao[1]

[1] School of Computer, National University of Defense Technology, Changsha, 410073, Hunan, P. R. China
[2] School of Computer, Beijing University of Technology, 100124, Beijing, P. R. China
liufanbao@gmail.com

Abstract. PPPoE is a network protocol for encapsulating PPP frames inside Ethernet. It is widely used by commercial ISPs to authenticate peers, who want to surf the Internet by paying the bills. In this paper, we analyze the security of the PPPoE network, we find that we can easily collect information about both the peers and the PPPoE authentication servers. We can use such information to recover the peer's authentication password by silently impersonating the server, which is undetectable in the network. We impersonate the server in the peers' LAN and get their passwords by hijacking the peers' PPPoE connections and negotiating for using the PAP authentication protocol. We further propose an efficient password recovery attack against the CHAP authentication protocol. We first recover the length of the password through on-line queries, based on the weakness of MD5 input pre-processing. Then we crack the known length password off-line, using the probabilistic context-free grammars. For MS-CHAP has already been proved to be weak, and the more secure EAP authentication methods can be by-passed through roll-back attack, which negotiates the weak protocols, the authentication passwords of the PPPoE networks are truly in danger. We point out that PPPoE can't be used any more, until all of the weak authentication protocols including PAP, CHAP, MS-CHAP are abolished right now and replaced with more secure EAP methods.

Keywords: PPPoE, Authentication Protocol, Password Recovery, PPP, PAP, CHAP.

1 Introduction

Point-to-Point Protocol (PPP) is a data link protocol commonly used in establishing a direct connection between two networking nodes [15]. It can provide connection authentication, transmission encryption privacy, and compression, et al. PPP can be used by Internet service providers (ISPs) for customer dial-up access to the Internet. Along with the wide deployment of the Ethernet, an encapsulated form of PPP, Point-to-Point Protocol over Ethernet (PPPoE), was proposed to provide the ability to connect a network of hosts over a simple bridging access device to a remote Access Concentrator[9].

PPPoE, a method for transmitting PPP over Ethernet, has the advantage that neither telephone companies nor ISPs need to provide any special support.

PPPoE is now used most commonly by ISPs to establish an Internet service connection with customers. Through PPPoE, an office or a building that full of users, sharing an Ethernet, can access the Internet independently and freely.

PPPoE has two distinct stages, the PPPoE discovery stage and the PPP session stage. The first stage is used by a peer to get information about the authentication server for starting the PPP session, where the server authenticates the peer or even mutual authentication happens. Only after the PPP session is validated, the peer can freely surf the Internet, and the server will count the bills for him.

In this paper, we analyze the security of the PPPoE network. We find that there exists a fatal security hole in the PPPoE discovery stage, which can be used by attackers to easily hijack the peers' PPPoE connections. We also find that the current using authentication protocols in PPP are vulnerable to password recovery attack.

Our Contributions. By carefully analyzing the PPPoE discovery stage, we find that it is easy for an attacker to collect enough information to impersonate the PPPoE authentication server and further hijack the peer's connection.

Most of the prevalent operating systems such as "Windows 7" support the PAP authentication protocol, which transfers the peer's password in clear over the untrusted network, in the default setting of the PPPoE security. We can easily recover the peer's password by hijacking his connection and negotiating for using PAP.

Further, we propose an efficient password recovery attack to CHAP authentication protocol in PPPoE. We first recover the length of the password by on-line queries, using a known security hole in the MD4 hash family. Second, based on the efficient password cracking using probabilistic context-free grammars, we further improve it to quickly crack the known length password.

We confirm that it is unsuitable to further deploy PPPoE in commercial using, until that all of the proved weak authentication protocols are abolished and replaced with more secure EAP authentication methods.

The rest of the paper is organized as follows. In section two, we introduce some preliminaries about PPPoE. We present some related work about password recovery attack in section three. We show how to hijack the PPPoE connection and get the peer's password by negotiating PAP protocol. And in section five, we propose an efficient password recovery attack to CHAP. We conclude the paper in the last section.

2 Preliminaries

2.1 PPPoE

PPP over Ethernet (PPPoE) provides the ability to connect a network of hosts over a simple bridging access device such as switch to a remote Access Concentrator. With this model, each host utilizes its own PPP stack and the user is presented with a familiar user interface. With PPPoE, the access concentrator has the ability to implement access control, billing and type of service on a per-user, rather than a per-site, basis [9]. PPPoE is widely used by the commercial ISPs to provide customers with Internet accessing and billing counting.

A simple example of the PPPoE network topology used by some ISPs is shown in Fig. 1. A switch connects all hosts of a building or even a community. A PPPoE server, connecting to the router, authenticates the legal peers and let them surf the Internet through the router. There may be an aggregation switch between switch and router depending on the service scale of the ISP. A PPPoE server may reside on the router in a small ISP's PPPoE network.

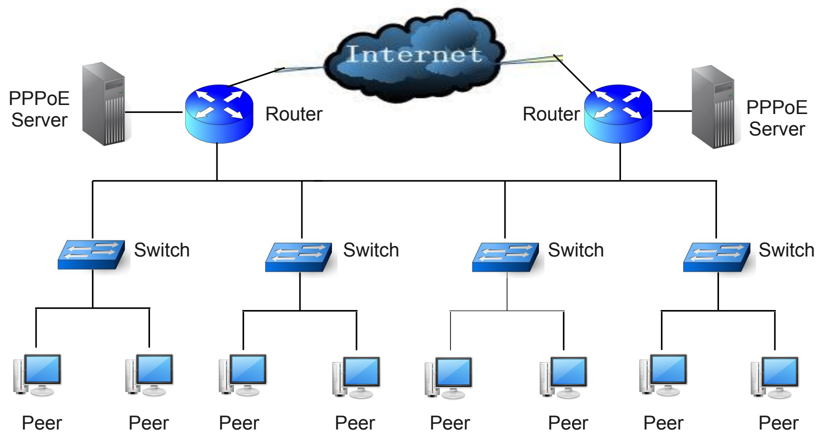


Fig. 1. A simple example of the PPPoE network topology

PPPoE has two distinct stages, PPPoE discovery stage and PPP session stage. The target of the PPPoE discovery stage is to collect needed information for staring the PPP session stage. For example, the Ethernet address of the remote server.

PPPoE Discovery Stage If a peer wishes to connect to the PPPoE access concentrator (the authentication server) to establish a valid PPP connection for surfing the Internet, he must first initiate a PPPoE discovery stage. In this stage, the peer discovers the server, which may be more than one depending on the network topology. The peer can identify the Ethernet MAC address of the server and establish a PPPoE session ID which can be further used in the following PPP session stage. When this stage is finished, both the peer and the server get enough information for establishing a valid PPP session.

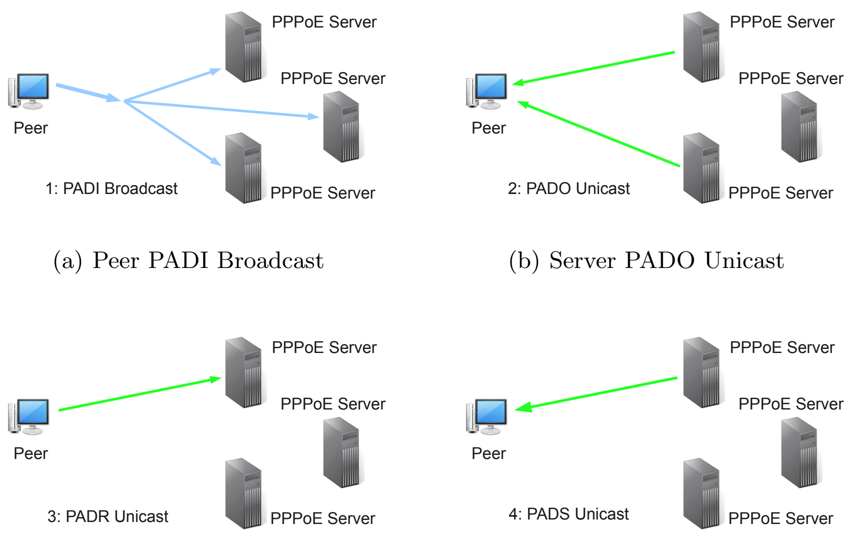
The PPPoE discovery stage consists of four steps, which are shown as follows.

1. The peer broadcasts a PADI (PPPoE Active Discovery Initiation) packet with code 0x09 to the network, which is shown in Fig.2(a). The PADI packet includes a destination address of the broadcast address, a session ID 0x0000, and may further includes a service name of the intended server.
2. All PPPoE authentication servers in the network receive the PADI packet. Each server may check the service name in the packet to decide whether or not to reply a PADO (PPPoE Active Discovery Offer) packet with code 0x07, which is shown in Fig.2(b). The PADO packet includes a destination address set to the peer's unicast address, a session ID 0x0000, and provides AC-Name and service name.
3. The peer may receive more than one PADO packet, since the PADI packet was broadcast. The peer chooses one of the authentication server, according to the AC-Name or the service name provided in the PADO packet. In most implementation of PPPoE, the peer simply chooses the first arrived PADO packet to reply, which is shown in Fig.2(c). The peer sends a PADR (PPPoE Active Discovery Request) packet with code 0x19 to the chosen server. The PADR packet includes the destination address of the server's unicast address, a session ID 0x0000, and provides one tag of the requesting service name.
4. When the peer-chosen authentication server receives the PADR packet, it prepares to begin a PPP session. The server generates a unique session ID for the PPPoE session and replies to the peer with a PADS (PPPoE Active Discovery Session-confirmation) packet with code 0x65, which is shown in Fig.2(d). The packet includes the destination address set to the peer's unicast address and the unique generated session ID.

In some situations, the server or the peer may send a PADT (PPPoE Active Discovery Termination) packet to terminate the connection.

PPP Session Stage After the completion of the PPPoE discovery stage, the PPP session stage starts. Link Control Protocol (LCP) of PPP will do some configurations about the connection, such as negotiating authentication method. After that, both sides may authenticate each other. The phases of the PPP session stage are listed as follows.

1. **Link establishment phase.** This phase is where Link Control Protocol (LCP) negotiations such as authentication method are attempted. If successful, control goes to the authentication phase.
2. **Authentication phase.** In this phase, both sides may authenticate each other depending on the negotiated authentication protocol, which will be discussed in detail later. If the authentication succeeds, control goes to the network-layer protocol phase.
3. **Network-Layer protocol phase.** This phase is where each desired protocols' Network Control Protocols are invoked. For example, IPCP is used in establishing IP service over the line. Data transport for all protocols which



(a) Peer PADI Broadcast (b) Server PADO Unicast

(c) Peer PADR Unicast (d) Server PADS Unicast

Fig. 2. The process of PPPoE discovery stage

are successfully started with their network control protocols also occurs in this phase. Only in this phase the peer can surf the Internet.

4. **Link termination phase.** This phase closes down the connection. This can happen if there is an authentication failure, or if the peer decides to hang up his connection.

2.2 Authentication Protocols of PPP

There are four kinds of authentication protocols in PPP, they are PAP, CHAP, MS-CHAP and EAP, in fact the last one is an authentication framework.

PAP The Password Authentication Protocol (PAP) is a simple method using passwords in clear text [8]. It is used by the PPP server to validate the peer before allowing him accessing to the server's resources. The peer repeatedly sends a username/password pair in clear to the authentication server until the authentication is acknowledged or the connection is terminated, which is shown in Fig.3. This is done only upon the completion of the initial link establishment. The value of the protocol field of PAP in the PPP frame is 0xC023.

CHAP The Challenge-Handshake Authentication Protocol [16] is an authentication scheme used by the PPP server to validate the identity of a remote

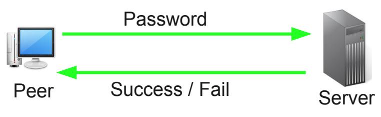


Fig. 3. PAP Authentication

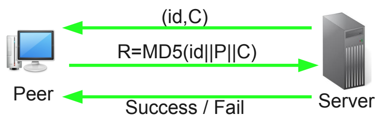


Fig. 4. CHAP Authentication

peer through a shared password P. Unlike PAP, the password is further hashed, using a random challenge C and an id both sent by the server, in order not to be transported in clear through an unsafe network.

As shown in Fig.4, the server verifies the peer using a 3-way handshake, which first happens after the completion of the link establishment phase, and may happen again at any time afterwards. The details of the 3-way handshake authentication are shown as follows [16].

1. The server sends a random challenge string C and a unique id to the peer.
2. The peer responds a R calculated using a one-way hash function on id, C and P combined.
3. The server checks the response R against its own calculation of the expected hash value. If the values match, the server acknowledges the authentication; otherwise it should terminate the connection.
4. At random intervals, the server sends new C' and id' to the peer and repeats steps 1 to 3.

There are four kinds of CHAP packets shown in Table 1, which stand for challenge, response, success and failure, respectively. The value of the protocol field of CHAP in the PPP frame is 0xC223.

Table 1. Four kinds of CHAP packets

Description	Sender	1 Byte	1 Byte	2 Bytes	1 Byte	Variable	Variable
Challenge	Server	Code 1	id	Length	C Length	C Value	Name
Response	Peer	Code 2	id	Length	R Length	R Value	Name
Success	Server	Code 3	id	Length		Message	
Failure	Server	Code 4	id	Length		Message	

The response R is calculated with MD5 [12] using the identifier id and the challenge C both sent by the server, and the shared password P. The calculation is shown as follow, in which || means concatenation.

$$R = MD5(id||P||C)$$

With random id and C chosen preliminariesby the authentication server, CHAP provides protection against re-play attack launched by malicious peer.

The use of repeated challenges is intended to limit the time of exposure to any single attack.

Brief description of MD5 MD5 [12] is a typical Merkle-Damgård structure hash function, which takes a variable-length message M as input and outputs a 128-bit hash value.

M is first padded to be multiples of 512 bits, a '1' is added at the tail of M, followed by '0's until the bit length becomes 448 on modulo 512, and finally, the length of the unpadded M is added to the last 64 bits.

The padded M' is further divided into 512-bit blocks $(M_0, M_1, ..., M_{n-1})$. The MD5 compression function (CF) takes M_i and a 128-bit chaining variable H_i, initialized to H_0, as input, and outputs H_{i+1}. For example, $H_1 = CF_{H_0}(M_0)$, and $MD5(M) = H_n = CF_{H_{n-1}}(M_{n-1})$. We omit the details of the CF, since it is irrelevant to this paper.

Padding rule. Let $|M|$ be the length of M. For two arbitrary distinct messages M and M', if $|M|$ equates $|M'|$, then, the padding bits of these two messages are just the same. This padding rule is applicable to MD5, MD4 and SHA-1 et al., since they share the same padding procedure.

Length extension attack. For an unknown message M and its corresponding hash $R = MD5(M)$, if we know $|M|$, we can compute the padding bits pad of M. For arbitrary message x, we can compute the padding bits $pad1$ for $M||pad||x$, then we can generate the hash value $MD5(M||pad||x)$ by computing $CF_R(x||pad1)$, without any knowledge of M except its length.

MS-CHAP MS-CHAP is the Microsoft version of the CHAP protocol, which exists in two versions, MS-CHAPv1 [20] and MS-CHAPv2 [19]. MS-CHAPv1 is enabled by negotiating CHAP Algorithm '0x80' ('0x81' for MS-CHAPv2).

Compared with CHAP, MS-CHAP provides an authenticator-controlled password change mechanism, and an authenticator-controlled authentication retry mechanism. MS-CHAPv2 provides mutual authentication between the peer and the server by piggybacking a peer challenge on the Response packet and an authenticator response on the success packet.

MS-CHAPv2 was introduced since some fatal flaws had been found in MS-CHAPv1 [13]. However, several weaknesses have also been found in MS-CHAPv2 [4], some of which severely reduce the complexity of brute-force attacks to the shared password making it unsuitable for using.

EAP Extensible Authentication Protocol [3] is a widely used authentication framework frequently used in PPP connections and 802.1X wireless networks. EAP provides some common functions and negotiation of authentication methods called EAP methods, such as EAP-MD5, EAP-OTP, EAP-GTC, EAP-TLS, EAP-IKEv2, EAP-SIM, EAP-AKA and the new proposed EAP-EKE [14]. The EAP method EAP-MD5 is analogous to CHAP [3].

Generally speaking, the security of the EAP-based applications is dependent on the security of the underlying EAP method.

2.3 Context-Free Grammar

Context-free grammars have long been used, in the study of natural languages [6, 11, 5], to generate (or parse) strings with particular structures.

A context-free grammar is quadruple form of $G = (V, \Sigma, \mathbf{S}, P)$, in which V is a finite set of *variables* or *non-terminals*, Σ is finite set of *terminals*, $\mathbf{S}$ is the *start variable*, and P is a finite set of *productions* with the form shown as follow.

$$\alpha \to \beta$$

α is a single variable and β is a string consisting of variables and(or) terminals. The language of the grammar is the set of strings consisting of all terminals derivable from the start symbol.

Probabilistic context-free grammars simply have probabilities associated with each production. For a specific left-hand side (LHS) variable, all probabilities of the associated productions are added up to 1. With the probabilities, the grammar employs the next production in the decreasing probability order to generate the language, instead of random order.

3 Related Work

3.1 Previous Work Related to CHAP Attack

Cheating CHAP Attack In 2002, Krahmer proposed an MITM attack to cheat the CHAP of PPP [7] in a hub-based network, the overall strategy is shown as follows.

1. The attacker *Eve* eavesdrops the communications between the PPP server and the peer to get enough information about them.
2. Eve tries to connect to the server and gets a challenge C and an id both sent by the server.
3. Eve masquerades the server and sends the received C and id to the peer.
4. Eve gets the response R from the peer.
5. Eve sends R to the server and establishes a valid connection to the server.

However, this attack only works in the hub-based LAN, and in fact, most of the LANs are switch-based.

Length Recovery Attack of $MD4(P||C)$ In 2008, Wang et al. presented a method to recover the length of the password P used in $MD4(P||C)$ oracle [17].

Let len and pad be the length and the padding bits of $P||C$, the value of the pad only depends on the len, according to the padding rule of the MD4 family hash functions. First, an attacker randomly generates a challenge C, sends it to the oracle of $MD4(P||C)$, and gets a response R. Second, the attacker generates $C' = C||pad||r$, where r is randomly generated, sends C' to the oracle, and gets a response R'. Finally, the attacker computes $pad1$ for C', and checks if $R' = CF_R(r||pad1)$ holds, where CF is the compression function of MD4. If so, the guess of len is correct. Otherwise, the guess is wrong. For a password P with l characters, it needs only $2l$ queries to recover l.

3.2 Password Cracking using Probabilistic Context-Free Grammar

In 2009, Weir et al. proposed an efficient way, using probabilistic context-free grammars to generate password structures in highest probability order, to perform a dictionary-based password cracking [18].

The set of passwords is a subset of the natural language. To define the password structure, let an *alpha string L* be a sequence of alphabet symbols, a *digit string D* be a sequence of digits, and a *special string S* be a sequence of non-alpha and non-digit symbols. For the password "$password123", the simple structure is defined as SLD. The base structure is defined similarly but also captures the length of the observed substrings, in the example this would be $S_1L_8D_3$.

The information of the base structures and their corresponding probabilities is derived from some training sets, which are publicly disclosed password lists. Based on the collected information, the probabilistic context-free grammar could be automatically created, and further be applied to generate word-mangling rules in the highest probability order in the password cracking.

4 Breaking PPPoE Network

In this section, we show how to break PPPoE network by recovering the peer's password. We achieve this target through three steps. First, we need to collect enough information about the authentication server and the peers. Second, we silently listen to the network and hijack peers' PPPoE connections. Finally, we get peers' passwords in clear by negotiating PAP authentication protocol.

4.1 Information Collection in the PPPoE Discovery Stage

When a peer wishes to initiate a PPPoE session, it must first perform a PPPoE discovery stage to identify the Ethernet MAC address of the authentication server and establish a PPPoE session ID. Based on the network topology, there may be more than one server that the peer can communicate with. The discovery stage allows the peer to discover all servers and then select one. When discovery stage completes successfully, both the peer and the selected authentication server have the information they will use to build their point-to-point connection over Ethernet.

Active Information Collection In order to collect information about the PPPoE authentication servers in the same LAN with victim peers, we need to initiate a PPPoE discovery stage where the authentication server will reply with some useful information. The details of the active information collection are shown as follows.

1. Send a PADI packet with a "blank" service name since we can't know it in advance, which will be broadcast to the entire network.

2. Receive a (or some, this depends the topology of the network) PADO packet, which includes the information about the authentication server such as MAC address, AC-Name, and the service name.

Since the PPPoE discovery stage is stateless, we don't need to send termination packet to any server, and the server will know nothing about this information collection. Through the above active process, we can collect needed information about the authentication server. But a server may not reply if there is not a right service name in the received PADI packet, we must find another way to collect enough information in this case.

Passive Information Collection We notice that if there is a PPPoE authentication server who checks the service name in the received PADI packets, then there are also some legal peers knowing the exact service name. So, we can silently listen to the network to eavesdrop such peers' PADI packets including service name, instead of sending one then waiting for the server's reply. In this way, we learn the server's service name, and both the peer and the server will not detect that, for the PADI packet is always broadcast. But is this enough? Of course not, with the eavesdropped service name, we must redo the active information collection to get everything about the server.

4.2 Hijack the PPPoE Connection

After we collect enough information about the PPPoE authentication server, we can hijack the peers' PPPoE connection to recover their passwords, by masquerading the server. The overall strategy is shown as follows.

1. Change the MAC address of our network card to the authentication server's. Set the information of the AC-Name and the service name to the legal server's. Listen to the network and wait for the PADI packets.
2. If a PADI packet arrives, immediately reply a PADO packet to the peer with the faked information.
3. Since the authentication server is always far from the peers' LAN, our faked PADO packet will arrive first at the peer, then the legal server's PADO packet follows. For the PADO packet we sent provides the right information and arrives first, the peer will immediately reply to us with a PADR packet. Hence, the legal server's PADO packet will be ignored by the peer.
4. After the PADR packet is received, send a PADS packet to the peer with a unique session ID to establish the PPPoE connection. The PPPoE connection hijacking is finished.

4.3 Get Password in Clear by Negotiating PAP in PPP Session

Since we have established a valid PPPoE connection with the peer, we can negotiate with him for using PAP in the followed PPP session. Then easily, we get the peer's password in clear. After the password is received, we terminate the PPPoE connection. The peer will be notified and may try to reconnect, we can simply ignore the PADI packet sent by the same MAC address whose password has been recovered. With this tactic, the PADI packet will be handled by the legal authentication server, hence the peer will establish a valid PPPoE connection to surf the Internet. The peer will have no idea that his password has been leaked to us.

Practical Analysis Most of the prevalent operating systems, such as "Windows 7" and "Ubuntu 10.10", support the PAP authentication protocol in the default setting of PPPoE security. Hence, the easiest way to break PPPoE network is that we fist hijack the peers' PPPoE connections by masquerading the server and then negotiate for using PAP in PPP session.

Most of the PPPoE authentication servers allow the users' multiple connection, which means that an attacker can freely use the legal peer's identity and password to surf the Internet without pay anything, and both the ISPs and the peers will not notice that the PPPoE accounts are under used maliciously. In some situations, a peer's account may be bound with his MAC address, this problem can be solved easily that we clone the victim's MAC address for we already know it.

We have implemented this kind of attack in some commercial ISPs' PPPoE networks. During our experiments, we find that most of the peers keep the default passwords, which are assigned by their ISPs, unchanged. After we collect enough passwords, we notice that some ISPs assign really simple passwords for the peers, which just consist of 6 characters of digits.

5 Breaking CHAP

Though we can always succeed in getting passwords in clear, when we can negotiate the PAP authentication protocol with the peer. Sometimes, the PPPoE clients may not accept PAP as authentication method. In this section, we propose an efficient password cracking method to break the CHAP authentication protocol used in PPPoE.

MD5 and MD4 share the same padding procedure, and id and C used in CHAP are both sent by the server. It means that the *length recovery attack* to $MD4(P||C)$, with some modification, is also applicable to CHAP, if we can impersonating the server. After the length of the used password is recovered, we employ a high efficient password cracking using probabilistic context-free grammars to break it. So, we divide the password cracking of CHAP into two parts, the first is to recover the length of the used password by on-line queries, the second is to crack the password with known length off-line, using a probabilistic context-free grammar based on fixed password length.

5.1 On-Line Length Recovery Attack to CHAP

CHAP employs an identifier id, a challenge C and a shared secret P to generate response. We have full control over id and C sent to the peer, since we can easily impersonate the authentication server in the PPPoE network. We implement the *length recovery attack* to CHAP by keeping the id unchanged in each on-line query. The overall strategy is shown as follows.

1. Randomly generate an id and a C, send them to the peer as the authentication challenge.
2. Receive the response $R = MD5(id||P||C)$ from the peer, reply an authentication acknowledgment signal to the peer.
3. Initiate the length of P with a *guessing* $l = 1$.
4. Generate pad for $id||P||C$ with the *guessing* l of P.
5. Generate $C' = C||pad||r$, where r is randomly generated, and send it with the unchanged value id to the peer.
6. Receive R' from the peer, reply an affirmative signal to it, generate $pad1$ for C' and check if $R' = MD5_R(r)$ holds. If so, l is the very length. Otherwise, set $l = l + 1$, go to step 4.

The peer will reply any challenge from the faked server during the CHAP authentication, since we masquerade the legal authentication server in the PPPoE connection.

In order to recover the password's length l, we need to send $l+1$ packets to the peer with different challenges but the same id, and $l + 1$ packets of affirmation. The time of the *length recovery attack* to CHAP is negligible. We send the challenges one by one continuously in only one session, for the server has full control over the time interval to launch another authentication by sending a challenge.

Once the length of password is recovered, we terminate the PPPoE connection. Since the peer will be notified, and he/she may try to reconnect to the server, we simply keep quiet and let the legal PPPoE authentication server to handle the PADI packet sent by the peer. With this tactic, the peer will establish a real connection with the server after a "normal" termination of the first try, both the peer and the server will not notice the *length recovery attack* we have launched.

5.2 Off-Line Fixed Length Password Cracking

In last subsection, we showed how to successfully recovered the length of the password used in CHAP. After studying some publicly disclosed password lists [2, 1, 10], we notice that passwords with different lengths have different base structures. We can improve the password cracking by combining the advantage of the efficient password cracking method using probabilistic context-free grammars [18] and the known password length.

First, we collect the base structures of the fixed length passwords and their corresponding probabilities from publicly disclosed password lists [2, 1, 10]. For example, in Table 2, we list the base structures and their corresponding probabilities of the disclosed passwords with 6 characters, which are mostly used. In the table, the superscripts mean multiples of concatenation.

Table 2. Example probabilistic context-free grammar with 6 characters

LHS→RHS	Probability	LHS→RHS	Probability
$\mathbf{S} \to L_6$	0.8826	$\mathbf{S} \to (D_1L_1)^3$	0.0027
$\mathbf{S} \to S_6$	0.0018	$D_1 \to 1$	0.8137
$\mathbf{S} \to D_6$	0.0158	$D_1 \to 2$	0.0891
$\mathbf{S} \to L_5D_1$	0.0739	$D_1 \to 3$	0.0567
$\mathbf{S} \to L_4D_2$	0.0027	$D_1 \to 6$	0.0162
$\mathbf{S} \to L_3D_3$	0.0079	$D_1 \to 7$	0.0081
$\mathbf{S} \to L_2D_4$	0.0009	$D_1 \to 8$	0.0081
$\mathbf{S} \to (L_2D_1)^2$	0.0009	$D_1 \to 9$	0.0081
$\mathbf{S} \to L_1S_1D_4$	0.0018	$D_2 \to 10$	0.2500
$\mathbf{S} \to (L_1D_1)^3$	0.0027	$D_2 \to 23$	0.2500
$\mathbf{S} \to D_3L_3$	0.0018	$D_2 \to 45$	0.2500
$\mathbf{S} \to D_2L_3D_1$	0.0009	$D_2 \to 69$	0.2500
$\mathbf{S} \to D_1L_5$	0.0027	$D_3 \to 111$	0.0910
$\mathbf{S} \to D_1L_2D_2L_1$	0.0009	$D_3 \to 123$	0.9090

Second, based on these base structures and their corresponding probabilities, we automatically create a group of probabilistic context-free grammars, each grammar based on fixed length passwords. The grammars employ the next production in the highest probabilistic order. In Table 2, we list some production rules of one probabilistic context-free grammar with associated probabilities.

Finally, with the known id, C, R, the recovered password length l and a chosen dictionary file, we use probabilistic context-free grammar to generate word-mangling rules in the highest probability order to greatly improve the password cracking.

With this tactic, we can always crack passwords faster and more than any other password cracking tools. This happens for the known length of the password and part for the distinct word-mangling rules used in the passwords with different lengths.

Time and Space Complexity Analysis. For a base structure like $L_xS_yD_z$ in a chosen training set, the numbers of production rules of S_x and D_z are fixed, let them be n_{sy} and n_{dz}, respectively. The number of words of length x, n_{lx}, can be calculated in a known dictionary file. Hence, the total search space for $L_xS_yD_z$ can be calculated by $n_{lx} \times n_{sy} \times n_{dz}$. The maximum time to find the truth of the intended password is totally dependent on the used dictionary file. Let N be the maximum password length in the training set, for each password with n ($1 \le n \le N$) characters, let the number of the base structures with L_i ($0 \le i \le n$) be bs_i, and the number of the words with i characters in the dictionary file be d_i, in which $d_0 = 1$. The maximum time of cracking a password with n characters is $\sum_{i=0}^{n} bs_i \times d_i$. The maximum time of cracking a password is $\sum_{n=1}^{N}\sum_{i=0}^{n} bs_i \times d_i$ in [18], since there is no information available about passwords' length, all base structures with different lengths must be searched. Hence, the space complexity of ours and [18] are $\sum_{i=0}^{n} bs_i \times d_i \times n$ bytes and $\sum_{n=1}^{N}\sum_{i=0}^{n} bs_i \times d_i \times N$ bytes, respectively. The time and space complexity between our work and [18] are compared in Table 3.

Table 3. Time and space complexity compared with [18]

Features	Ours	[18]
Known password length	Yes	No
Fixed length grammar	Yes	No
Time complexity	$\sum_{i=0}^{n} bs_i \times d_i$	$\sum_{n=1}^{N}\sum_{i=0}^{n} bs_i \times d_i$
Space complexity (bytes)	$\sum_{i=0}^{n} bs_i \times d_i \times n$	$\sum_{n=1}^{N}\sum_{i=0}^{n} bs_i \times d_i \times N$

During our password cracking experiments in the PPPoE network, we also notice the distinct probabilities (of %) of different password lengths, which are mostly commonly used, we list them in Table 4.

Table 4. The probabilities of passwords with different lengths

Length	3	4	5	6	7	8	other
Probability	2.56	9.4	17.54	36.0	18.78	12.51	3.21

6 Conclusion

We show that the PPPoE discovery stage has a serious security hole, which can be easily used by malicious users to hijack peers' PPPoE connections. Most of the prevalent operating systems support the PAP authentication protocol in the default setting of the PPPoE security. It is easy to get the peers' passwords by negotiating PAP in the PPP session stage. More seriously, the commercial ISPs will not know that those PPPoE accounts are used by malicious users.

We also propose an efficient way to break the CHAP authentication protocol used in PPP. It seems that all authentication protocols in PPP except EAP framework are broken, and EAP can be easily by-passed through negotiating for the broken protocols such as CHAP or MS-CHAP.

Finally, we point out that the PPPoE can't be used any more until all of the weak authentication protocols including PAP, CHAP, MS-CHAP are abolished right now and replaced with more secure EAP methods.

Acknowledgement. This work was partially supported by the National High Technology Research and Development Program of China (2009AA01Z437), the

National Key Basic Research Program of China (2007CB311100) and the Opening Project of Key Lab of Information Network Security, Ministry of Public Security.

References

1. A list of popular password cracking wordlists (2005), http://www.outpost9.com
2. John the Ripper password cracker (2011), http://www.openwall.com
3. Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., Levkowetz, H.: Extensible Authentication Protocol (EAP). RFC 3748 (Proposed Standard) (Jun 2004), http://www.ietf.org/rfc/rfc3748.txt, updated by RFC 5247
4. Baumgart, R., Schneier, B., Mudge, Wagner, D.: Cryptanalysis of Microsofts PPTP Authentication Extensions (MS-CHAPv2). In: Secure Networking CQRE [Secure] 99, Lecture Notes in Computer Science, vol. 1740, pp. 782–782. Springer Berlin / Heidelberg (1999)
5. Chomsky, N.: Three models for the description of language. Information Theory, IRE Transactions on 2(3), 113–124 (1956)
6. Hopcroft, J., Motwani, R., Ullman, J.: Introduction to automata theory, languages, and computation, vol. 3. Addison-wesley Reading, MA (1979)
7. Krahmer, S.: Cheating CHAP. http://dl.packetstormsecurity.net (2002)
8. Lloyd, B., Simpson, W.: PPP Authentication Protocols. RFC 1334 (Proposed Standard) (Oct 1992), http://www.ietf.org/rfc/rfc1334.txt, obsoleted by RFC 1994
9. Mamakos, L., Lidl, K., Evarts, J., Carrel, D., Simone, D., Wheeler, R.: A Method for Transmitting PPP Over Ethernet (PPPoE). RFC 2516 (Informational) (Feb 1999), http://www.ietf.org/rfc/rfc2516.txt
10. McMillan, R.: Phishing attack targets Myspace users. http://www.infoworld.com (2006)
11. Rabiner, L.: A tutorial on hidden Markov models and selected applications in speech recognition. Proceedings of the IEEE 77(2), 257–286 (1989)
12. Rivest, R.: The MD5 Message-Digest Algorithm. RFC 1321 (Informational) (Apr 1992), http://www.ietf.org/rfc/rfc1321.txt, updated by RFC 6151
13. Schneier, B., Mudge: Cryptanalysis of Microsoft's point-to-point tunneling protocol (PPTP). In: Proceedings of the 5th ACM conference on Computer and communications security. pp. 132–141. CCS '98, ACM, New York, NY, USA (1998)
14. Sheffer, Y., Zorn, G., Tschofenig, H., Fluhrer, S.: An EAP Authentication Method Based on the Encrypted Key Exchange (EKE) Protocol. RFC 6124 (Informational) (Feb 2011), http://www.ietf.org/rfc/rfc6124.txt
15. Simpson, W.: The Point-to-Point Protocol (PPP). RFC 1661 (Standard) (Jul 1994), http://www.ietf.org/rfc/rfc1661.txt, updated by RFC 2153
16. Simpson, W.: PPP Challenge Handshake Authentication Protocol (CHAP). RFC 1994 (Draft Standard) (Aug 1996), http://www.ietf.org/rfc/rfc1994.txt, updated by RFC 2484
17. Wang, L., Ohta, K., Kunihiro, N.: Password recovery attack on authentication protocol MD4(Password||Challenge). In: Proceedings of the 2008 ACM symposium on Information, computer and communications security. pp. 3–9. ASIACCS '08, ACM, New York, NY, USA (2008)
18. Weir, M., Aggarwal, S., Medeiros, B.d., Glodek, B.: Password cracking using probabilistic context-free grammars. In: Proceedings of the 2009 30th IEEE Symposium on Security and Privacy. pp. 391–405. IEEE Computer Society, Washington, DC, USA (2009)

19. Zorn, G.: Microsoft PPP CHAP Extensions, Version 2. RFC 2759 (Informational) (Jan 2000), http://www.ietf.org/rfc/rfc2759.txt
20. Zorn, G., Cobb, S.: Microsoft PPP CHAP Extensions. RFC 2433 (Informational) (Oct 1998), http://www.ietf.org/rfc/rfc2433.txt